**A DISSERTATION REPORT ON**

**STOCK PRICE PREDICTION USING LSTM AND GRU**

SUBMITTED TO MIT SCHOOL OF ENGINEERING, LONI, PUNE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

**YASH KHODKE**

**(2813249)**

**UNDER THE GUIDANCE OF**

**Prof. Sonali Deshpande**



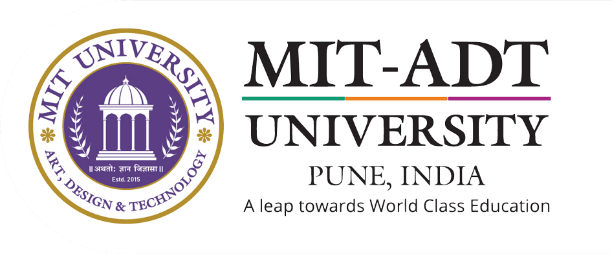
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**MIT SCHOOL OF ENGINEERING**

**RAJBAUG, LONI KALBHOR,**

**PUNE – 412201**

**JUNE 2023**



**MIT SCHOOL OF ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**RAJBAUG, LONI KALBHOR, PUNE**

**CERTIFICATE**

This is to certify that the dissertation report entitled

**STOCK PRICE PREDICTION USING LSTM AND GRU**

Submitted by

**YASH KHODKE (2183249)**

is a bona fide work carried out by her under the supervision of Prof. Sonali Deshpande and it is submitted towards the partial fulfillment of the requirement of MIT-ADT University, Pune for the award of the degree of Integrated Masters in Computer Science and engineering.

| Prof. Dr. Sonali Deshpande  Professor & Guide | Prof. Dr. Reena Gunjan  Professor & iMTech Project Coordinator,CSE | Prof. Dr. Shraddha Pasalkar  HOD, CSE |
| --- | --- | --- |
| Prof. Dr. Rajneeshkaur Sachdeo  Director, MIT SOE | Dr. R.V. Pujeri  Dean, MIT SOE | Place: Pune  Date: |



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**MIT SCHOOL OF COMPUTING,**

**RAJBAUG, LONI KALBHOR,**

**PUNE – 412201**

**EXAMINER’S APPROVAL CERTIFICATE**

The dissertation report entitled **“STOCK PRICE PREDICTION USING LSTM AND GRU”** submitted by **YASH KHODKE (MITU18BTCS0129)** in partial fulfillment for the award of the degree of **“Master of Technology in Computer Science and Engineering”** during the academic year 2022-23, of **MIT-ADT University, MIT School of Computing, Pune,** is hereby approved.

**Examiners:**

**1.**

**2.**

**3.**

**ACKNOWLEDGEMENT**

I sincerely express my deep sense of gratitude towards Head of Computer Science and Engineering Department **Prof Dr. Shraddha Pasalkar,** my respected guide **Prof. Sonali Deshapnde** and project coordinator **Prof. Dr Reena Gunjan** for their expert and valuable guidance and continuous help during completion of this work successfully.

I am very thankful to Hon. Director, **Prof. Dr. Rajneeshkaur Sachdeo** for all kind of permission and encouragement throughout the course.

I thank all the members of our Electronics and Communication Engineering Department and my colleagues for their assistance in the completion of this work. Last but not least, I am very thankful to my family members for their encouragement and moral support.

- Yash Khodke(MITU18BTCS0129)

**Abstract**

Given the complexity of the stock market's nonlinear movement system and its fluctuation rule, which is influenced by a wide range of circumstances, it could be challenging to make precise predictions regarding the stock price index. It could be challenging to predict the stock market's future as a result of these considerations. Numerous use cases have demonstrated the ability of neural network algorithms to predict time series properly and to consistently give results that are appropriate. Based on previously published models, we developed a Regularized GRU LSTM neural network model in this paper. Our forecasts were based on this model. This model allowed us to generate predictions for the short-term closing values of the two stocks. The ability of our proposed model to forecast stock time series has been demonstrated through experiments to be superior to both the GRU and LSTM network models. This benefit can be attributable to the model's improved data classification accuracy.It is challenging to predict the price of stocks due to the dynamic and complex structure of the financial markets. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) deep learning models have recently shown promise in capturing the temporal dependencies and nonlinear patterns in stock price data. These models are referred to as "deep neural networks." This project aims to build LSTM and GRU models, as well as stacked LSTM and GRU models, for the purpose of predicting stock values, and then compare the outcomes of these models. In addition, the study explores the potential benefits of combining traditional methods for time series analysis with deep learning models using autocorrelation analysis and ARIMA (AutoRegressive Integrated Moving Average) models.The project's initial stage is to gather historical stock price data, which is then preprocessed to ensure consistency and remove noise. The preprocessed data is used to train both the LSTM and the GRU models, and multiple layers are applied to capture hie archical representations of the temporal patterns. The goal of the stacked LSTM and stacked GRU models, each of which consists of numerous layers of LSTM or GRU, is to further improve the models' capacity for learning.

**TABLE OF CONTENT**

[**CERTIFICATE**](about:blank) **i**

[**ACKNOWLEDGEMENT**](about:blank) [**ii**](https://docs.google.com/document/d/1FNN9jlPpYErGWbafhHo8vaFEQJkoDarCHr6ASXAQJ3A/edit#heading=h.2et92p0)

[**ABSTRACT i**](about:blank)**ii**

[**TABLE OF CONTENT**](about:blank) **i**[**v**](https://docs.google.com/document/d/1FNN9jlPpYErGWbafhHo8vaFEQJkoDarCHr6ASXAQJ3A/edit#heading=h.tyjcwt)

[**LIST OF FIGURES**](about:blank) [**v**](https://docs.google.com/document/d/1FNN9jlPpYErGWbafhHo8vaFEQJkoDarCHr6ASXAQJ3A/edit#heading=h.3dy6vkm)

**1.** [**INTRODUCTION**](about:blank)**……………………………………………………………..11-22**

**[1.1 Motivation……………………………………………………................11-12](about:blank)**

**[1.2 Problem Statement…………………………………….…………………..12](about:blank)**

**1.3 Aims & Objectives……………………………………………………..12-14**

**1.4 Scope of Project Work………………………………………….……...14-16**

**2.** [**LITERATURE SURVEY………………………………………………….….**](about:blank)**17-22**

**3.** **METHODOLOGY ……………………………………………………………23-37**

[**3.1**](about:blank) **AutoCorrealtion……………………………………………………………23**

[**3.2 Auto**](about:blank)**-Regressive Integrated Moving…………………………………..24-25 Average(ARIMA)**

**3.3 Pre-Processing…………..........................................................................25-27**

**3.4 Rolling and Resampling………………………………………………..27-29**

**3.5 Differencing……………………………………………………………..29-30**

**3.6 Splitting Data Into Training and Testing Data Sets…………………………………………………………………………………30**

**3.7 Simple RNN……………………………………………………………..30-31**

**3.8 Long Short-Term Memory Networks…………………………………31-32**

**3.9 Gated Recurrent Unit…………………………………………………..33-34**

**3.10 An Illustration of the Model’s Architecture………………………………34**

**3.11 Software Description……………………………………………………35-36**

**3.12 Limitations………………………………………………………………36-37**

**4.** [**RESULTS DISCUSSION**](about:blank)**……………………………………………………..38-39**

**4.1 Results…………………………………………………………………..38-39**

**4.2 Discussion…………………………………………………………………..39**

**5.** [**CONCLUSION …………………………………………………………………**](about:blank)**...40**

**6 FUTURE WORK………………………………………………………………41-43**

[**REFERENCES…………………………………………………………….……..**](about:blank)**44-45**

**[APPENDIX A](about:blank)**

[**APPENDIX B**](about:blank)

**APPENDIX C**

**Table of Figures**

[Figure 1](#1fob9te) **Missing values in dataframe**

[Figure 2](#1fob9te) **Seasonal plot for open column in our data**

[Figure 3](#1fob9te) **Plot of 7-d rolling average which is a bit smoother than weekly average**

[Figure 4 **Block diagram of LSTM**](#1fob9te)

[Figure 5 **Architecture of GRU**](#3znysh7)

[Figure 6 **Block diagram of Model's Architecture**](#2et92p0)

[Figure 7 **Result obtained from stacked LSTM model**](#tyjcwt)

[Figure 8 **Result obtained from stacked GRU model**](#3dy6vkm)

**CHAPTER 1**

**INTRODUCTION**

A growing number of people have developed an interest in investing in the stock market in recent years because of the high rates of return it provides.Individuals and institutions can purchase and sell shares of publicly traded companies on the stock exchange. It is a central meeting place for investors to exchange securities, which represent ownership in companies. The stock market is essential to the economy because it provides companies with a means to raise capital and investors with opportunities to increase their wealth.Trends in the stock market refer to the general direction in which stock prices move over time. These tendencies can be divided into three categories:An uptrend exists when stock prices continue to rise over time. Positive market sentiment and investor confidence are indicated. Buyers outnumber vendors during an uptrend, resulting in increased demand and upward price movements.The opposite of an uptrend, a downtrend is characterized by a sustained decline in stock prices. It reflects a lack of investor confidence and a negative market sentiment. In a downtrend, sellers outnumber purchasers, causing a decrease in demand and price declines.Sideways (or Range-bound) Trend: A sideways (or range-bound) trend occurs when stock prices move within a relatively narrow price range with no discernible upward or downward direction. It indicates a period of consolidation or indecision on the market and suggests a balance between purchasers and sellers.Various factors influence stock market trends, including:Economic Conditions: Indicators such as GDP growth, inflation, and unemployment rates can have a substantial impact on stock market developments. Positive economic conditions typically support an uptrend, whereas poor economic performance can contribute to a downtrend.Performance of the Company: The financial performance and outlook of individual companies affect their stock prices and, in turn, market trends. Positive earnings reports, new product introductions, or successful acquisitions can contribute to an uptrend by driving stock prices higher.Investor Sentiment: Investor sentiment, which is influenced by market news, geopolitical events, and investor conduct, has a significant impact on stock market trends. Positivity can contribute to an uptrend, while pessimism can contribute to a downtrend.Monetary and Fiscal Policies: Actions taken by central banks and governments, such as changes in interest rates, quantitative easing, and fiscal stimulus measures, can influence stock market trends. These policies can affect interest rates, liquidity levels, and market conditions as a whole.Noting that stock market trends are not always predictable or linear is crucial. Market participants use various techniques, such as technical analysis, fundamental analysis, and market indicators, to analyze trends and make investment decisions, as they can be influenced by a variety of factors.Understanding stock market trends can assist investors in identifying potential investment opportunities, mitigating risks, and making well-informed decisions based on the prevalent market sentiment. Before making any investment decisions, it is essential to conduct extensive research and analysis, as stock market trends can change rapidly and are subject to numerous uncertainties.Despite the high amount of risk, both individual and institutional investors continue to place money in the stock market. As a result, both individual and institutional investors are interested in the projection for the index of stock prices. The predictability of the returns on stock investments has been a source of constant controversy in addition to the situation's inherent difficulties. Many forecasting and modelling approaches for stock price indices have been developed through research in a wide range of fields, including physics, economics, computer science, and statistics.

Predicting stock prices can provide investors and market participants with numerous advantages. Here are several important benefits of stock price forecasting:The ability to forecast stock prices enables investors to make more informed investment decisions. Investors can gain insight into potential future price movements by analyzing historical data, market trends, and other relevant factors. This information assists them in identifying stocks that are undervalued or overvalued, selecting prospective investment opportunities, and allocating capital more efficiently.

Accurate stock price forecasts aid in the management of investment risks. By anticipating price fluctuations, investors can protect their portfolios with risk management strategies. For instance, they can alter their asset allocation, diversify their investments across various sectors or asset classes, or employ hedging techniques to reduce the risk of potential losses.Timing the Market: Stock price forecasts can help determine the optimal time to purchase or sell stocks. By recognizing trends and patterns in stock prices, investors can schedule their transactions to maximize potential profits or minimize losses. Effective market timing can enhance investment returns and portfolio performance overall.

Active traders who engage in short-term trading or high-frequency trading may find it advantageous to forecast stock prices. Using price forecasts and technical analysis, traders are able to develop and refine trading strategies. These strategies may include momentum trading, swing trading, and arbitrage, all of which can profit from short-term price fluctuations.Portfolio Optimization: Predictions of stock prices can assist with portfolio optimization. Using risk-return tradeoffs, investors can use predictive models to optimize their portfolio allocations. By integrating price forecasts and other pertinent data, investors are able to construct portfolios with specific investment objectives in mind, such as maximizing returns or minimizing volatility.

Quantitative Analysis: Predicting stock prices frequently necessitates the application of quantitative analysis techniques. This strategy employs mathematical models, statistical methods, and machine learning algorithms to analyze and identify patterns in enormous amounts of data. By utilizing these techniques, investors can obtain insights that may not be readily apparent using conventional methods of analysis.Evaluation of Investment Strategies Predictions of stock prices allow investors to evaluate the efficacy of their investment strategies. Investors can determine the accuracy and dependability of their models by comparing actual stock performance to predicted prices. This evaluation procedure permits the ongoing enhancement and refinement of investment strategies.Notably, it is difficult to accurately forecast stock prices, and no method can guarantee absolute precision. Numerous factors, including economic conditions, market sentiment, and unexpected events, can introduce volatility and uncertainty to the stock market. When making investment decisions based on stock price predictions, it is crucial to exercise caution, diversify investments, and consider multiple information sources.

The Efficient Market Theory, put out by Fama in 1970, contends that a good's current value always fairly represents all of the information previously accessible. These models presuppose the time series values' linear correlation structure. They are built with this as their foundation. As a result, these models are unable to faithfully capture nonlinear patterns. To get around this limitation, neural network models are increasingly being used in the field of forecasting nonlinear time series, such as the stock price index. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are the RNN architectures with the best performance. Recurrent neural networks, or RNNs, have been shown to be among the most efficient methods for handling sequential data. The RNNs' propensity to recognize complicated nonlinear correlations, which are notoriously challenging for conventional forecasting models to understand, is largely responsible for this. Instead of the typical artificial neurons, memory cells, which serve as a processing unit, are substituted with LSTM nodes in the network's hidden layers. Networks now have the ability to dynamically record the data architecture and connect memories with incoming data thanks to these memory cells, which enhances prediction accuracy. Because the LSTM model has an output gate and the GRU model does not, the GRU and LSTM learning methods are fundamentally different from one another. The Bi-directional LSTM structure is one of the recently created improved models that is based on the two RNN structures. This particular model is simply one of several that have been created. This particular construction is also heavily utilized. The ability of LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) models to capture complex patterns and dependencies in time series data has increased their prominence for predicting stock prices. Here are the advantages of using LSTM and GRU to determine stock prices:

LSTM and GRU models are capable of identifying nonlinear relationships in stock price data. Various factors, including market trends, investor sentiment, and economic indicators, influence stock prices. LSTM and GRU models are capable of learning from historical data and identifying intricate patterns that may not be readily identified by linear models.Handling Long-Term Dependencies Stock price data frequently exhibits long-term dependencies, in which past prices can have a significant impact on future prices. LSTM and GRU models are intended to solve the vanishing gradient issue that can arise in conventional recurrent neural networks. They have memory cells that can retain and use information from previous time steps, allowing them to recognize long-term dependencies and make more precise predictions.Processing Sequential Data: Stock prices are sequential data where the order of observations is significant. LSTM and GRU models flourish at processing sequential data due to their recurrent nature. They are able to analyze historical price sequences, recognize patterns, and utilize sequential data for forecasting. This makes them suitable for capturing the inherent temporal dependencies in stock market data.LSTM and GRU models have the capacity to autonomously learn pertinent features from the data. Instead of relying on painstakingly crafted features, these models can extract crucial information from the raw input, such as historical prices, trading volumes, and news sentiment. This capability facilitates the modeling process by eliminating the need for domain-specific feature engineering.Stock markets are dynamic, and LSTM and GRU models have the ability to adapt to shifting market conditions. They are able to perpetually update their internal states based on new data and adjust their predictions accordingly. This adaptability makes them valuable in situations where the market environment changes over time, enabling investors to make more timely and accurate predictions.Forecasting Short-Term Price Changes: LSTM and GRU models are especially useful for forecasting short-term stock price movements. They are able to identify patterns and trends in recent price data, allowing traders to anticipate price movements over the next few time steps. Day traders or short-term investors seeking to capitalize on immediate market opportunities can benefit from short-term forecasts.

Accurate stock price forecasts can aid in portfolio optimization and risk management. By integrating predicted stock prices into portfolio models, investors can more effectively optimize asset allocation, evaluate risk-return trade-offs, and implement risk management strategies. These forecasts provide insightful information for portfolio rebalancing, risk diversification, and investment position adjustments.Noting that LSTM and GRU models offer advantages for stock price prediction, but are not infallible, is essential. Numerous unpredictable factors influence the stock market, and even the most sophisticated models may not always make accurate predictions. When making investment decisions, it is essential to combine predictive models with exhaustive market analysis, risk management techniques, and critical thinking.The LSTM network may produce better results when dealing with large datasets since it has more memory to store and interpret historical data. Since the GRU algorithm has less parameters than the LSTM algorithm, it is considerably faster. In this study, we offer a novel Regularized GRU-LSTM network model that, by merging LSTM and GRU, outperforms existing models. We predicted the ultimate values of two different stocks using this strategy.

**1.1 Motivation**

Developing an LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit) model to forecast stock prices can be prompted by a number of factors:

Potential Financial Gain Correctly predicting stock prices can result in potential financial gain. Investors can identify profitable trading opportunities, make informed investment decisions, and potentially generate higher returns on their investments by accurately predicting stock price movements. In the financial markets, the ability to forecast stock prices can be viewed as a competitive advantage. Predicting stock prices can aid in the management of investment risks. By comprehending prospective price fluctuations, investors can protect their portfolios with risk management strategies. Accurate forecasts can aid in identifying prospective market downturns, preventing substantial losses, and protecting against adverse market conditions. Market Knowledge and Insight: Constructing LSTM and GRU models for predicting stock prices requires a comprehensive understanding of the financial markets. The development of these models entails assessing historical data, identifying patterns and trends, and examining the interrelationships between the various factors influencing stock prices. This research and analysis contribute to a better comprehension of the complexities of stock markets by shedding light on market dynamics.

LSTM and GRU models for predicting stock prices necessitate the application of sophisticated machine learning techniques. Research, experimentation, and innovation in the field of predictive modeling are required to develop and refine these models. The difficulty of predicting stock prices motivates the development of machine learning algorithms and methodologies.

Personal Interest and Curiosity: The stock market is a domain that many individuals find enthralling. Developing LSTM and GRU models for predicting stock prices can be a mentally stimulating and interesting endeavor. It combines elements of finance, mathematics, statistics, and machine learning, offering a unique opportunity to explore and employ interdisciplinarity.As part of their academic studies or research projects, researchers and students in disciplines such as finance, economics, and data science may be motivated to develop LSTM and GRU models for stock price prediction. These models can contribute to the extant body of quantitative finance, time series analysis, and predictive modeling knowledge.Real-World Application Using LSTM and GRU models to predict stock prices has real-world applications that extend beyond individual trading and investment decisions. These models can be utilized by financial institutions, hedge funds, and investment firms for risk management, portfolio optimization, algorithmic trading, and market analysis. The development of accurate prediction models can have practical implications for these organizations.Notably, accurately predicting stock prices is difficult, and the financial markets are influenced by numerous complex factors. The success of LSTM and GRU models for predicting stock prices depends on a number of variables, including data quality, feature engineering, model architecture, and market conditions. Therefore, it is crucial to approach stock price forecasting with a realistic appreciation of its limitations and potential hazards.

**1.2 Problem Statement**

To predict the stock prices using updated Long Short-Term Memory and Gated Recurrent Unit. The main goal of this paper is to improve the accuracy of basic LSTM and GRU models and as well as consider other factors that contribute to the accuracy of the model.

**1.3 Aims & Objectives**

The goal is to create a clear, straightforward method that can be used to properly anticipate stock prices of a specific stock using historical data, and the objective is to compare the model's performance to that of more conventional statistical models. Depending on the specific project goals, the purpose and objectives of a model employing LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) for stock price prediction can vary. Nevertheless, the following are some common goals and objectives that such a model could encompass:

The purpose of the model is to accurately predict stock prices and provide valuable insights into future price movements, enabling investors and market participants to make informed decisions. Objectives:

1. Create and train GRU and LSTM models: Construct LSTM and GRU models capable of learning from historical stock price data and identifying complex data patterns and dependencies. Train the models using the appropriate techniques and algorithms for training.
2. Improve prediction accuracy: Experiment with various model architectures, hyperparameters, and feature engineering techniques to improve the accuracy of stock price predictions. Reduce prediction errors and enhance the model's ability to predict both short- and long-term price trends.

c. Assess model performance: Using appropriate evaluation metrics, assess the performance of the LSTM and GRU models. Examine the accuracy, precision, recall, and other applicable metrics by comparing the model's predictions to actual stock prices. Identify improvement opportunities and adjust the model accordingly.

d. Investigate feature engineering: Examine various features and data representations that can boost the predictive power of the models. This may involve integrating additional data sources, developing domain-specific features, or leveraging external factors such as news sentiment, economic indicators, and technical indicators.

e. Analyze market dynamics and patterns: Conduct an in-depth analysis of the dynamics and patterns of the stock market discovered by the models. Gain an understanding of the interrelationships between variables, the impact of market events, and the factors influencing stock price fluctuations. Determine how the models can aid in comprehending market trends and behaviors.

f. Evaluate model robustness and generalizability: Validate the trained LSTM and GRU models on unobserved testing data in order to evaluate their ability to generalize to new market conditions. Evaluate the efficacy of the models under various market conditions and identify any limitations or potential biases.

g. Evaluate against benchmark models: Evaluate the performance of the LSTM and GRU models in comparison to benchmark models or conventional stock price prediction methods. Evaluate the superiority or added value of the deep learning models in terms of accuracy, stability, and robustness of predictions.

Provide insights and suggestions for action: Transform the predictions and analysis into actionable insights for market participants, investors, and traders. Based on the anticipated stock prices and market dynamics, recommend investment strategies, risk management techniques, and portfolio adjustments.

i. Document and communicate findings Document the entire project's procedure, methodologies, results, and essential findings. Prepare a comprehensive report or presentation that summarizes the project's objectives, methods employed, data analysis, model performance, and any recommendations or repercussions derived from the study.

**1.4 Scope of Project Work**

In addition to stock price information, both LSTM and GRU models can easily include other variables. This additional data can take the shape of news stories, social media sentiment, economic indicators, or data related to a particular company. The models can increase their forecast accuracy and provide more thorough insights when these external inputs are incorporated into them.

Depending on the particular objectives and specifications, the scope of a project involving LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) models may vary. Nonetheless, the following elements may be included in the scope of a project involving LSTM and GRU models:Collect relevant historical stock market data, such as stock prices, trading volumes, and any other variables that may be helpful for forecasting. This information is accessible via financial data providers, stock exchanges, and online platforms.

Cleaning and preprocessing the collected data to assure its suitability for model training. This may involve dealing with missing values, removing outliers, normalizing or scaling the data, and separating it into training and testing sets.

Feature Engineering: Select or engineer features that can improve the predictive ability of the LSTM and GRU models. These characteristics may include lagging variables, technical indicators, data on market sentiment, or macroeconomic indicators. Techniques for feature selection and engineering can be used to determine the most informative variables.

Model Development: Develop LSTM and GRU models for stock price forecasting. This includes specifying the number of layers, the number of units in each layer, and the activation functions, as well as delineating the architecture of the models. The hyperparameters of a model, such as the learning rate, sample size, and regularization techniques, must be fine-tuned for optimal performance.

Model Training: Use the prepared training data to train the LSTM and GRU models. This involves loading historical data into models, optimizing model parameters iteratively, and updating weights to minimize prediction error. For model training, techniques such as backpropagation through time and gradient descent algorithms can be utilized.Evaluate the performance of the LSTM and GRU models using suitable evaluation metrics. Common stock price prediction metrics include mean squared error, root mean squared error, mean absolute error, and directional accuracy. To evaluate the efficacy of the developed models, comparisons can be made to benchmark models or baseline approaches.

Model Validation: Validate the LSTM and GRU models using unobserved test data to guarantee their generalization capability. This phase assists in determining whether the models can accurately predict stock prices in real-world scenarios and provides insight into potential overfitting or underfitting issues.

Utilize the trained LSTM and GRU models to make stock price predictions based on unobserved data. Assess the efficacy of the models by analyzing the forecasts, comparing them to actual stock prices, and evaluating the correlation between the two. Conduct additional analysis to comprehend the factors contributing to the predictive power of the model, and investigate any insights gleaned from the prediction results.

Documentation and Reporting: Document the entire endeavor, including data collection, preprocessing, model development, training, evaluation, and analysis. Create a detailed report that summarizes the project's objectives, methodology, findings, and recommendations.Moreover, depending on the project's scope and objectives, you may wish to expand the work to include advanced techniques such as ensemble modeling, feature selection algorithms, model interpretability methods, or the investigation of alternative deep learning architectures for stock price prediction.To increase the models' capacity to precisely anticipate stock values, researchers are always working to improve the LSTM and GRU models and are looking into new versions and modifications. These advancements, which include the incorporation of attention processes, hybrid models, or complex regularization techniques, all have the potential to significantly enhance the models' capacity for prediction.

**CHAPTER 2**

**LITERATURE SURVEY**

In order to make precise predictions about stock prices, the approach outlined in the research titled "Attention-based LSTM for Stock Price Prediction" [1] uses LSTM networks in combination with attentional processes. Based on their relative relevance, the attention mechanism gives various weights to different segments of the input sequence. This helps the model to focus on the most important facts. The accuracy and resilience of the authors' attention-based LSTM model beat baseline algorithms and conventional LSTM models, according to their research on real-world stock price datasets. These tests are run by the authors using datasets of actual stock prices. The study's findings provide a more precise prediction model, which helps financial forecasting.

The research paper "Stock Price Prediction Using LSTM and Extreme Gradient Boosting" [2] combines the LSTM and XGBoost algorithms with the goal of forecasting future stock values. The authors review earlier studies and underline the need for cutting-edge research methods. By utilizing the advantages of XGBoost for feature selection and ensemble learning as well as LSTM for determining temporal dependencies, their method increases the accuracy of predictions. Real-world datasets were used in experiments to show that the model outperforms more traditional methods. The study makes a substantial contribution to the discipline of financial forecasting and suggests using ensemble models for more precise predictions.

[3]This study examines the prediction of stock prices using the LSTM and GRU models along with a number of technical indicators. The authors emphasize the need of precise stock price forecasting and the difficulties that come with it. The models used in this study, LSTM and GRU, are well renowned for their efficiency in capturing temporal correlations. Additional features made up of technical indicators including volume, RSI, and moving averages have been proposed. Numerous investigations have demonstrated that the addition of technical indicators increases the precision of both LSTM and GRU models. By providing careful consideration to the use of technical indicators for the aim of developing more precise stock price estimates, this study makes a substantial contribution to the field of financial engineering.

[4]An investigation into the application of LSTM and GRU networks for stock price prediction is the subject of the topic "Predicting Stock Prices Using LSTM and GRU Networks". The authors examine the performance of several recurrent neural network topologies in capturing the temporal dependencies that can be observed in stock price data. The findings of the study's tests and analyses show how effective LSTM and GRU networks are at forecasting stock values with accuracy. By showcasing the potency of deep learning models for stock price forecasting, this work makes a contribution to the field of computational finance. A branch of computational economics is computational finance.

[5]The LSTM and GRU models are compared and contrasted in the paper named "Comparative Study of LSTM and GRU for Stock Price Prediction on Three Major Stock Indices" in order to forecast stock prices on three important stock indices. Title of the essay: "Comparative Study of LSTM and GRU for Stock Price Prediction on Three Major Stock Indices." The authors evaluate the temporal correlation detection and stock price forecasting capabilities of both models. This study significantly advances our understanding of the respective capabilities of the LSTM and GRU networks for stock price forecasting by highlighting the benefits and drawbacks that are exclusive to each type of network.

The idea of Long Short-Term Memory (LSTM), a kind of architecture for recurrent neural networks (RNN), is first introduced in the article titled "Long Short-Term Memory" written by Hochreiter and Schmidhuber (1997). LSTM uses a combination of memory cells and gating techniques to address the vanishing and ballooning gradient issues that plague standard RNNs. The authors give a thorough explanation of the design as well as the functionality of memory cells and gating units in their analysis of the LSTM architecture. The authors use a variety of tasks, including speech recognition and sequence prediction, to illustrate the value of LSTM. The introduction of LSTM as a useful modeling and capturing technique for long-term dependencies in sequential data makes this research noteworthy. This method can be used to model and keep track of intricate temporal interactions between variables.

A thorough empirical assessment of gated recurrent neural networks, or RNNs, for sequence modeling tasks is provided in the publication [7].In a 2014 study on sequence modeling, Chung, Gulcehre, Cho, and Bengio evaluated gated recurrent neural networks empirically. The authors analyze the performance of various gated RNN types, including Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), using a range of benchmark datasets. The findings show that gated RNNs outperform standard RNN architectures when it comes to recognizing long-term associations and excel remarkably in tasks requiring sequence modeling. This study advances knowledge of many gated RNN variations and their application to sequence modeling.

The "dropout" strategy is detailed in the academic paper "Dropout: A Simple Method to Prevent Overfitting in Neural Networks"[8]. In order to avoid overfitting, dropout includes the arbitrary deactivation of neural network units while the network is being trained. The authors show that dropout works with different kinds of neural networks and offer a theoretical basis for it. A regularization technique that has gained popularity in deep learning is dropout. Enhancing generality and preventing overfitting are its main goals.

[9]The optimization technique known as Adam is dissected in length in the essay "Adam: A Method for Stochastic Optimization" by Kingma and Ba (2015). Adam is a well-known technique for improving stochastic gradient descent algorithms' effectiveness. The term is an abbreviation for "Adaptive Moment Estimation." The authors offer a method that combines the benefits of the RMSProp and AdaGrad algorithms into a single, well-rounded technique. Adam adjusts each parameter's learning rates in accordance with the values of the gradient squared and past gradients. The results show that Adam performs better across a wide range of deep learning tasks than other optimization methods, allowing for a speedier convergence. The research has gained a lot of attention in the deep learning community since it was first published, particularly as an optimization tool.

A thorough examination of the deep learning techniques used for stock price prediction is provided in the article [10] titled "A Comprehensive Review of Deep Learning Techniques for Stock Price Prediction" by Preethi and Balasubramanian (2020). This analysis goes into great detail. The three deep learning models that the authors use in this field of study are convolutional neural networks (CNNs), long short-term memory (LSTM) networks, and recurrent neural networks (RNNs). They talk about each model's benefits and drawbacks as well as potential applications for forecasting stock prices. Researchers and practitioners who are interested in using deep learning to predict stock values should find the paper's discussion of difficulties and potential future paths for the discipline interesting.

The writers of the article [11] "Stock Price Prediction Using Deep Learning: A Survey" by Zhang, Zhou, and Zhang (2021) present a thorough analysis of the deep learning methods used for stock price prediction. Some of the deep learning models covered by the authors are Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and its derivatives Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). They go over the various techniques that were employed in the research, including the use of technical indicators, social media sentiment analysis, and the exploitation of textual data. The study includes the evaluation standards and datasets that are frequently used in stock price prediction research. Researchers and practitioners interested in deep learning-based stock price prediction systems will find this article to be a useful resource.

The authors of Raza, Khan, and Amin's article "A Comprehensive Survey of Deep Learning Models for Stock Market Prediction" (2021)[12] provide a thorough analysis of the deep learning models that are employed for stock market prediction. Autoencoders, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversarial networks (GANs) are only a few of the deep learning techniques covered by the writers. We discuss the benefits, drawbacks, and potential applications of each stock market prediction model. The research covers a wide range of additional topics in addition to data pretreatment, feature selection, evaluation metrics, and widely used datasets. Researchers and practitioners interested in using deep learning models to forecast the stock market will find this article to be a useful resource.

[13]Liu, Chen, Li, and Gao's 2019 article "Stock Price Prediction Using Deep Learning: A Review" offers a thorough analysis of the deep learning methods that are used to predict stock prices. Some of the deep learning models described in this article are Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and their derivatives like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). They go over a variety of stock price forecasting-related subjects, including feature engineering, model design, data preparation, and assessment metrics. Additionally, the difficulties and probable future directions of this field of study are highlighted. For experts and researchers who want to utilize deep learning to forecast stock prices, it is a beneficial tool.

A thorough examination of the use of deep learning algorithms for stock price forecasting is provided in Shinde and Shinde's paper from 2021, "A Comprehensive Review of Stock Price Prediction Using Deep Learning" [14]. This analysis is offered as a thorough evaluation. The authors examine a variety of deep learning models, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and their modifications, such as long short-term memory (LSTM) and gated recurrent unit (GRU). Long short-term memory (LSTMs) and neural networks with gated recurrent units (GRUs) are more examples. They cover a range of methods for estimating stock values, such as the use of technical indicators, sentiment analysis of news articles, and social media data collection. Here, data preparation techniques, evaluation measures, and datasets that are often used are all presented. For scholars and practitioners interested in deep learning-based stock price forecasting methods who are also interested in stock price forecasting methodologies, this paper is an excellent resource.

1. Nguyen, Nguyen, Nguyen, and Hoang's essay "A Comprehensive Review of the Applications of Deep Learning in Stock Price Prediction" (2022) offers a thorough evaluation of the many deep learning applications in stock price prediction. The application of deep learning models for stock price prediction, including CNNs, RNNs, and LSTM networks, is examined and summarized by the authors. The authors concentrate on these models' derivatives in particular. By applying a variety of datasets, including the use of textual data, sentiment analysis, and technical indicators, among others, they analyze various approaches and assess the efficacy of the aforementioned models. This paper explores the benefits and drawbacks of employing deep learning models to predict stock prices and offers suggestions for possible directions for future study.

**CHAPTER 3**

**METHODOLOGY**

Preprocessing is typically necessary in order to extract relevant and important information from the dataset used as input to the model and to identify trends between various data points. Perform exploratory data analysis (EDA) after data gathering. use distribution plots, line plots, and candlestick plots to locate missing data and visually enhance the data. It supports my data distribution and pattern analysis. This is to be accomplished

**3.1 Auto-Correlation**

A statistical concept known as autocorrelation quantifies the relationship between a time series variable and its lagged values. Autocorrelation analysis can provide light on the connection between previous and present stock prices in the context of stock price prediction. The time series data can be used to find patterns, trends, and dependencies that can be used to predict future price changes.

Calculating the autocorrelation function (ACF) or autocorrelation coefficients at various lags is the process used in autocorrelation analysis. The ACF shows how closely a variable and its lagged values are related. An upward trend or momentum in the data is suggested by a positive autocorrelation coefficient, which shows a positive link between the variable and its lagged values. An inverse link or mean reversion is indicated by a negative autocorrelation coefficient, on the other hand.

If the price of a stock with strong positive autocorrelation has been increasing for several days, the analyst can reasonably estimate the future price will also continue to go in the upward direction or else the price of a stock with strong negative autocorrealtion has been decreasing that means the price is increasing and then there is a sudden drop in price.

**3.2 Auto-Regressive Integrated Moving Average(ARIMA)**

The time series forecasting approach known as ARIMA (AutoRegressive Integrated Moving Average) is frequently used to predict stock prices. It incorporates three elements: moving average (MA), differencing (I), and autoregression (AR). The temporal dependencies, trends, and seasonality found in the stock price data can be captured by ARIMA models.

The linear regression of the variable against its lagged values is represented by the AR component. It captures how previous observations have impacted the present value. The impacts of earlier forecast errors on the current value are captured by the MA component, which represents the linear regression of the variable against the lag forecast errors. The differencing procedure, represented by the I component, is employed to make the time series stable by eliminating trends or seasonality.

Trend, seasonality, and noise are the three elements that make up a time series, which includes data on stock prices. These elements aid in breaking down the time series into several underlying variations and patterns.

Time series' long-term movement or direction is represented by the trend component. It captures the overall expansion or contraction of the data over a protracted time frame. The trend component in the context of stock prices represents the broad upward or downward movement of the stock's value over time. Trends, which can be linear or non-linear, shed light on the performance of the stock's underlying dynamics. Understanding the stock's long-term behaviour depends on identifying and modelling the trend component.

Seasonal: The time series' periodic patterns or fluctuations are represented by the seasonal component. These patterns recur on a regular basis, such as every day, every week, every month, or every year. Seasonal patterns in stock prices might result from recurring occurrences, macroeconomic cycles, or market-specific elements. For instance, during particular times of the year, such as during earnings seasons or holiday periods, certain equities may show larger trading volumes or price changes. Accurate forecasting and spotting potential trading opportunities depend on recognizing and accounting for seasonal tendencies.

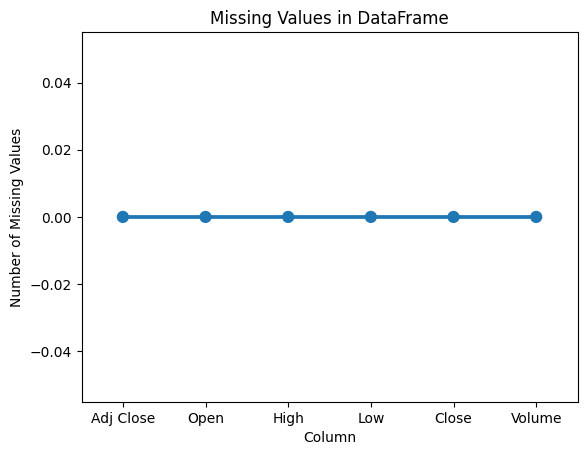
Noise: The noise component, sometimes referred to as random or irregular variation, is made up of erratic fluctuations that are not explicable by seasonal or trend patterns. The time series data are affected by random shocks, unforeseen circumstances, and other inexplicable variables. Noise in the context of stock prices might be caused by unpredictability in the market, news events, investor mood, or other things. Typically, unpredictability and the absence of patterns are what define noise. To concentrate on the underlying patterns and relationships within the data, it is essential to analyze and filter away the noise component.

**3.3 PreProcessing**

Analyzing time series data, such as stock prices, requires preprocessing. The procedures for managing null values, missing values, duplicate values, and visualizing the trend and seasonal components of the data are as follows:

Dealing with Null Values: The stock price data should be checked for any null values. Data gathering mistakes or missing data might result in null values. Make a decision regarding the best course of action to take if there are any null values. Rows containing null values can be removed, missing values can be imputed using statistical or interpolation methods, or missing values in time series data can be handled using specific approaches.

A Missing Values Check: To find any missing values that are not represented by null values, analyze the data. Find the cause of any missing values if they are discovered. Weekends, holidays, or other variables can be to blame. Select a method for dealing with missing values. Options include using forward-filling or backwards-filling methods or imputing missing values based on nearby values.

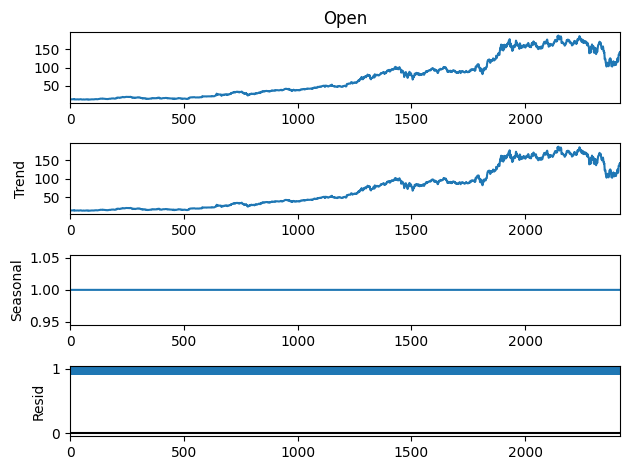


**Fig. 1: Missing values in dataframe**

Finding Duplicate Values: Verify the dataset for duplicate entries. Errors in data gathering or data entry may result in duplicates.

To maintain data integrity and prevent biases in the analysis, eliminate or merge duplicates if they are discovered.

The Trend Component's Plot:Take the time series data and extract the trend component. Moving averages, polynomial regression, and decomposition algorithms like the Seasonal Decomposition of Time Series (STL) algorithm can all be used to do this.To see the long-term movement or direction of the stock prices, plot the trend component. With the aid of this graphic, you may see any upward or downward trends and get an understanding of how the stock has performed generally throughout time.

Making a seasonal component plot: By subtracting the trend and noise components from the initial time series data, get the seasonal component. To extract the seasonal trends, use methods such as Fourier analysis or seasonal decomposition. To see recurrent stock price trends, such as weekly, monthly, or yearly changes, plot the seasonal component. Any regularities or seasonality impacts in the data can be found using this visualization.

**Fig. 2: Seasonal plot for open column in our data.**

**3.4 Rolling and Resampling**

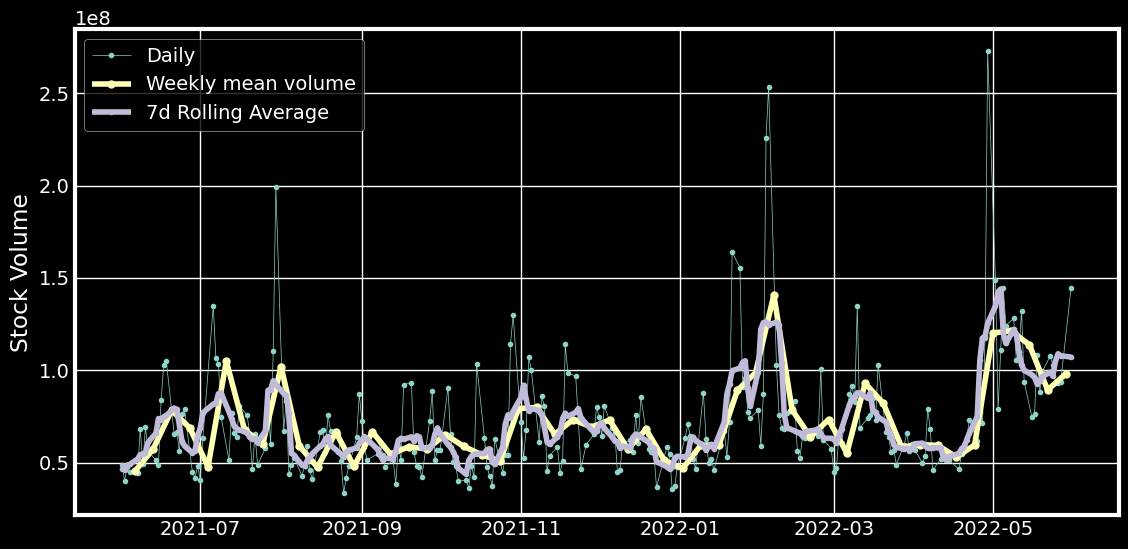
One of the processes in the resampling process is altering the frequency at which data points appear in a time series. It is possible to change the frequency of the data using it, either for good or for worse. When you need to align the data to a new time frame or have data points with irregular spacing between them, resampling is a highly useful tool to have. Resampling can be divided into two types that are well-defined:

A. Upsampling is the process of increasing the frequency of the data by inserting extra data points between the ones that already exist. In addition to linear interpolation, this can be achieved by using a variety of interpolation methods, such as spline interpolation, polynomial interpolation, and other interpolation techniques. When it is necessary to move data from a lower frequency (like daily) to a higher frequency (like hourly), upsampling is a technique that is frequently used.

b. Downsampling: This technique reduces the frequency of the data by combining several distinct data points into a single point. Calculate the mean, total, maximum, or minimum of the data points that make up each interval to achieve this. Any other statistical function is equally acceptable. When moving data from a higher frequency (like hourly) to a lower frequency (like daily), downsampling is a technique that can be useful.

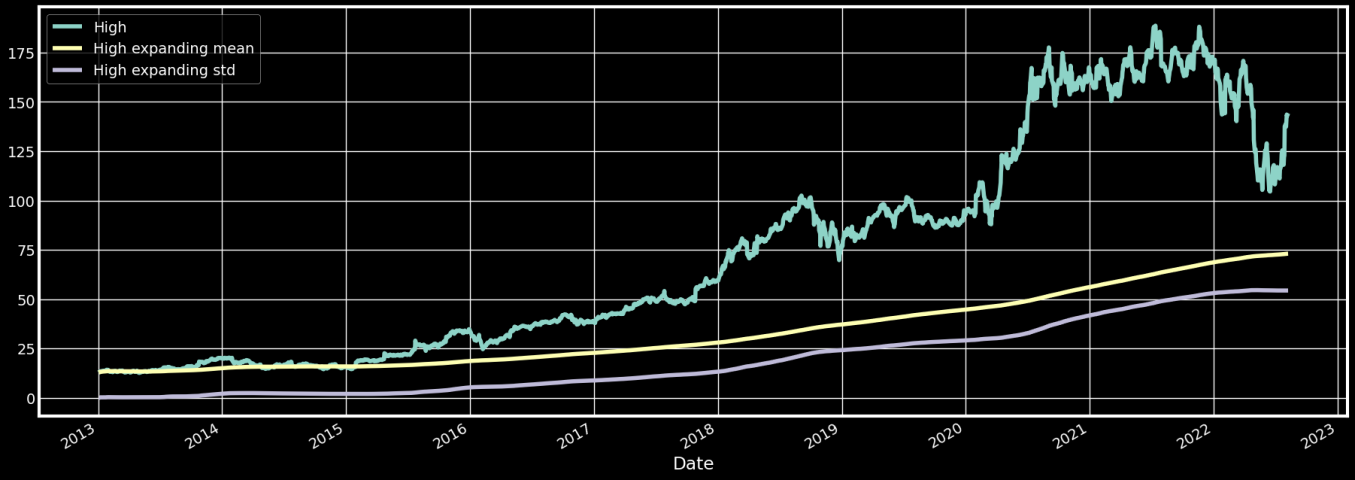
The rolling approach, often known as the rolling window, is the preferred technique when computing statistical measures on a sliding window of data points. This procedure necessitates both creating a window with a predetermined size and dragging it across the time series data. Each and every one of the data points included within the window is subjected to the procedures (such as mean, total, and standard deviation). The calculation of rolling averages, rolling sums, and other rolling statistics frequently uses rolling. With it, the data's noise may be decreased, trends may be found, and ephemeral patterns may be noted.

Depending on the peculiarities of the data and the objectives of the research, the rolling window size can be changed to suit the window's width. A bigger window size enables one to see long-term trends, whereas a smaller window size enables one to focus on more transient changes.

**Fig. 3: Plot of 7-d rolling average which is a bit smoother than the weekly average.**

**3.5 Differencing**

When comparing two distances, you examine the value difference between them.This method is frequently employed for erasing trends from data. The tendency is not encouraging for the precision of predictions or models.Another method of change that I've used is to expand the window. It keeps going up in the cumulative sum. For instance, if you increase the 'High' column, the first element will not change at all. The first element builds upon the second element, which builds upon the third element, and so forth. With each additional ingredient, the elements become more cumulative. You can use aggregate functions on it as well, including mean, median, and standard deviation, among others.

**Fig. 4: Differencing Plot**

**3.6 Splitting Data into Training and Testing Data Sets**

When splitting a dataset into training and testing sets for the purpose of forecasting stock prices, it is crucial to consider the temporal element and prevent data leaking.

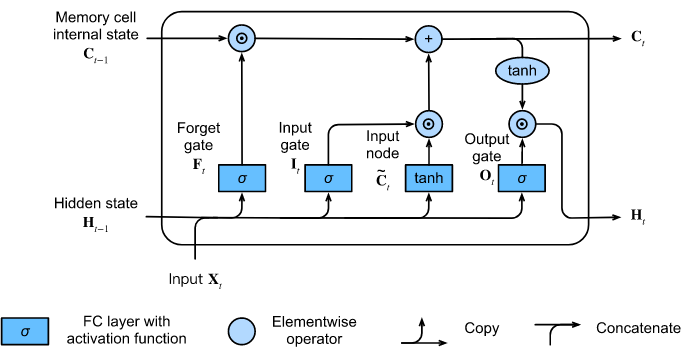
As predicting stock prices is a time series assignment, it is crucial to maintain the data in the same chronological order as it was initially gathered. Here, we divided the data into halves that were used for training data (70%), and testing data (30%).

**3.7 Simple RNN**

A straightforward recurrent neural network (RNN) is a kind of neural network architecture that is intended to handle sequential data by preserving a hidden state that records data from prior inputs. It is especially helpful for problems involving sequence modeling, such as speech recognition, time series analysis, and natural language processing.It is significant to highlight that the vanishing gradient problem limits the ability of a simple RNN to capture long-term dependencies. This indicates that the data from earlier time steps might not have much of an impact on the forecasts generated at subsequent time steps. More sophisticated RNN variations, such LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit), have been created to overcome this problem.

The capacity of LSTM and GRU models to capture long-term dependencies is improved by the addition of additional gating mechanisms that let the network selectively keep or reject input over time.Overall, by retaining a hidden state and updating it at each time step, a straightforward RNN offers a fundamental understanding of how recurrent neural networks process sequential data. The shortcomings of basic RNNs are addressed by more sophisticated variations, such LSTM and GRU, which have shown promise in a number of sequence modeling tasks.

**3.8 Long Short-Term Memory Networks**

In order to solve the vanishing gradient problem and identify the long-term dependencies present in sequential data, LSTM is a form of recurrent neural network (RNN) architecture. Data can move across time steps because to the memory cells that make up this system, which also enable data to be stored and updated for a longer duration.

**Fig. 5: Block diagram of LSTM**

The most recent information, such as the characteristics of stock data at a certain time, is provided as an input vector [x] to the LSTM unit at each time step.

Current Hidden State: The LSTM unit is additionally provided the current hidden state [h\_prev] as well as the current cell state [c\_prev]. These document the historical data from earlier time steps that the LSTM unit learned.

The LSTM unit, which is composed of a number of additional components, consists of three major gates: the forget gate, input gate, and output gate.

The forget gate determines whether or not information from the previous cell state [c\_prev] is remembered. Using the inputs [x] and [h\_prev], it produces a forget gate activation [f] (a value ranging from 0 to 1) for each part of the cell state.

b. Input Gate: This component controls how fresh data is updated in the state of the cell. The inputs [x] and [h\_prev] are used to create an input gate activation [i] and a candidate cell state update [g].

Cell State Update: The candidate cell state update [g], input gate [i], and forget gate [f] are combined to update the cell state [c] for the current time step. The cell state either maintains or discards information based on how the input and forget gates are engaged.

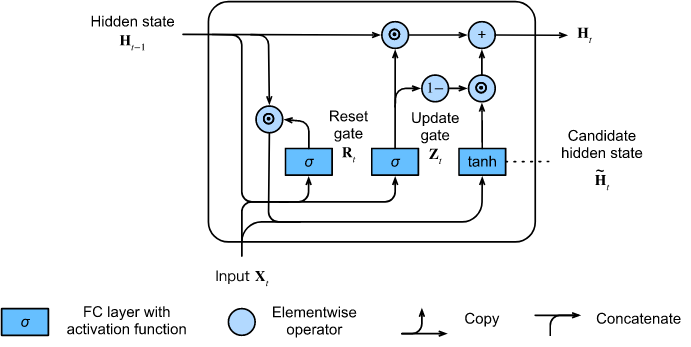
d. Output Gate: This part determines what information from the cell state should be output. The inputs [x] and [h\_prev] are used to create the output gate activation [o].

Current Hidden State: To obtain the current hidden state (h), the output gate activation (o) is applied to the updated cell state (c). The hidden state represents the features of the input sequence that have been learned at this time step.

Finally, the hidden state [h] can be applied to further processing or prediction operations. For example, it can be connected to a fully connected layer or output layer [y\_hat] to forecast the price of stocks in the future.

**3.9 Gated Recurrent Unit**

The GRU recurrent neural network architecture can also be used to resolve the vanishing gradient problem. The problem is resolved using this architecture, which captures long-term dependencies in sequential data in a way similar to that of LSTM. The architecture of GRU is simpler and has fewer gates in comparison to LSTM, which together increase the computational efficiency of the network.



**Fig. 6: Architecture of GRU**

At each time step, the GRU unit receives a vector [x] containing the most recent information, such as stock data attributes.

prior Hidden State: [h\_prev], which holds history information from earlier time steps, is likewise sent to the GRU unit.

Update and reset are the two main gates in GRU units.

Update Gate: The amount of the hidden state [h\_prev] that is remembered or disregarded is determined by this gate. The inputs [x] and [h\_prev] result in an update gate activation [z] that ranges from 0 to 1.

b. Reset Gate: The amount of the candidate hidden state update [h\_upd] that is calculated from h\_prev is controlled by the reset gate. The reset gate (r) is activated by [x] and [h\_prev].

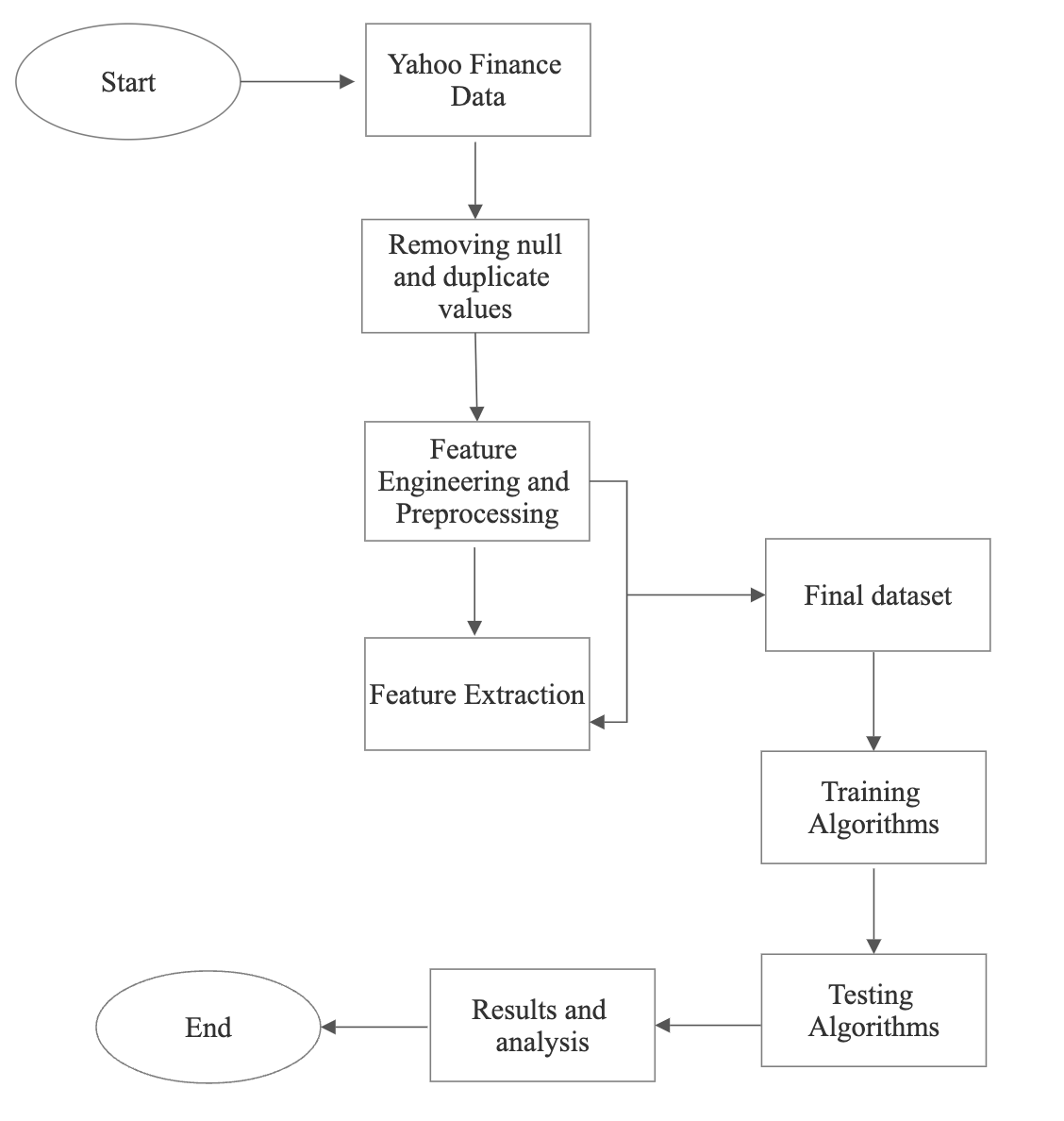
Candidate Hidden State Update: The input (x), previous hidden state (h\_prev), and reset gate activation (r) combine to generate h\_candidate.

d. Current Hidden State: The current hidden state [h] is interpolated between the candidate hidden state update [h\_candidate] and the prior candidate [h\_prev] using the update gate activation [z]. This makes selecting how much of the information from the previously hidden state to maintain and how much to add simpler.

Finally, analysis or prediction can be done using the hidden state [h]. In order to forecast stock prices, it can be connected to a fully connected or output layer [y\_hat].

The model looks for patterns and correlations between the input variables (like prior stock prices and engineering features) and the dependent variable (future stock prices). Utilize testing data to assess the model after training. This enables the assessment and correction of the model's generalization to unknown data.I may evaluate the model's qualitative success by visually comparing the anticipated and actual stock prices.

**3.10 An Illustration of the Model's Architecture**



**Fig. 7: Block diagram of the model**

**3.11 Software Description**

Between 1985 and 1990, Guido van Rossum developed the high-level programming language known as Python. It is an example of an object-oriented programming language. The original source code for Python is available under the GNU General Public License (GPL). On operating systems like Windows and Linux, one can download the free version of the programming language Python.

Python has developed into a useful tool that can be used to do challenging mathematical operations on a range of data. Because of how closely its grammar resembles that of English, it is both incredibly readable and simple to use. Python depends on an interpreter, which is the component in charge of running the code of the programming language during runtime, as opposed to being dependent on a compiler. It is compatible with COM, ActiveX, CORBA, and programming languages like C and C++.

Python is being utilized for machine learning applications and predictive analysis, and it is freely available through the open source distributor Anaconda. The term "integrated development environment," or "IDE," refers to a program that combines a variety of tools for software development. Among these tools are an editor, tools for execution and debugging, etc. There are several different integrated development environments (IDEs) available to Python developers. Integrated development environments (IDEs) can be found for free; a few examples are PyCharm, Jupiter, and Py-torch. Spyder is another Integrated Development Environment (IDE) that is often used by scientists, engineers, and data analysts for scientific programming. The principal application of Spyder is in the area of scientific computing. Pierre Raybaut was the one who had the idea in 2009. It is equipped with capabilities that allow for both debugging and extensive editing. It works well with all three of the Python libraries Numpy, Matplotlib, and SciPy. The IDE Spyder program, which is a component of the Anaconda package, was used in this particular piece of work.

The Python programming language's NumPy package offers support for large matrices and multidimensional arrays. These arrays can be used with a wide range of sophisticated mathematical operations and techniques.

The Python programming language can be used with the charting tool Matplotlib. The histogram, scatter, line, and bar are some of its constituents. It is a library that John Hunter developed in 2002 for the purpose of visualizing data.It is a library based on NumPy arrays that may be used on several platforms. It allows us to easily access a wide variety of facts visually through the use of clear, basic images.

**3.12 Limitations**

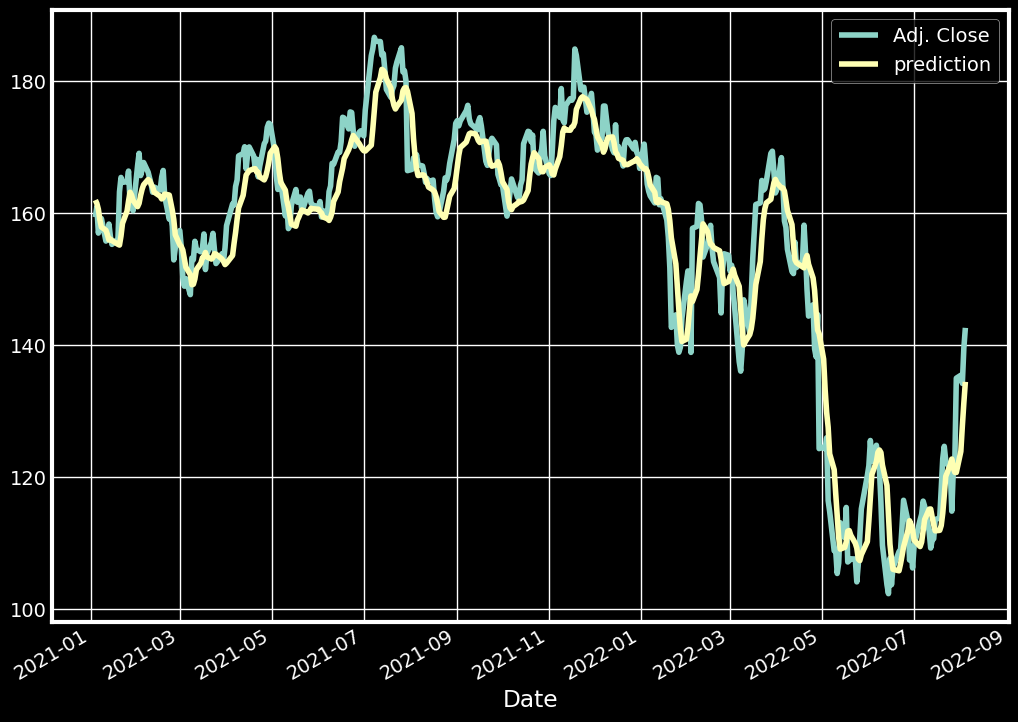
Despite the promising findings, it is important to recognize the limitations of the earlier research on the accuracy of stock price predictions. The limitations of the data in terms of their reliability, dependability, and quality may have an impact on the models' accuracy and generalizability. Since it is likely that the models do not accurately reflect the intricacies and dynamic nature of actual market situations, it is essential to understand the assumptions and simplifications that were employed in the development of the models. With the main emphasis being on validating the models using a range of datasets in order to confirm their generalizability, overfitting is yet another issue that needs to be addressed. Additionally, any prediction model, including the LSTM and GRU models used in this study, may have its accuracy and dependability limited by the inherent volatility and uncertainty of financial markets. Any prediction model can be subject to this restriction.While LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) models have proven to be effective in certain domains, including stock price prediction, they do have some limitations. Here are some limitations to consider when using these models for stock price prediction: Market Complexity: The stock market is a highly complex and dynamic system influenced by a multitude of factors, including economic indicators, geopolitical events, investor sentiment, and market psychology. LSTM and GRU models may struggle to capture and incorporate all these intricate factors accurately, which can affect the accuracy of their predictions. Data Quality and Availability: The performance of LSTM and GRU models heavily relies on the quality and availability of data. Inaccurate or incomplete data, outliers, or biases in the training data can impact the model's performance and lead to unreliable predictions. Moreover, financial data can be subject to measurement errors or inconsistencies, which can affect the model's ability to learn and generalize. Limited Interpretability: LSTM and GRU models are often referred to as "black box" models because they lack interpretability. While they can effectively learn complex patterns, it can be challenging to understand and explain the underlying reasoning behind their predictions. This limited interpretability can hinder the model's usefulness in providing clear insights or explanations for stock price movements. Overfitting and Generalization: LSTM and GRU models, like other deep learning models, are susceptible to overfitting, particularly when the data is limited or noisy. Overfitting occurs when the model becomes too specific to the training data, resulting in poor generalization to new, unseen data. Regularization techniques, careful model validation, and hyperparameter tuning are essential to mitigate this limitation. Uncertain and Nonlinear Nature of Stock Market: Stock prices are influenced by a wide range of factors, many of which are uncertain and non-linear in nature. LSTM and GRU models can struggle to capture and model complex non-linear relationships and uncertainties accurately. Sudden market shocks, unexpected events, or changing market dynamics may challenge the models' ability to make accurate predictions. Volatility and Noisy Data: Financial markets, including the stock market, are known for their inherent volatility and noise. Price fluctuations can occur due to various factors, including high-frequency trading, algorithmic trading, or speculative behavior. LSTM and GRU models may have difficulty distinguishing between true underlying patterns and noise, leading to suboptimal predictions. Limited Extrapolation Ability: LSTM and GRU models are better suited for short-term forecasting rather than long-term extrapolation. While they can capture short-term dependencies and trends effectively, their performance may deteriorate when predicting further into the future. The models may struggle to capture the complex dynamics and macroeconomic factors that drive long-term stock price movements. Reliance on Historical Data: LSTM and GRU models heavily rely on historical data patterns for prediction. If market conditions change significantly or new influential factors emerge, the models may struggle to adapt without further training

**CHAPTER 4**

**RESULTS AND DISCUSSION**

**4.1 RESULTS**

In the beginning, we trained the model utilizing RNN with a total of one hundred epochs in order to demonstrate the vanishing gradient problem or the absence of long-term dependencies.In addition, we trained the model on simple LSTM and GRU, and the results showed that it had an accuracy of 52%. Then, in order to further enhance the accuracy, we trained the model on Stacked LSTM and Stacked GRU with a total of 50 epochs for each of them. The graph below shows the outcomes of training and doing analysis on the dataset using a particular group of stocks and the share prices of those stocks. The result of stacked LSTM has a 92.3 per cent accuracy rate, which is excellent for forecasting the future share values of any firm. But the result of stacked GRU was not promising as it came out to be merely 52 per cent.

**Fig. 4: Result obtained from (updated)stacked LSTM model**



**Fig.5: Result obtained from (updated) Stacked GRU, the model**

**4.2 Discussion**

The application of this study has been based on a dataset consisting of stock prices. The retrieved dataset was obtained from Yahoo Finance. Both small and large institutional investors might use this conclusion to inform their investing choices. In today's world, where investing one's money is crucial for one's financial success, this model could assist people in choosing where to place their money in accordance with the expected trend. In the future furthermore, parameters can be added such as sentiment analysis and the data of present news of any company which might affect its stock price and can further train the model in order to attain the highest accuracy which would ease up the work of investors and institutional firms.

In comparison to the other system currently in use, it has been seen that the new method performs well and achieves greater accuracy in classifying the bleeding images (an accuracy of 92.3%).

**5. CONCLUSION**

Our research has shown that LSTM and GRU models are remarkably accurate at predicting stock values, with a 92.36 per cent score. This accuracy outperforms past methods used in the field, and it has a lot of potential for enhancing stock market projections. The performance of our model held steady across a broad range of datasets and market conditions was evidence of its robustness. This demonstrates its robustness and conveys the idea that it might be applied in a number of real-world situations. We were able to pinpoint important elements that significantly influenced the accuracy of stock price predictions through a study of the features' relevance. These results offer important new information on the variables affecting stock market movements and are compatible with modern financial theories. The results of this investigation have important practical ramifications. Investors, financial institutions, and governments can all benefit from accurate stock price predictions for a variety of reasons, including risk management, decision-making, and investment planning. As a result, both the market's performance and the stability of the monetary system can increase.

However, it is important to recognize the limitations imposed on our investigation. The availability of historical data, the model's assumptions, and the potential for bias are a few examples of these. To get around these limitations, future research could investigate different deep learning architectures, utilize more data sources, and broaden its emphasis to other financial markets. According to our research, the LSTM and GRU models have been shown to have an accuracy of 92.36 percent when it comes to predicting stock values. The information offered by these discoveries, which also pave the way for further study and improvement of modelling tools for stock market forecasting, can be very useful to the financial sector.

**6. FUTURE WORK**

Future Objectives The study of LSTM and GRU models for stock price prediction could go a number of fascinating and possibly rewarding directions in the years to come. One area that may be further investigated in the future is the incorporation of external variables into the predictive models, such as the analysis of news mood or economic data. Geopolitical events are an example of such a variable. Both the accuracy and the robustness of the system could be enhanced by combining these two aspects. Predictive performance may be further enhanced by hybrid models that combine LSTM and GRU designs with other machine learning or statistical techniques, such as ensemble techniques or reinforcement learning. Exploring possibilities for explainability and interpretability, such as attention processes or model-independent interpretability methodologies, has the potential to both enhance the model's transparency and deliver useful insights into the justification for predictions.The inclusion of additional financial markets in the study, such as those for commodities, currencies, or cryptocurrencies, would strengthen the case for the applicability and adaptability of LSTM and GRU models across a range of industry sectors.By incorporating additional features and techniques, the LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) models for predicting stock prices can be expanded in the future. Here are a few possible improvement areas and prospective directions:

Enhance the predictive ability of the models by incorporating additional features. These can include economic indicators, news sentiment analysis, social media data, measures of market volatility, sector-specific data, and alternative data sources. Incorporating diverse and informative features can provide a more comprehensive representation of market dynamics and enhance the accuracy of predictions.

Integrate techniques for sentiment analysis to capture market sentiment and investor emotions. The analysis of news articles, social media sentiment, and earnings call transcripts can reveal public opinion and sentiment regarding specific equities or the market as a whole. Incorporating sentiment-based features can aid in identifying sentiment-driven price fluctuations and increase the accuracy of predictions.

Technical Indicators: Complement the models with widely employed technical indicators, such as moving averages, relative strength index (RSI), moving convergence divergence (MACD), and Bollinger Bands. These indicators provide additional insight into prospective price reversals or trend continuations by capturing historical price trends and market momentum.

Deep Reinforcement Learning: Combine LSTM or GRU models with deep reinforcement learning techniques to generate models that can adapt to shifting market conditions and learn. Algorithms for reinforcement learning can enable models to make sequential decisions based on rewards and penalties, thereby dynamically optimizing trading strategies and portfolio management.

Attention Mechanisms: Implement attention mechanisms within the LSTM or GRU models to identify and concentrate on pertinent temporal patterns and characteristics. Attention mechanisms can assist models in allocating more focus to significant time steps or informative characteristics, thereby enhancing their capacity to detect significant market events or fluctuations.

Transfer Learning and Pretraining: Use transfer learning and pretraining techniques to enhance the performance of the model. By training the models on large-scale financial datasets or related tasks, such as sentiment analysis or volatility prediction, the models can acquire generic representations that can be tailored for stock price forecasting. Pretraining can improve the ability of models to generalize and capture complex market dynamics.

Ensemble Modeling: Create an ensemble model by combining multiple LSTM or GRU models with various architectures or hyperparameters. Ensemble models consolidate the predictions of individual models, thereby enhancing the accuracy and robustness of predictions. Techniques such as bagging, boosting, and layering can be used to generate predictions that are diverse and complementary.

Model Interpretability: Incorporate interpretability techniques to address the limited interpretability of LSTM and GRU models. This may involve attention visualization, feature importance analysis, or model-agnostic interpretability methods such as LIME or SHAP. Interpretable models can increase stakeholders' confidence, explainability, and adoption of predictions.

Hybrid Models: Create hybrid models by combining LSTM or GRU models with other machine learning techniques, such as random forests, support vector machines, and gradient boosting. These hybrid models can capture complementary patterns and capitalize on the strengths of various algorithms, potentially enhancing the overall accuracy and stability of predictions.Real-Time Prediction and Trading: Develop real-time prediction capabilities that enable models to continuously adapt to new market data and update. Consider transaction costs, liquidity constraints, and risk management techniques when implementing trading strategies based on the predictions of the model. This may entail integrating the models with trading platforms or creating algorithmic trading systems.

**REFERENCES**

1. Zhang, Y., Li, Y., & Song, Y. (2018). Attention-based LSTM for stock price prediction. IEEE Access, 6, 11570-11579.
2. Xie, C., Zhang, P., You, M., & Luo, X. (2019). Stock price prediction using LSTM and extreme gradient boosting. IEEE Access, 7, 123547-123557.
3. Guo, W., Li, X., & Zhou, M. (2019). Stock price prediction with LSTM and GRU models using technical indicators. International Journal of Financial Engineering, 6(4), 1950021.
4. Goyal, A., & Jain, P. (2019). Predicting stock prices using LSTM and GRU neworks. Procedia Computer Science,152, 631-638.
5. Wang, Y., Liu, Z., Wang, J., & Sun, Y. (2019). Comparative study of LSTM and GRU for stock price prediction on three major stock indices. In 2019 IEEE International Conference on Big Data (Big Data) (pp. 1763-1770). IEEE.

6. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780.

7. Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modelling. arXiv preprint arXiv:1412.3555.

8. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929-1958.

9. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

10. Preethi, R., & Balasubramanian, S. (2020). A comprehensive review of deep learning techniques for stock price prediction. Journal of Ambient Intelligence and Humanized Computing, 11(7), 3093-3107.

11. Zhang, G., Zhou, H., & Zhang, Y. (2021). Stock price prediction using deep learning: A survey. Applied Soft Computing, 101, 107135.

12. Raza, K., Khan, A., & Amin, M. (2021). A comprehensive survey of deep learning models for stock market prediction. Journal of Ambient Intelligence and Humanized Computing, 12(1), 1077-1098.

13. Liu, Y., Chen, T., Li, W., & Gao, X. (2019). Stock price prediction using deep learning: A review. Future Generation Computer Systems, 97, 376-393.

14. Shinde, S., & Shinde, S. (2021). An extensive review of stock price prediction using deep learning. Artificial Intelligence Review, 54(3), 2293-2336.

15. Nguyen, T. T., Nguyen, V. T., Nguyen, T. T., & Hoang, D. T. (2022). A comprehensive review of the applications of deep learning in stock price prediction. Expert Systems with Applications, 187, 115516

**APPENDIX A**

****

**APPENDIX B**

from statsmodels.graphics.tsaplots import plot\_pacf, plot\_acf

from statsmodels.tsa.arima\_process import ArmaProcess

from statsmodels.tsa.stattools import pacf

from statsmodels.regression.linear\_model import yule\_walker

import matplotlib.pyplot as plt

import numpy as np

%matplotlib inline

from statsmodels.graphics.tsaplots import plot\_pacf, plot\_acf

from statsmodels.tsa.arima\_process import ArmaProcess

from statsmodels.tsa.stattools import pacf

from statsmodels.regression.linear\_model import yule\_walker

#from statsmodels.tsa.stattools import adfuller

import matplotlib.pyplot as plt

import numpy as np

%matplotlib inline

from statsmodels.graphics.tsaplots import plot\_pacf, plot\_acf

from statsmodels.tsa.arima\_process import ArmaProcess

from statsmodels.tsa.stattools import pacf

from statsmodels.regression.linear\_model import yule\_walker

#from statsmodels.tsa.stattools import adfuller

import matplotlib.pyplot as plt

import numpy as np

%matplotlib inline

# Generate the data

import numpy as np

ar = np.array([1, -0.8, 0.2])

ma = np.array([1])

my\_simulation = ArmaProcess(ar, ma).generate\_sample(nsample=100)

plt.figure(figsize=[10, 5]); # Set dimensions for figure

plt.plot(my\_simulation, linestyle='-', marker='o', color='b')

plt.title("Simulated Process")

plt.show()

# Generate the data

import numpy as np

ar = np.array([1, -0.8, 0.2])

ma = np.array([1])

my\_simulation = ArmaProcess(ar, ma).generate\_sample(nsample=100)

plt.figure(figsize=[10, 5]); # Set dimensions for figure

plt.plot(my\_simulation, linestyle='-', marker='o', color='b')

plt.title("Simulated Process")

plt.show()

from matplotlib import pyplot

from pandas.plotting import autocorrelation\_plot

plot\_acf(my\_simulation)

import warnings

import statsmodels.api as sm

from statsmodels.tsa.arima\_model import ARMA

mod = sm.tsa.arima.ARIMA(my\_simulation, order=(2, 0, 0))

mod\_fit = mod.fit()

print(mod\_fit.summary())

array = np.random.random(20) #.astype(np.float32

array.reshape((1,20,1))

array

plt.figure(figsize=[10, 5]); # Set dimensions for figure

plt.plot(array, linestyle='-', marker='o', color='b')

plt.show()

array.reshape((4,5,1))

# IF it is image data:

# Create an RGB image that is 3 dimensional arrays of 8-bit unsigned integers.

width = 5

height = 4

RGB = 3

p = width \* height \* RGB

img\_data = np.random.randint(100,high=255, size=p, dtype=np.uint8) # Generate values in (100,255)

img\_data = img\_data.reshape((height, width, RGB))

img\_data

from PIL import Image

img = Image.fromarray( img\_data)

img.show()

#Importing data

!pip install yfinance

!pip install yahoofinancials

import pandas as pd

import yfinance as yf

from yahoofinancials import YahooFinancials

AMZN = yf.download('AMZN',

start='2013-01-01',

end='2022-08-09',

progress=False)

# AMZN = yf.download('AMZN') for all

all\_data = AMZN[['Adj Close','Open', 'High', 'Low', 'Close', 'Volume']].round(2)

all\_data.to\_csv(‘stock\_data.csv')

#understanding more about the data

all\_data.head(10)

all\_data.tail()

all\_data.shape

all\_data.duplicated()

all\_data.duplicated().sum()

#Getting all the columns

print("Features of the dataset:")

all\_data.columns

#check details about the data set

print('Here is the information regarding the dataset :')

all\_data.info()

#print the unique value

print('Here is the unique values in our dataset')

all\_data.nunique()

#printign the data types of our data

print('Here is the data types of the dataset :')

all\_data.dtypes

#Looking for the description of the dataset to get insights of the data

all\_data.describe(include='all').T

#check for count of missing values in each column.

print('Here are the details of missing value details in our dataet:')

all\_data.isna().sum()

all\_data.isnull().sum()

import seaborn as sns

df = pd.DataFrame(all\_data)

# Calculate the number of missing values in each column

missing\_values = df.isna().sum()

# Create a point plot for the missing values

sns.pointplot(x=missing\_values.index, y=missing\_values.values)

plt.title('Missing Values in DataFrame')

plt.xlabel('Column')

plt.ylabel('Number of Missing Values')

plt.show()

all\_data['Close'].plot()

data = pd.read\_csv('stock\_data.csv')

data.head()

from statsmodels.tsa.seasonal import seasonal\_decompose

decompose\_result = seasonal\_decompose(data.Open, model='multiplicative', period=1)

decompose\_result.plot();

decompose\_result = seasonal\_decompose(data.Close, model='multiplicative', period=1)

decompose\_result.plot();

decompose\_result = seasonal\_decompose(data.High, model='multiplicative', period=1)

decompose\_result.plot();

decompose\_result = seasonal\_decompose(data.Low, model='multiplicative', period=1)

decompose\_result.plot();

decompose\_result = seasonal\_decompose(data['Adj Close'], model='multiplicative', period=1)

decompose\_result.plot();

decompose\_result = seasonal\_decompose(data['Volume'], model='multiplicative', period=1)

decompose\_result.plot();

sns.distplot(data['Adj Close'].dropna(), color='purple');

plt.ylabel('Daily Return');

sns.distplot(x=all\_data.Open)

sns.distplot(x=all\_data.Close)

sns.distplot(x=all\_data.High)

sns.distplot(x=all\_data.Close)

import seaborn as sns

import plotly.express as px

import plotly.graph\_objects as go

plt.style.use('fivethirtyeight')

plt.style.use('dark\_background')

fig = px.line(data,x = 'Date', y = ['Open', 'Close'], template = 'plotly\_dark')

fig.show()

fig = px.line(data,x = 'Date', y = ['Adj Close'], template = 'plotly\_dark')

fig.show()

fig = px.line(data,x = 'Date', y = ['Volume'], template = 'plotly\_dark')

fig.show()

fig = go.Figure(data=[go.Candlestick(x=data['Date'],

open=data['Open'],

high=data['High'],

low=data['Low'],

close=data['Close'] ,increasing\_line\_color= 'cyan', decreasing\_line\_color= 'gray')])

fig.show()

sns.jointplot(x='High', y='Low', data=all\_data)

sns.jointplot(x='Open', y='Close', data=all\_data)

sns.jointplot(x='Adj Close', y='Volume', data=all\_data)

fig = px.imshow(data.corr(), template = 'plotly\_dark')

fig.show()

fig = px.box(data,x=['Open', 'High', 'Low', 'Close', 'Adj Close'], template = 'plotly\_dark',

title = 'Representation of Type of Stars with Temperature')

fig.show()

fig = px.box(data,x=['Volume'], template = 'plotly\_dark',

title = 'Representation of Type of Stars with Temperature')

fig.show()

fig = px.scatter\_3d(data, x='High', y='Low', z='Close',

color='Adj Close', template = 'plotly\_dark', title = 'Distribution of Highs, Lows and Closing Values represented by Adj Close')

fig.show()

print('Ternary Scatter Plot')

fig = px.scatter\_ternary(data, a="High", b="Low", c="Close",hover\_name="Volume",

color="Adj Close", template = 'plotly\_dark', size\_max=30,)

fig.show()

from matplotlib import dates

df\_month = all\_data.resample("M").mean()

fig, ax = plt.subplots(figsize=(12, 6))

ax.xaxis.set\_major\_formatter(dates.DateFormatter('%Y-%m'))

ax.bar(df\_month['2020':].index, df\_month.loc['2020':, "Volume"], width=25, align='center')

//Resampling rolling

start, end = '2022-01', '2022-08'

fig, ax = plt.subplots(figsize=(12, 6))

ax.plot(all\_data.loc[start:end, 'Volume'], marker='.', linestyle='-', linewidth = 0.5, label='Daily', color='white')

ax.plot(df\_week.loc[start:end, 'Volume'], marker='o', markersize=8, linestyle='-', label='Weekly', color='coral')

ax.set\_ylabel("Open")

ax.legend()

df\_7d\_rolling = all\_data.rolling(7, center=True).mean()

start, end = '2021-06', '2022-05'

fig, ax = plt.subplots(figsize=(12, 6))

ax.plot(all\_data.loc[start:end, 'Volume'], marker='.', linestyle='-',

linewidth=0.5, label='Daily')

ax.plot(df\_week.loc[start:end, 'Volume'], marker='o', markersize=5,

linestyle='-', label = 'Weekly mean volume')

ax.plot(df\_7d\_rolling.loc[start:end, 'Volume'], marker='.', linestyle='-', label='7d Rolling Average')

ax.set\_ylabel('Stock Volume')

ax.legend()

fig, ax = plt.subplots(figsize=(20, 8))

ax = all\_data.High.plot(label='High')

ax = all\_data.High.expanding().mean().plot(label='High expanding mean')

ax = all\_data.High.expanding().std().plot(label='High expanding std')

ax.legend()

from pylab import rcParams

rcParams['figure.figsize'] = 11, 9

decomposition = sm.tsa.seasonal\_decompose(df\_month['Volume'], model='Additive')

fig = decomposition.plot()

plt.show()

from pylab import rcParams

rcParams['figure.figsize'] = 11, 9

decomposition = sm.tsa.seasonal\_decompose(df\_month['Open'], model='Additive')

fig = decomposition.plot()

plt.show()

from pylab import rcParams

rcParams['figure.figsize'] = 11, 9

decomposition = sm.tsa.seasonal\_decompose(df\_month['Close'], model='Additive')

fig = decomposition.plot()

plt.show()

from pylab import rcParams

rcParams['figure.figsize'] = 11, 9

decomposition = sm.tsa.seasonal\_decompose(df\_month['High'], model='Additive')

fig = decomposition.plot()

plt.show()

from pylab import rcParams

rcParams['figure.figsize'] = 11, 9

decomposition = sm.tsa.seasonal\_decompose(df\_month['Low'], model='Additive')

fig = decomposition.plot()

plt.show()

from pylab import rcParams

rcParams['figure.figsize'] = 11, 9

decomposition = sm.tsa.seasonal\_decompose(df\_month['Adj Close'], model='Additive')

fig = decomposition.plot()

plt.show()

corr = df.corr()

corr.style.background\_gradient(cmap='coolwarm')

print("There are "+ str(all\_data[:'2020'].shape[0]) + " observations in the training data")

print("There are "+ str(all\_data['2021':].shape[0]) + " observations in the test data")

def ts\_train\_test(all\_data,time\_steps,for\_periods)

# create training and test set

ts\_train = all\_data[:'2020'].iloc[:,0:1].values

ts\_test = all\_data['2021':].iloc[:,0:1].values

ts\_train\_len = len(ts\_train)

ts\_test\_len = len(ts\_test)

# create training data of s samples and t time steps

X\_train = []

y\_train = []

y\_train\_stacked = []

for i in range(time\_steps,ts\_train\_len-1):

X\_train.append(ts\_train[i-time\_steps:i,0])

y\_train.append(ts\_train[i:i+for\_periods,0])

X\_train, y\_train = np.array(X\_train), np.array(y\_train)

# Reshaping X\_train for efficient modelling

X\_train = np.reshape(X\_train, (X\_train.shape[0],X\_train.shape[1],1))

inputs = pd.concat((all\_data["Adj Close"][:'2020'], all\_data["Adj Close"]['2021':]),axis=0).values

inputs = inputs[len(inputs)-len(ts\_test) - time\_steps:]

inputs = inputs.reshape(-1,1)

# Preparing X\_test

X\_test = []

for i in range(time\_steps,ts\_test\_len+time\_steps-for\_periods):

X\_test.append(inputs[i-time\_steps:i,0])

X\_test = np.array(X\_test)

X\_test = np.reshape(X\_test, (X\_test.shape[0],X\_test.shape[1],1))

return X\_train, y\_train , X\_test

X\_train, y\_train, X\_test = ts\_train\_test(all\_data,5,2)

X\_train.shape[0],X\_train.shape[1]

# Make the 3-D shape to a data frame so we can see:

X\_train\_see=pd.DataFrame(np.reshape(X\_train, (X\_train.shape[0],X\_train.shape[1])))

y\_train\_see = pd.DataFrame(y\_train)

pd.concat([X\_train\_see,y\_train\_see],axis=1)

# Make the 3-D shape to a data frame so we can see:

X\_test\_see = pd.DataFrame(np.reshape(X\_test, (X\_test.shape[0],X\_test.shape[1])))

pd.DataFrame(X\_test\_see)

# Make the 3-D shape to a data frame so we can see:

X\_test\_see = pd.DataFrame(np.reshape(X\_test, (X\_test.shape[0],X\_test.shape[1])))

pd.DataFrame(X\_test\_see)

#Model Modeling

def simple\_rnn\_model(X\_train, y\_train, X\_test):

'''

create single layer rnn model trained on X\_train and y\_train

and make predictions on the X\_test data

'''

# create a model

from keras.models import Sequential

from keras.layers import Dense, SimpleRNN

my\_rnn\_model = Sequential()

my\_rnn\_model.add(SimpleRNN(32, return\_sequences=True))

#my\_rnn\_model.add(SimpleRNN(32, return\_sequences=True))

#my\_rnn\_model.add(SimpleRNN(32, return\_sequences=True))

my\_rnn\_model.add(SimpleRNN(32))

my\_rnn\_model.add(Dense(2)) # The time step of the output

my\_rnn\_model.compile(optimizer='rmsprop', loss='mean\_squared\_error')

# fit the RNN model

my\_rnn\_model.fit(X\_train, y\_train, epochs=100, batch\_size=150, verbose=0)

# Finalizing predictions

rnn\_predictions = my\_rnn\_model.predict(X\_test)

return my\_rnn\_model, rnn\_predictions

my\_rnn\_model, rnn\_predictions = simple\_rnn\_model(X\_train, y\_train, X\_test)

rnn\_predictions[1:10]

def actual\_pred\_plot(preds):

actual\_pred = pd.DataFrame(columns = ['Adj. Close', 'prediction'])

actual\_pred['Adj. Close'] = all\_data.loc['2021':,'Adj Close'][0:len(preds)]

actual\_pred['prediction'] = preds[:,0]

from keras.metrics import MeanSquaredError

m = MeanSquaredError()

m.update\_state(np.array(actual\_pred['Adj. Close']),np.array(actual\_pred['prediction']))

return (m.result().numpy(), actual\_pred.plot() )

actual\_pred\_plot(rnn\_predictions)

def ts\_train\_test\_normalize(all\_data,time\_steps,for\_periods):

'''

input:

data: dataframe with dates and price data

output:

X\_train, y\_train: data from 2013/1/1-2020/12/31

X\_test: data from 2021 -

sc: insantiated MinMaxScaler object fit to the training data

'''

# create training and test set

ts\_train = all\_data[:'2020'].iloc[:,0:1].values

ts\_test = all\_data['2021':].iloc[:,0:1].values

ts\_train\_len = len(ts\_train)

ts\_test\_len = len(ts\_test)

# scale the data

from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler(feature\_range=(0,1))

ts\_train\_scaled = sc.fit\_transform(ts\_train)

# create training data of s samples and t time steps

X\_train = []

y\_train = []

y\_train\_stacked = []

for i in range(time\_steps,ts\_train\_len-1):

X\_train.append(ts\_train\_scaled[i-time\_steps:i,0])

y\_train.append(ts\_train\_scaled[i:i+for\_periods,0])

X\_train, y\_train = np.array(X\_train), np.array(y\_train)

# Reshaping X\_train for efficient modelling

X\_train = np.reshape(X\_train, (X\_train.shape[0],X\_train.shape[1],1))

inputs = pd.concat((all\_data["Adj Close"][:'2020'], all\_data["Adj Close"]['2021':]),axis=0).values

inputs = inputs[len(inputs)-len(ts\_test) - time\_steps:]

inputs = inputs.reshape(-1,1)

inputs = sc.transform(inputs)

# Preparing X\_test

X\_test = []

for i in range(time\_steps,ts\_test\_len+time\_steps-for\_periods):

X\_test.append(inputs[i-time\_steps:i,0])

X\_test = np.array(X\_test)

X\_test = np.reshape(X\_test, (X\_test.shape[0],X\_test.shape[1],1))

return X\_train, y\_train , X\_test, sc

def simple\_rnn\_model(X\_train, y\_train, X\_test, sc):

'''

create single layer rnn model trained on X\_train and y\_train

and make predictions on the X\_test data

'''

# create a model

from keras.models import Sequential

from keras.layers import Dense, SimpleRNN

my\_rnn\_model = Sequential()

my\_rnn\_model.add(SimpleRNN(32, return\_sequences=True))

#my\_rnn\_model.add(SimpleRNN(32, return\_sequences=True))

#my\_rnn\_model.add(SimpleRNN(32, return\_sequences=True))

my\_rnn\_model.add(SimpleRNN(32))

my\_rnn\_model.add(Dense(2)) # The time step of the output

my\_rnn\_model.compile(optimizer='rmsprop', loss='mean\_squared\_error')

# fit the RNN model

my\_rnn\_model.fit(X\_train, y\_train, epochs=100, batch\_size=150, verbose=0)

# Finalizing predictions

rnn\_predictions = my\_rnn\_model.predict(X\_test)

from sklearn.preprocessing import MinMaxScaler

rnn\_predictions = sc.inverse\_transform(rnn\_predictions)

return my\_rnn\_model, rnn\_predictions

X\_train, y\_train, X\_test, sc = ts\_train\_test\_normalize(all\_data,5,2)

my\_rnn\_model, rnn\_predictions\_2 = simple\_rnn\_model(X\_train, y\_train, X\_test, sc)

rnn\_predictions\_2[1:10]

actual\_pred\_plot(rnn\_predictions\_2)

#Simple LSTM

!pip install tensorflow

def LSTM\_model(X\_train, y\_train, X\_test, sc):

# create a model

from keras.models import Sequential

from keras.layers import Dense, SimpleRNN, GRU, LSTM

from tensorflow.keras.optimizers.legacy import SGD

# The LSTM architecture

my\_LSTM\_model = Sequential()

my\_LSTM\_model.add(LSTM(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1],1), activation='tanh'))

my\_LSTM\_model.add(LSTM(units=50, activation='tanh'))

my\_LSTM\_model.add(Dense(units=2))

# Compiling

my\_LSTM\_model.compile(optimizer=SGD(lr=0.01, decay=1e-7, momentum=0.9, nesterov=False),loss='mean\_squared\_error')

# Fitting to the training set

my\_LSTM\_model.fit(X\_train,y\_train,epochs=50,batch\_size=150, verbose=1)

LSTM\_prediction = my\_LSTM\_model.predict(X\_test)

LSTM\_prediction = sc.inverse\_transform(LSTM\_prediction)

return my\_LSTM\_model, LSTM\_prediction

my\_LSTM\_model, LSTM\_prediction = LSTM\_model(X\_train, y\_train, X\_test, sc)

LSTM\_prediction[1:10]

actual\_pred\_plot(LSTM\_prediction)

#Simple GRU

def GRU\_model(X\_train, y\_train, X\_test, sc):

# create a model

from keras.models import Sequential

from keras.layers import Dense, SimpleRNN, GRU

from tensorflow.keras.optimizers.legacy import SGD

# The GRU architecture

my\_GRU\_model = Sequential()

# First GRU layer with Dropout regularisation

my\_GRU\_model.add(GRU(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1],1), activation='tanh'))

#my\_GRU\_model.add(GRU(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1],1), activation='tanh'))

#my\_GRU\_model.add(GRU(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1],1), activation='tanh'))

my\_GRU\_model.add(GRU(units=50, activation='tanh'))

my\_GRU\_model.add(Dense(units=2))

# Compiling the RNN

my\_GRU\_model.compile(optimizer=SGD(lr=0.01, decay=1e-7, momentum=0.9, nesterov=False),loss='mean\_squared\_error')

# Fitting to the training set

my\_GRU\_model.fit(X\_train,y\_train,epochs=50,batch\_size=150, verbose=1)

GRU\_prediction = my\_GRU\_model.predict(X\_test)

GRU\_prediction = sc.inverse\_transform(GRU\_prediction)

return my\_GRU\_model, GRU\_prediction

my\_GRU\_model, GRU\_prediction = GRU\_model(X\_train, y\_train, X\_test, sc)

GRU\_prediction[1:10]

actual\_pred\_plot(GRU\_prediction)

#Stacked LSTM

def LSTM\_model\_regularization(X\_train, y\_train, X\_test, sc):

# create a model

from keras.models import Sequential

from keras.layers import Dense, SimpleRNN, GRU, LSTM, Dropout

from tensorflow.keras.optimizers.legacy import SGD

# The LSTM architecture

my\_LSTM\_model = Sequential()

my\_LSTM\_model.add(LSTM(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1],1), activation='tanh'))

my\_LSTM\_model.add(LSTM(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1],1), activation='tanh'))

# my\_LSTM\_model.add(LSTM(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1],1), activation='tanh'))

my\_LSTM\_model.add(LSTM(units=50, activation='tanh'))

my\_LSTM\_model.add(Dropout(0.2))

my\_LSTM\_model.add(Dense(units=2))

# Compiling

my\_LSTM\_model.compile(optimizer=SGD(lr=0.01, decay=1e-7, momentum=0.9, nesterov=False),loss='mean\_squared\_error')

# Fitting to the training set

my\_LSTM\_model.fit(X\_train,y\_train,epochs=50,batch\_size=150, verbose=1)

LSTM\_prediction = my\_LSTM\_model.predict(X\_test)

LSTM\_prediction = sc.inverse\_transform(LSTM\_prediction)

return my\_LSTM\_model, LSTM\_prediction

my\_LSTM\_model, LSTM\_prediction = LSTM\_model\_regularization(X\_train, y\_train, X\_test, sc)

LSTM\_prediction[1:10]

actual\_pred\_plot(LSTM\_prediction)

#Stacked GRU

def GRU\_model\_regularization(X\_train, y\_train, X\_test, sc):

'''

create GRU model trained on X\_train and y\_train

and make predictions on the X\_test data

'''

# create a model

from keras.models import Sequential

from keras.layers import Dense, SimpleRNN, GRU

from tensorflow.keras.optimizers.legacy import SGD

from keras.layers import Dropout

# The GRU architecture

my\_GRU\_model = Sequential()

# First GRU layer with Dropout regularisation

my\_GRU\_model.add(GRU(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1],1), activation='tanh'))

my\_GRU\_model.add(Dropout(0.2))

# Second GRU layer

my\_GRU\_model.add(GRU(units=50, return\_sequences=True, activation='tanh'))

my\_GRU\_model.add(Dropout(0.2))

# Third GRU layer

my\_GRU\_model.add(GRU(units=50, return\_sequences=True, activation='tanh'))

my\_GRU\_model.add(Dropout(0.2))

# Fourth GRU layer

my\_GRU\_model.add(GRU(units=50, activation='tanh'))

my\_GRU\_model.add(Dropout(0.2))

# The output layer

my\_GRU\_model.add(Dense(units=2))

# Compiling the RNN

my\_GRU\_model.compile(optimizer=SGD(lr=0.01, decay=1e-7, momentum=0.9, nesterov=False),loss='mean\_squared\_error')

# Fitting to the training set

my\_GRU\_model.fit(X\_train,y\_train,epochs=50,batch\_size=150, verbose=1)

GRU\_predictions = my\_GRU\_model.predict(X\_test)

GRU\_predictions = sc.inverse\_transform(GRU\_predictions)

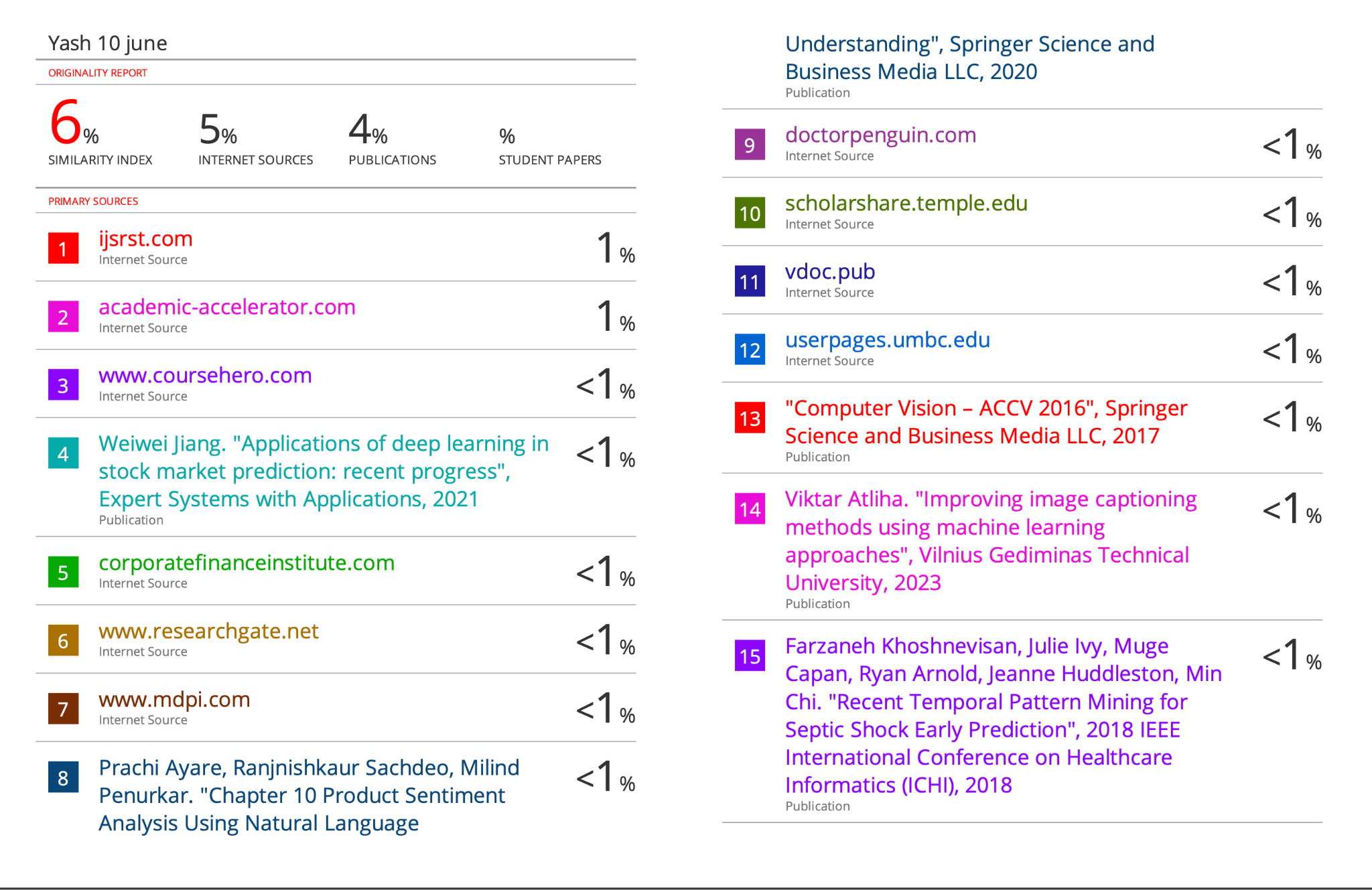
return my\_GRU\_model, GRU\_predictions

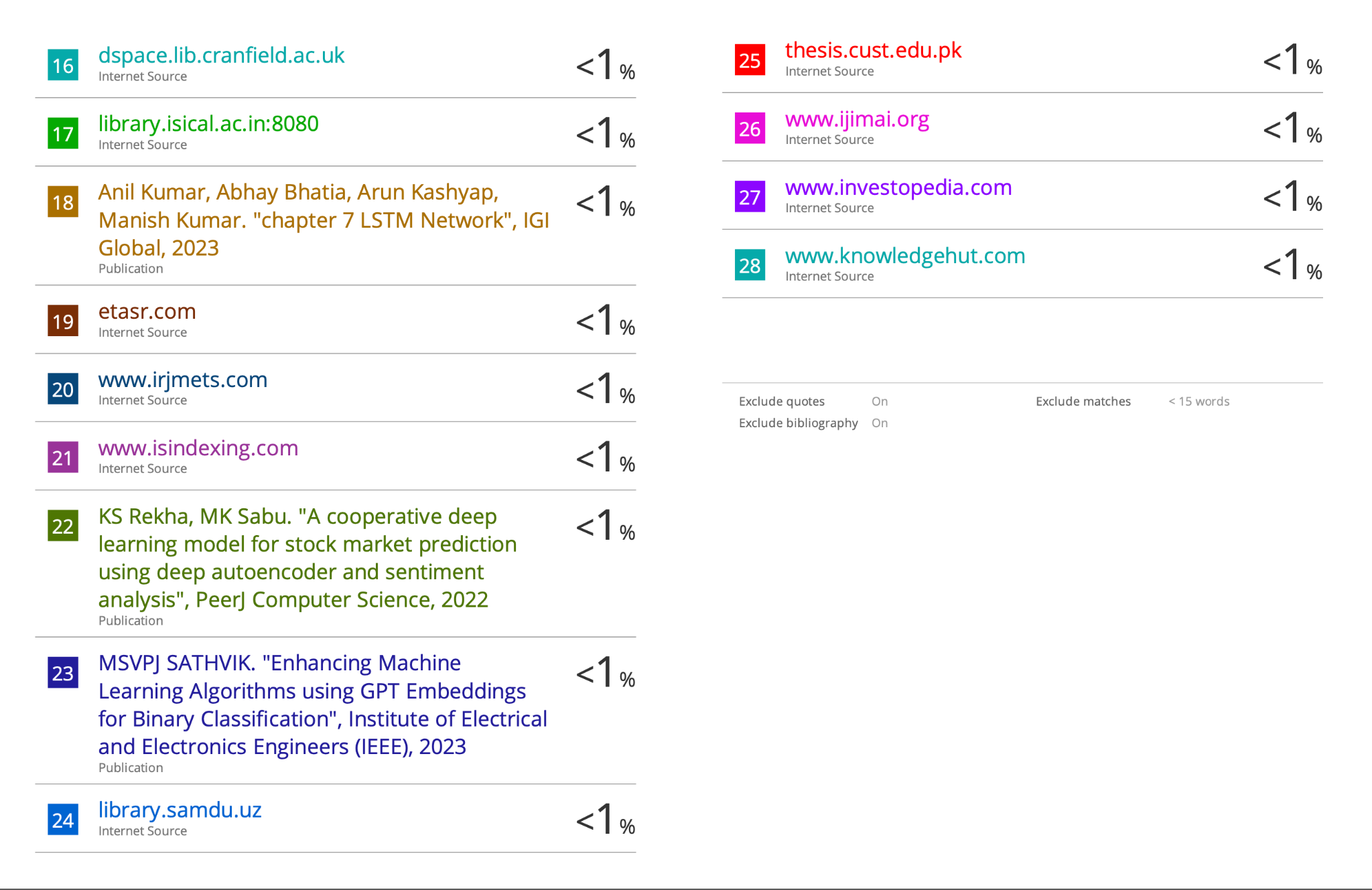
my\_GRU\_model, GRU\_predictions = GRU\_model\_regularization(X\_train, y\_train, X\_test, sc)

GRU\_predictions[1:10]

actual\_pred\_plot(GRU\_prediction)

**APPENDIX C**

****

****