Enhancing Option Trading Predictions on Nifty-50 Using Ensemble Technique

THESIS SUBMITTED TO

Symbiosis Institute of Geoinformatics

FOR PARTIAL FULFILLMENT OF THE M. Sc. DEGREE

By

PRAVEEN CHOUDHARY

(Batch 2023-2025/ PRN: 23070243065)

Symbiosis Institute of Geoinformatics

Symbiosis International (Deemed University)

5th Floor, Atur Centre, Gokhale Cross Road Model Colony,

Pune-411016

UNDERTAKING

The thesis titled **"Enhancing Option Trading Predictions on Nifty-50 Using Ensemble Technique"** is the Bonafede work of Mr. Praveen Choudhary, University can use and reuse the dissertation work in future.

Date: 13 September 2024

Name of the student: Praveen Choudhary

Signature

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF EQUATION

LIST OF TABLE

ABBREVIATION LIST

1. LSTM - Long Short-Term Memory
2. RMSE - Root Mean Square Error
3. ReLU - Rectified Linear Unit
4. MAE - Mean Absolute Error
5. NSEI - Nifty (National Stock Exchange Fifty)
6. Rs. - Indian Rupees (Currency Symbol)
7. GRU - Gated Recurrent Unit
8. MSE - Mean Squared Error

ACKNOWLEDGEMENT

I would like to pay my respect and profound regard to my internal guide of this study in this area **Sahil Shah** sir for extending his support and constant encouragement during the conduction of the research. The knowledge and experience he has shared in the project have been instrumental in coming up with this research work.

I would also like to extend my thanks to Symbiosis Institute of Geoinformatics and for providing me the opportunity and a conducive setting in which I could perform my study. The knowledge and experience gained during my time have been instrumental in shaping this project.

The project would have not been completed without the immense help and worthy experience. I was able to research and analyze a lot of topics during this project. It helped me expand my knowledge and skill.

In the end, I am thanking to my colleagues for their constant support and feedback which really helped me to refine my work.

- Praveen Choudhary

PREFACE

In the fast-paced world of daily options trading, precision is paramount. Traders must navigate the complexities of a highly volatile market, where the difference between success and failure can hinge on the ability to make timely and accurate decisions. Traditional trading indicators, while useful, often struggle to keep up with the rapid shifts in market dynamics, leading to missed opportunities or costly errors.

However, the aim of this research is to use several deep learning models to develop an advanced option trading indicator. Thus the aim is to develop an application that by analyzing historical data of the market provides traders with buy, sell and hold signals that are more effective and accurate than other approaches.

This is done with the help of the deep learning structures including networks, LSTM and GRU as well as with the help of a combined model based on the advantages of both the GRU and the LSTM. This is thanks to the flexibility of the ensemble approach of using many models in parallel to detect several patterns and temporal structures in data collected. Therefore, unlike in models mentioned above, the novelty of this project lies in the ensemble method, which involves the outcomes of GRU, LSTM as well as the proposed model GRU-LSTM model. In particular, the ensemble model works on these predictions in a very basic selection of the signal from the output of the various other models. This increases the reliability of the predictions eliminating the chance of relying on one model and equipping the trader with an instrument suitable for a fast-changing market.

This has led to the development of this advanced options trading indicator that is all geared towards ensuring that the daily trader acquires all the relevant information that may enhance his/her trading strategy hence resulting in better performance, lower risk factor and overall higher returns. This introduction precedes the detailed description of the methods used, difficulties met and achievements gained during the creation of this effective trading instrument.

Praveen Choudhary

INTRODUCTION

In the high risky daily options dealing environment where seconds count, traders usually come across the ask of dealing within a market that is highly volatile and full of complications. The nature of options trading in general exposes it to a higher level of risk bearing and return as compared to the other forms of trading and that is why the buying, selling and holding decision has to be accurate and precise. The problem with traditional trading indicators, however, is that they generally lack the versatility necessary to adapt to high-velocity, complex environments such as options markets. Such decisions end up being more of a compromise, and this means that opportunities are lost and, occasionally, lots of money too.

Two significant tools in the toolkits of AI/ML are data analysis and pattern search while dealing with large sets of data which cannot be deciphered through conventional methods in the financial trading field, this is why AI/ML provides new prospects in this sphere. Of these advancement, the subset of machine learning called Deep learning is viable in application that emphasize in pattern recognition and prediction. There are also several popular types of deep learning models, which steadily grow more important in the field of finance: these models are capable of sorting through large data, and identify hierarchal relations in it.

Turning to the identification of an accurate options trading signal, this work considers deep learning. For its specific purpose, it uses a multitude of deep learning models in order to produce buy, sell and hold signals that are more specific and accurate as compared to the simple synthetic signals. As an effort to pull off a robust and adaptive trading automation this work will capture Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRU), and a mixture of GRU & LSTM. Islam to Examine and Ekanbah Hossain (2021).

The ensemble technique combines every model's prediction with a view to enhance the reliability of trade signal. The ensemble mitigates severe or less desirable qualities of a certain model by using a model-based approach to determine the final trading signal as opposed to the output of one model. This guarantees that the highest often expected outcome throughout all the models is eventually selected and enables the trader to get a more likely signal.

Thus, the likelihood that this effort may change the daily problems solving process of options traders makes it important. The goal of this software is to allow traders to be armed with the knowledge they need in order to better navigate this sea of listed options through the use of modern deep learning. The final goal is to enhance traders' performance in trading by increasing the level of accuracy of the market forecasts and reducing the probability of errors that hinder the possibility of getting better earnings.

In the introduction part the project's goals are stated, the technologies behind the project, and potential impact of the project on the options trading market is discussed. The subsequent segments shall dwell into the nature of the particular approaches that were used, the challenges that were experienced and the results that were achieved offering a comprehensive analysis with details on the development and implementation of this complex trading signal.

PROBLEM STATEMENT

This is important because it presents the possibility of changing the daily decision making of options traders. The goal of this software is to make traders better prepared to navigate the complexities of the options market through giving them a way to utilize current deep learning algorithms. The general goal is the enhanced trading performance due to increased predictions' reliability and reduced risks of mistakes that can be achieved to enhance traders' financial outcome.

At first, we provide an overview of the project over its purpose for existence, of the technologies employed and the expected potential influence on the options trading market. The next sections will discuss details of the particular strategies applied, issues met and the outcomes to give a clear picture of how this involved trading signal was developed and then implemented.

OBJECTIVE

- **Maximize Profitability:** They should derive a higher level of sophistication for the options trading indicator that would assist the traders to arrive at better buy/sell/hold recommendations. The idea is to employ a complex of deep learning models at the same time and to find various intricate tendencies in the market to generate more precise prediction and, consequently, build more efficient revenues.

- **Reduce Risk:** Minimise the possibility of monetary losses on day to day options trading by providing more consistent and frequent trading signals. When the market is volatile, the ensemble model's approach of the model minimizes the mistakes that might occur. It helps the trader do not make mistakes that can be costly since it gathers the forecast from different deep-learning architectures.

- **Learn from Previous Market Data:** Include historical market data through which the deep learning models will be trained so that they recognize various trends and patterns. The goal of the project is to develop an application capable of identifying historical behavior and reflecting changes in the market environment in order to maintain the effectiveness of trading signals.

- **Low-Risk Trading Approach:** Offer traders a versatile and effective signal to trade indicator that will allow one to adopt a low risk strategy. The project aims at minimizing the level of uncertainty involved in options trading through using the power of GRU, LSTM, and the composite model of GRU-LSTM to give traders better guessing angles regarding the options traded in the market.

- **Real-Time 5-Minute Predictions:** Refresh three deep learning models every 5 mins To forecast movements in the next 5 mins of the market. It is centred on creating signals that predict the future trading operations; traders are able to act promptly and effectively in response to short-term shift in market as accurately as possible.

- **Address Volatility in Options Trading:** It is also important to mention that options are extremely unpredictable, therefore the ability to predict fairly high. To address this issue, the project plans to use models tailored for capturing cited changes in the prices for options and is therefore likely to provide rather accurate estimates while reflecting more turbulence in the stock market environments.

EXPECTED OUTCOMES

- It should lead to some of the following outcomes; The advanced option trading indicator is likely to enhance the profitability of buy, sell and hold decisions because each of the deep learning models within its complicated set of features can identify subtle and complex trends in the market effectively and capitalize on them. What is expected is that this improvement in decision-making would result in greater resilience in the profitability formation under the varying market conditions.

- Also, the project will minimize the hazards of daily options trading from a financial perspective. The indicator will make trading decisions more accurate, as it combines the predictions of several deep learning models with the help of which it will be possible to reduce the risk of greatly costly mistakes and provide traders with safer decision-making even in rather illiquid conditions.

- These deep learning models are powered by the historical market data and they will keep on improving and evolving with data from the market. This flexibility helps to keep the trading indicator relevant in the long-run providing not only accurate patterns which take into account past performance but also current market conditions.

- Besides, those fulfil the function of making decisions more knowledgeable and certain, which will help to maintain the condition of a low-risk trading. The combination of GRU, LSTM and hybrid GRU-LSTM models will make the implementation more robust, reduce ambiguity and help traders be more careful and cautious towards the options trading which ultimately leads to a tremendous long-term impact on the overall financial market.

LITERATURE REVIEW

Analyzing the research on financial trading it can be observed that models of deep learning including GRU and LSTM networks exhibit applicability in predicting the trends in the markets as they are able to handle all forms of sequential data and identify complex patterns. In volatility forecasting and stock price prediction among other applications, these models have been seen to outcompete the traditional approaches. However, there are still problems, particularly when getting to options trading because it has more complexity in its mechanics.

While working on the patient data, hybrid models which include the GRU, LSTM, and neural network structure have been described below to enhance the accuracy of the model. Some of the methods of ensemble learning, which consist of several models' predictions' integrating, have also demonstrated the potential for reducing risks and increasing robustness. However, there are still some issues such as the interpretability of the models, flexibility of the models in front of the changes in the market conditions, the lack of enough attention to the peculiarity of options trading.

Dr. Manu et al (2020) study looks into the forecast of the Nifty 50 stock index using two different approaches: The two models used to achieve this are: The 'ARIMA model' and the 'Artificial Neural Networks model' The data used for the above models was split into two sets; the training data and the testing data in a ratio of 80/20 The data used was from 01/04/2013 to 31/03/2018 To ensure that the above models supplied good estimate of the stock prices the authors argue that there. The research work of **(Fathali et al, 2023)** deals with prediction of NIFTY-50 index stock prices with the help of deep-learning models; three types of models used here are RNN, LSTM, and CNN; these models are to examine historical data to acquire the forecast of future data streams. To this end, this project envisages filling the above gaps through designing an admixture of GRU, LSTM, and even the hybrid one to construct an evolved trading indicator for options trading based on historical market data that shall facilitate the attainment of superior profitability and lower risk levels. Also, the study carried out by Parminder Kaur & Ravi Singla (2022) presents a way to find the model selection metrics to forecast Nifty 50 closing prices with the help of ARIMA-GARCH models reveal only ARIMA (2,1,2) – EGARCH (1,1,1) as an optimal model to make static forecast that could be helpful in creating low risk trading portfolio for the portfolio manager as This study conducted by **(Goel at el, 2023)** aims to predict the Indian stock market (Nifty 50) using macroeconomic variables identified from literature across two periods: Cohort 1: Patients during the pre COVID-19 period, June 2011 to February 2020 and Cohort 2: Patients during COVID-19 period March 2020 to June 2021. Research methodology adopted here is secondary data collection method that relied on both government and regulatory data for the period 2011 to 2021. A forecasting technique that is used is an artificial neural network, and was trained using the scaled conjugate gradient approach, and the outcome is forecasted for the Nifty 50 index. From the results produced by the SCG algorithm it obtained a 96 percent reproduction of the natural uniformity of hair color. Before COVID-19 the accuracy of the model was about 99% and during the COVID -19 period the accuracy of the ANN model was about 99. 85% accuracy. The results of this study are useful for investors; the portfolio managers and any institutional investors. The study is quite valuable as very limited research has examined the application of ANN

models on predicting the Indian stock market, specifically during pre pre-COVID-19- and post-COVID times.

Evaluates stock price prediction for the NIFTY-50 dataset using two deep learning models by **(Hani et al, 2023)**: Comparing two types of RNNs: Long Short-Term Memory (LSTM) and Backward Elimination LSTM (BE-LSTM), the overall performance of which was evaluated with statistically increased accuracy of 95%, and the model was created to forecast the index's next 30 closing prices. The study by **(Gnanendra et al, 2022)** compares the efficiency of the Recurrent Neural Networks (RNN) having Long Short-Term Memory (LSTM) in the volatile and stable market conditions and in the wake of COVID-19 pandemic and research on the shortcomings of the RNN to predict large fluctuations in the market. In a study that was carried out by **(Jaydip et al, 2022)**, the use of machine learning and deep learning models in stock price prediction is examined taken specifically the NIFTY 50 index into account it is revealed that out of the four models that were implemented the best performance is achieved under the use of an univariate LSTM model which was used with actual one-week prior data and was used in predicting the next week's open values pointing. The study investigates by **(Akshat et el, 2023)** discusses the concept of a deep neural network in stock price prediction especially in the Nifty 50 index with the help of Recurrent Neural Networks like LSTM & GRU. As a result of the market noise and to avoid overfitting, the paper discusses about the combination of these models with the help of the ensemble method and enhance the performance with the help of tuning the parameters with PSO. Outcome reveals that LSTM-GRU attains accuracy of 56. 66% accuracy and this is actually increased to 57%. It achieves 72% accuracy when it is used with PSO-tuned LSTM model. They also show that precision has been improved from 0. 2882 to 0. 5485 such conclusions emphasizing the enhancement of model's ability to predict values by using ensembling along with PSO. Research conducted by **(Zahra et al, 2021)** Applying the three deep learning networks of RNN LSTM and CNN for predicting the NIFTY 50 stock prices on the Indian National Stock Exchange it can be said that The experimental analysis show that LSTM model has issued MSE and more performance compared to RNN and CNN On the basis of the above conclusion the present study reveals that LSTM is one of the best model among all the applied model for the stock market prediction.The study carried out by (**Sidra et al, 2020)** claim features to assert that Efficient Market Hypothesis (EMH) is far from being a possibility because they have demonstrated that randomness can be accurately predicted by designing models and have enunciated a statistical, machine learning, and deep learning paradigm and have formulated eight classification, and regression platforms besides a deep matrix, Long and Short-Term Memory (LSTM) is used to apprehend volatile price swings.

TABLE 1: Summary of Literature review

S.NO	Reference	Data	Methodology	Model Used	Result
1	(Ms. Shubhika , 2021)	Kaggle	The study used Feed Forward Back Propagation algorithm for the selected ANN model and used time series analysis, forecasting with the ARIMA (p,d,q) model by E-views 10 software package.	ARIMA ANN	The RMSE values were 0.015 for ANN and 0.07 for ARIMA. A lower RMSE indicates a better model performance, clearly favoring ANN over ARIMA in this study
2	(Zahra et al, 2022)	NSE	The historical data related to NIFTY 50 Index was used for the purpose of the present work. For evaluating their accuracy of prediction, CNN, LSTM, RNN models are trained and developed deep learning different measures are used.	Recurrent Neural Network, Long Short-Term Memory Network, Convolutional Neural Network	LSTM to be the best model for predicting NIFTY 50 movements, highlighting deeplearning potential in stock market forecasting.
3.	(Parminder et al, 2022)	Kaggle	The study collected and preprocessed historical NIFTY 50 Index data, then built and trained RNN, LSTM, and CNN models. These models were evaluated using metrics to measure their predictive performance.	LSTM, RNN and CNN	LSTM is best for predicting NIFTY 50, demonstrating deep learning's potential in stock market forecasting, suggesting ways to improve accuracy in future research.
4.	(Goel et al, 2023)	NSE	1. ANN analyzed historical stock data and	ANN	ANN to forecast NIFTY-50 closing prices effectively,

			forecasted future trends. 2. Multiple linear regression models evaluated factors like dollar price in stock price prediction. 3. Deep learning improved stock price prediction accuracy and market responsiveness.		highlighting Model 1(65,35)'s superior performance. Deep learning improved investment strategies, and linear regressions emphasized the dollar price's predictive role.
5	(Hani El-Chaarani et al, 2023)	NSE	Data preprocessing included quality checks and consistency measures. Backward Elimination identified relevant variables. After dividing the dataset into the training and testing data sets, an LSTM model was put into use to make predictions.	LSTM Model with Backward Elimination	Backward Elimination with LSTM improved NIFTY 50 Index Price predictions over LSTM without feature selection. The study underscores AI's role in enhancing stock market forecasting through innovative techniques.
5	(Gnanendra et al, 2022)	Finance	Univariate analysis and employed LSTM-based RNNs for forecasting. Data were split into 80% training and 20% validation sets to predict future prices in volatile markets, addressing gaps in machine learning prediction models.	RNN and LSTM	LSTM-based neural network models outperforming traditional methods in forecasting NIFTY 50 prices during market volatility, underscoring the need for enhanced predictive capabilities in financial applications.
6	(Jaydip Sen et al, 2022)	NSE	Create a regression-based	LSTM, MARS,	LSTM-based deep learning models'

			forecasting system by gathering past index values and predicting the daily movement of the NIFTY-50 price. The study involved building and evaluating multiple regression models and LSTM models to predict stock prices.	Bagging, XGBoost, RF, ANN and SVM	effectiveness in accurately predicting stock prices highlights the significance of sophisticated modeling methods in this field.
7	(Akshat et al, 2023)	Finance	Use LSTM and GRU architectures with various optimizers and dataset sizes. PSO was used for hyperparameter optimization of these models. Ensembling techniques were employed to leverage the strengths of different models. Evaluation metrics included Accuracy, Precision, Recall, and F1-Score to assess model performance.	(PSO + LSTM) + (LSTM + GRU)	PSO for hyperparameter tuning and ensembling in stock price prediction, achieving 57.72% accuracy. Future work aims to enhance accuracy through advanced time series models and sentiment analysis.
8	(Zahra Kodia et al, 2021)	Finance	The production of output, analysis, and neural network models depending on different assessment measures. The models were trained on daily stock price data,	RNN, LSTM and RNN	LSTM models effective for predicting stock price movements. It recommended combining models for better accuracy, optimizing hyperparameters, and integrating deep learning with multi-

			and different activation functions were examined for the network's layers.		agent systems for enhanced precision in predictions.
9	(Sidra Mehtab et al, 2020)	NSE	Using regression models based on deep learning, the goal was to create predictive frameworks for the daily values of the NIFTY-50 index. LSTM and CNN architectures were employed with different input data shapes and forecasting methods.	CNN and LSTM	The accuracy and reliability of predictive models for forecasting stock prices. Conclusions and future research directions were discussed, emphasizing the models' potential applications in financial forecasting.
10	(Jaydip Sen et al, 2020)	NSE	Constructing and testing various predictive models using machine learning techniques and LSTM networks. The models were trained on historical data and used to forecast future stock prices	MARS, Bagging, XGBoost, RF, ANN and SVM	The predictive models' performance results demonstrate how well machine learning and deep learning techniques work to anticipate stock values, especially the NIFTY-50 index.

METHODOLOGY

In the context of this research, we employ deep learning models to predict short-term price movements, which is a new concept in the field of financial analysis especially in the next ten minutes. Options indicators have in the past generated trading indications by analysis and predetermined methods. To enhance the outcomes, our proposed approach uses enhanced deep learning architectures including hybrid models, LSTMs and GRU.

This model operates a continuous cycle, where it is trained on the most recent data every 5 minutes. This real-time training process allows the model to adapt quickly to the evolving market

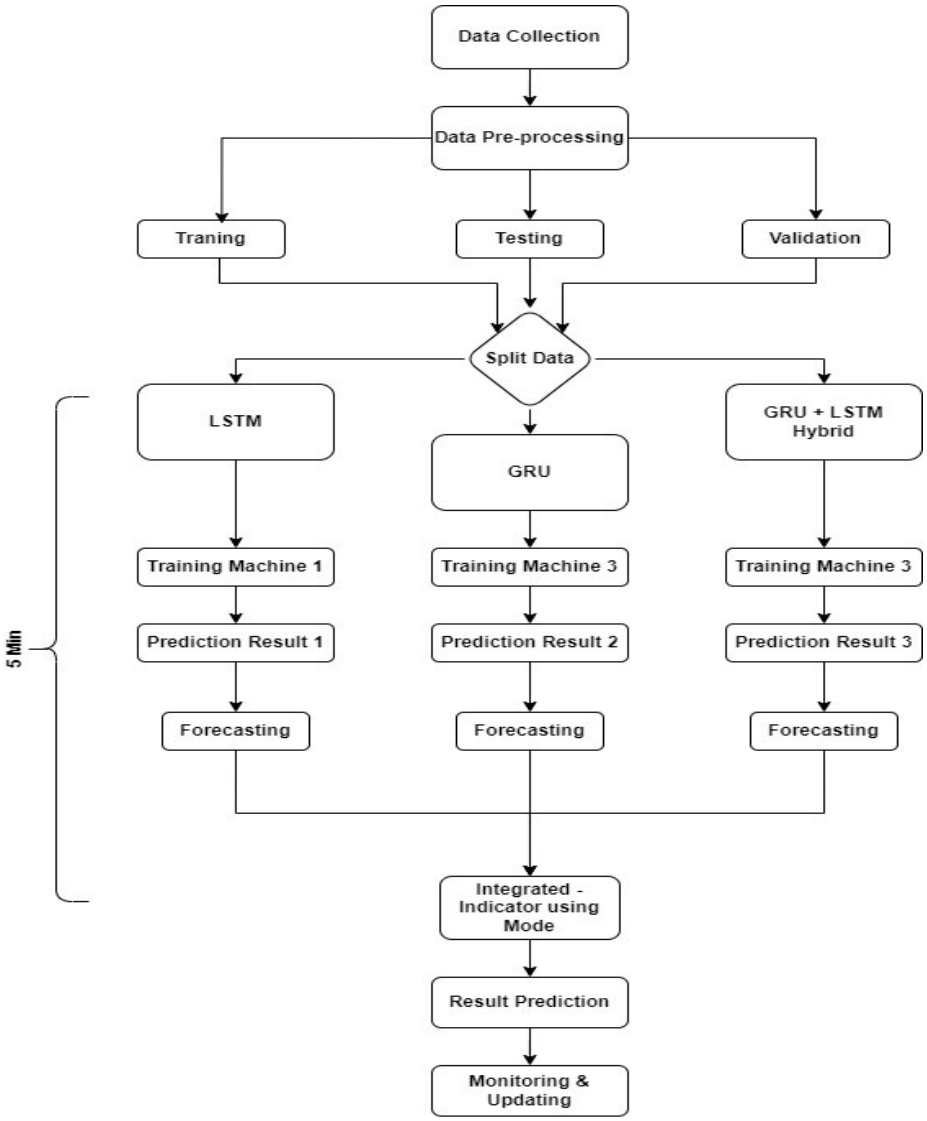

Figure 1 – Methodology Flowchart

conditions. After each 5-minute training session, For the following ten minutes, the model forecasts the price changes. These predictions are then fed into an indicator system that generates actionable buy, sell, or hold signals for traders. This cyclical process repeats every 5 minutes, ensuring that the trading signals are always based on the latest market data. This approach represents a groundbreaking advancement in the domain, offering a highly dynamic and responsive trading strategy.

DATA COLLECTION:

For this project, we used a secondary data collection method, sourcing our dataset from Kaggle dataset, a reputable platform that provides high-quality and reliable datasets. The dataset selected for this study pertains to the Nifty-50 index, covering January 2015 to January 2024. It contains 836,758 entries, each representing a 1-minute time interval.

The dataset includes the following key columns:

Table 2: Details of the Data

	Details
DATE	The specific date and time of each data point.
OPEN	the Nifty-50 index's opening price at the beginning of each minute.
LOW	The highest price is reached during each minute.
CLOSE	The Nifty-50 index closing price at the conclusion of each minute.

Table 3: Data Of Nifty- 50 (1-min)

Date	Open	High	Low	Close
09-01-2015 09:15	8285.45	8295.9	8285.45	8292.1
09-01-2015 09:16	8292.6	8293.6	8287.2	8288.15
09-01-2015 09:17	8287.4	8293.9	8287.4	8293.9
09-01-2015 09:18	8294.25	8300.65	8293.9	8300.65
09-01-2015 09:19	8300.6	8301.3	8298.75	8301.2
09-01-2015 09:20	8300.5	8303	8298.6	8300

Since Kaggle was the source of the data, there are 836758 rows and 5 columns (date, high, low, open, and close). Given that it provides a strong foundation for the training and testing of our deep learning models, we may assume with confidence that it is both clean and current. Our model's training is based on this historical dataset, which helps it learn from previous market behaviours and project future values with accuracy.

DATA PRE-PROCESSING:

The data utilized for this project was sourced from a Kaggle dataset, specifically focusing on historical Nifty-50 stock data with 1-minute intervals. Given the reputable nature of Kaggle's financial databases, the dataset was largely clean and required minimal data cleaning. However, to better align the data with our analysis objectives, the 1-minute interval data was aggregated into 5-minute intervals by resampling, using the mean price and summing volumes over each 5-minute period. This allowed for a more manageable and meaningful time frame for our analysis while maintaining the integrity of the trends in the stock prices.

INITIAL DATA CHECKS:

- Null Values: The dataset was examined for null values. As expected from a reliable financial source, no null values were found.
- Duplicate Values: We also checked for duplicates and confirmed that there were none in the dataset.
- Data Types: We ensured that the 'Date' column was of the DateTime type, while all other columns were correctly typed.

DATA TRANSFORMATION:

1. **Resample:** The data resample from a 1-minute to 5 minutes. This adjustment aligns with the trading strategy and allows for a more manageable dataset.

Table 4: Data Of Nifty- 50 (5-min)

date	open	high	low	close
09-01-2015 09:15	8285.45	8295.9	8285.45	8292.1
09-01-2015 09:20	8300.5	8303	8298.6	8300
09-01-2015 09:25	8301.65	8302.55	8294.9	8295
09-01-2015 09:30	8294.1	8295.75	8288.1	8289.75
09-01-2015 09:35	8289.1	8290.45	8283.8	8285.25
09-01-2015 09:40	8283.4	8286.05	8281.65	8284.95

2. **Indicator Calculation:** We introduced three new columns to calculate Bollinger Bands, which are critical for generating buy/hold/sell signals:
    - UPPER BAND: (20-Day SD × 2) + 20-Day SMA
    - MIDDLE BAND: 20-Day SMA
    - LOWER BAND: - [(20-day SD × 2) - 20-Day SMA
3. **Data Reduction:** Columns that were not needed for the model—such as High, Low, and Open—were dropped, leaving us with four columns: Date, Close, Upper Band, and Lower Band.

    After preprocessing, the dataset was reduced to 167,366 rows and four columns. Given that our model will be trained every 5 minutes to predict buy/sell/hold signals, we need a smaller, more focused subset of this data to ensure efficient training and enhanced prediction accuracy.

4. **Signal Indicator:** Create a signal based on the Bollinger Bands and the average close price section, which generates a new column called "Signal" with values of either "Buy" or "Sell." We will not include a "Hold" signal, as it is too volatile in options trading. This signal will help in checking the model accuracy is it the model predicts well or not.

Table 5: Data Of Nifty- 50 For Signal

Date	Open	High	Low	Close	ShortMa	LongMa	RSI	macd	macd_signal	Signal
01-01-2024 10:50	21737.6	21741.9	21732.2	21736.9	21728.7	21714.0	64.65	8.435	6.2475	buy
01-01-2024 10:55	21737.2	21739.7	21733.1	21736.1	21732.2	21716.1	58.55	8.984	6.7949	buy
01-01-2024 11:00	21736.7	21742.6	21726.4	21729.5	21733.4	21717.8	48.91	8.786	7.19320	buy
01-01-2024 11:05	21729.4	21739.6	21727.8	21737.3	21735.5	21719.6	56.11	9.148	7.58430	buy
01-01-2024 11:10	21741.9	21741.9	21732.2	21741.2	21736.2	21721.5	60.06	9.643	7.99616	buy
01-01-2024 11:15	21740.9	21747.1	21731.6	21742.1	21737.2	21724.2	66.58	9.989	8.3947	sell

(source: https://tradewithpython.com/generating-buy-sell-signals-using-python)

5. **Defining the Target Variable:** The target variable should reflect the signal you want to predict. In this case, we aim to predict whether the market will generate a buy, sell, or hold signals in the future.

Define the Target Variable:

Create Future Signal Labels: For each row, determine the signal for the next time step (e.g., the next 5-minute interval). This involves shifting the signal column to align with future values.

Assign Signals Based on Future Data: Label each row with a target signal (e.g., buy, sell, hold) based on the future direction of the price relative to the calculated Bollinger Bands.

MODEL BUILDING:

Stock Indicator Using (Deep Learning Model)

We have utilized three deep learning models for predicting buy, sell, and hold indicators:

1. **LSTM** (Long Short-Term Memory)
2. **GRU** (Gated Recurrent Unit)
3. **GRU+LSTM** (Hybrid)

LSTM:

It is also identified that there are certain areas like fading gradient issues, which are associated with the traditional type of RNN and due to which a certain particular type of RNN known as LSTM network has been invented. Long-term support vector machines actually known as LSTMs was introduced by Hochreiter and Schmidhuber in 1997 and is used nowadays to learn long term dependencies in every sequential data.

Two optimization techniques that are employed for LSTMs training are; gradient descent and over time backpropagation. These techniques adjusted the weights within the circuitry of the network depending on the error gradient. While the traditional approach to RNN design provides only one layer and involves the tanh activation function as a rule, LSTMs come with four main levels to regulate the stream of information.

The core of an LSTM is the cell state, which remains mostly unchanged throughout all timesteps. Through a sequence of gates that are individually managed by sigmoid neural network layers, information may be added to or withdrawn from the cell state. In order to determine how much information should travel through, these gates provide outputs between 0 and 1; 0 indicates "let nothing through," while 1 indicates "let everything through."

The LSTM process consists of three main steps:

1. Forget-Gate: determines, which information will be purged from the cell's state. Sigmod function is used to process the input from the previous timestep and the current one and always gives the output between 0 & 1's. The equation is:

$$f_t = b_f + \sigma(W_f.[h_{t-1}, x_t]$$

2. Input Gate: Determines from where fresh data will be obtained to be incorporated into the cell state. A tanh layer generates potential values for addition after a sigmoid layer decided which of the values should be updated. The equations are:

$$i_t = b_i + \sigma(W_t.[h_{t-1}, x_t]$$

$$C_t = b_c + tanh(W_c.[h_{t-1}, x_t]$$

3. Modify the output and cell state: update the cell state after computing the outcome. The new cell state is calculated as follows after combining the old state and fresh data:

$$C_t = f_t * C_{t-1} + i_t * C_t$$

The information to be extracted from the cell state is determined by a second sigmoid layer, and the results are scaled between -1 and 1 using a tanh function. The following is the final output equation:

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o$$

$$h_t = o_t * tanh(C_t)$$

(Islam, & Hossain, E., 2021)

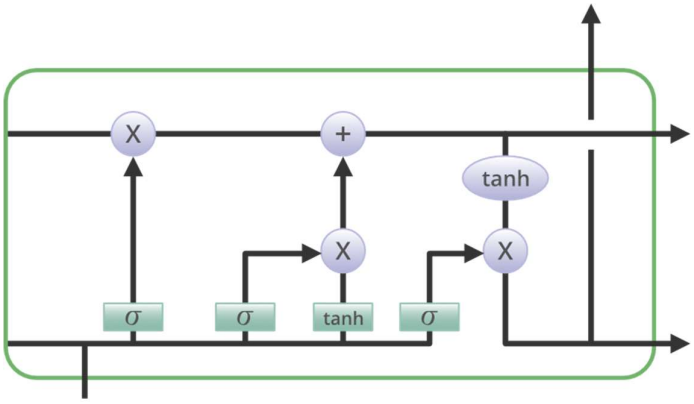

Figure 2 – LSTM-Architecture

(Source : https://www.geeksforgeeks.org/understanding-of-lstm-networks/)

For instance, to explain how LSTM model is employed in predicting and analyzing stock price indicators, the Nifty 50 index is employed in the study. First of all, cleaning of the data set is required: the data set includes both a training dataset and a test dataset. In order to make more meaningful comparison the MinMaxScaler is applied to scale the data between 0 and 1.

Table 6 : LSTM model architecture

Layer	Description	Units	Activation
Initial LSTM	LSTM Layer	256	-
Second LSTM	LSTM Layer	128	-
Dropout-Layer	Applied after each LSTM layer for regularization	-	-
Dropout-Rate	Dropout-Rate	0.1	-
Dense-Layer	Fully Connected Dense Layer	64	ReLU
Output Layer	Output Layer for Regression Tasks	-	Linear

The model is created using Adam optimizer; Two additional metrics included – Mean Absolute Error (MAE) The loss function was set to be Mean Square Error (MSE). The problem of over construction of the model can be avoided by introducing early stopping so that the parameter val_loss can be analyzed with respect to epochs through epoch 25. Training takes place in a number of batches given by 'batch_size' and extends to 300 epochs. If a coherent and measurable change has been made then the weights of the ideal model are put back into its best form.

As for the model evaluation, the R-squared score is applied to measure how close the data of the expected and actual values are. As for the anticipated values, they are scaled back to its original scale with the help of applying the inverse transform of MinMaxScaler. The close, upper, and lower band are the measures that can be employed for the evaluation of the performance of the developed model and all the presented measures are forecasted utilizing the different model for the next 10 minutes values only. As a way of displaying the results for each column, this paper provides the graphical representation of the actual and the predicted stock prices.

GRU:

Some of common modifications of the RNN that were invented to overcome vanishing gradient issue with standard RNNs include the Gated Recurrent Unit (GRU) that was developed by Cho et al. in the year 2014. Here, GRUs are a little bit favourable because it is similar to LSTM, but it has fewer parameters in it.

Three essential parts make up the GRU architecture: a tanh layer, an update gate and a reset gate. Combined, these elements regulate the access to the information and solve the problem of disappearing gradient.

1. **Update-Gate:** Next time steps' historical data varies depending on the update gate so as whether there's need to store it for the subsequent occasions. It is calculated using:

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

   Here, After every merge of x_t and previous hidden state h_(t-1), the same is passed through sigmoid activation function through which output values will zone between 0 and 1.

2. **Reset Gate:** This mechanism controls the amount of data which should be lost in other to ensure that only accurate data makes it to the historical records. It is computed as follows:It is computed as follows:

   $r_t = \sigma(W_t.[h_{t-1}, x_t])$

   This is similar to the update gate in the sense that it provides x_t times h_(t-1), multiplies it by a weight and passes the output through a sigmoid function to yield a value between 0 and 1.

3. **Current Memory Content:** Current Memory Content: Based on reset gate, this component generates new memory. The following formula is applied:The following formula is applied.
   $\tilde{h}_t = tanh(W.[r_t * h_{t-1}, x_t])$

   The reset gate r_t regulates the prior cell state, h_(t-1). New memory content considers a tanh activation when added to a current input, x_t, and forms the new memory content.

Final Memory Output: Update-gate controls the mixing of the old and new memory contents to generate the final output, h_t:

$z_t = z_t * \tilde{h}_t + (1 - z_t) * h_{t-1}$

If z_t is close to 0 the current content is ignored and the previous data retained, when it is close to 1 all the current content is kept. On the other hand, as z_t gets closer to 1, the model relies heavily on the new memory that has just been loaded for the update of the representation.

(Islam, & Hossain, E., 2021)

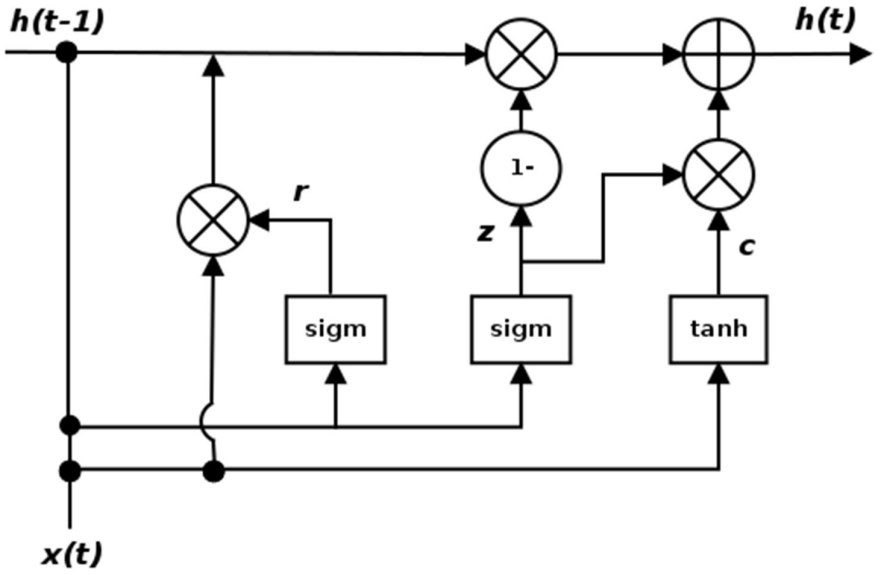

Figure 3 GRU-Architecture

(Source : https://www.geeksforgeeks.org/understanding-of-lstm-networks/)

The following describes a work with focus on GRU (Gated Recurrent Unit) models for the forecast of Nifty-50 index's stock price indicators. This means that values should be normalized in order to be falling between 0 and 1 and so before performing this, the data set must be divided into training and test data sets. Normalisation is then done using the MinMaxScaler in order to get all the features between 0 and 1.

Table 7 : GRU Model Architecture

Layer	Description	Units	Activation
Initial GRU	GRU Layer, configured to return sequences	256	-
Dropout-Layer	Used to regularize data at a pace of 0.1	-	-
Second GRU	GRU Layer	128	-
Dropout-Layer	Used to regularize data at a pace of 0.1	-	-
Dense-Layer	Fully Connected Dense Layer	32	ReLU
Output Layer	Output Layer for Regression Tasks	1	Linear

More metrics that are included in the loss function of the model are the Mean Absolute Error (MAE) and the Mean Squared Error (MSE). 300 epochs are used for training with a batch size of batch_size. To avoid overfitting, validation loss (val_loss) is monitored by early pausing with a 25 epoch patience. The optimal model weights are restored in the case that an improvement is found.

The last measure applied in the evaluation of the effectiveness of the model is R-squared that measures the adherence of the computed predictions to the actual value. In order to bring the predictions back to their original scale, the use of the inverse transform function of MinMaxScaler is made. The report includes plots that show how actual stock prices compare to the predicted values for each indicator, which are the closing, upper, and lower bands. To estimate the values for the next 10 minutes, several models are used for each indicator.

LSTM + GRU (Hybrid):

In consideration of the Nifty-50 index stock, the practical improvisation of a hybrid LSTM-GRU model for stock price anticipations is described in the study. To enjoy the benefits of the above architectures, it uses both GRU and LSTM layers in the model's structure. (Source: Islam, & Hossain, E., 2021)

When preparing the dataset, it is divided into the training and test datasets. The second one is to use the MinMaxScaler in order to scale values for making them normalized where the range of values is between 0 and 1.

The hybrid model architecture includes:

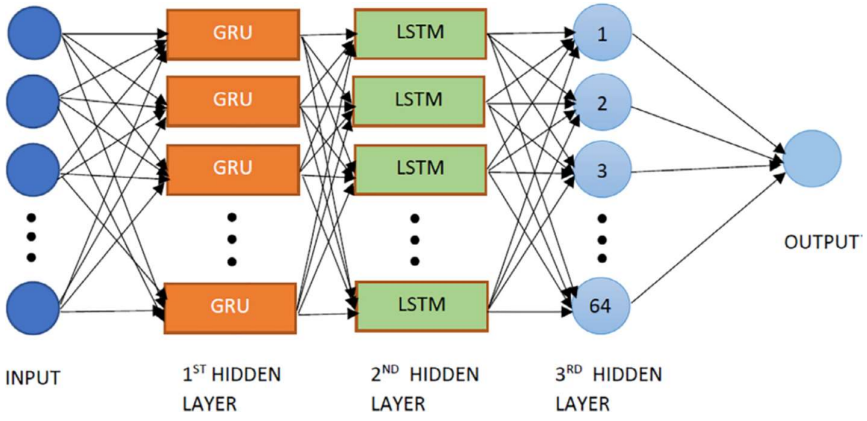

Figure 4 –Internal Structure Hybrid Model.

(Source : https://www.geeksforgeeks.org/understanding-of-lstm-networks/)

Table 8 : GRU+LSTM Model Architecture

Layer	Description	Units	Activation
Initial GRU	GRU Layer, configured to return sequences	256	-
Dropout-Layer	Used to regularize data at a pace of 0.1	-	-
LSTM-Layer	LSTM Layer	128	-
Dropout-Layer	Used to regularize data at a pace of 0.1	-	-
Dense-Layer	Fully Connected Dense Layer	64	ReLU
Output Layer	Output Layer for Regression Tasks	1	Linear

Adam optimiser is used to build the model, which also uses MSE and other metrics of MAE and MSE as loss functions. It is trained over 300 epochs with batch_size determining the batch size. Early stopping is used to monitor validation loss (val_loss) across a patient's 25 epochs in order to reduce overfitting. The ideal model weights are reinstated if a noticeable improvement is observed.

The model's performance is assessed using the R-squared score, which calculates how closely the predictions match the actual values. The expected values are then restored to their original scale using the MinMaxScaler inverse transform function. Plots comparing actual stock prices to the projected values for each indicator are included in the report to visualize the results.

Different models are employed to forecast the closing, upper, and lower bands' values for the following ten minutes.

(Akshat et. Al., 2023)

Function For Indicator:

- Buy
- Sell
- Hold

The Bollinger Bands and current price patterns serve as the basis for the indicator function that generates Buy, Sell, or Hold recommendations. The following is a summary of the process:

1. **Determine the Mean of Previous 10 Data Points:**
   Using the past ten data points, calculate the average value for the Upper Band (UB) and Lower Band (LB).

2. **Generate Signals:**

- **Buy Signal:** released in the event that the closing price is equal to or less than the average of the previous ten data points for both LB and UB.

Buy if $\frac{\sum_{i=1}^{10} UB_i}{10} \leq Close$

- **Hold Signal:** Issued if:
  - The Upper Band (UB) is less than the Lower Band (LB), or
  - The last ten data point averages for UB are higher than the closing price as of right now, whereas the last ten data point averages for LB are lower.

Hold if UB < LB or ($\frac{\sum_{i=1}^{10} UB_i}{10} > Close \; and \; \frac{\sum_{i=1}^{10} LB_i}{10} < Close$)

(Source: Jafar et al, 2023)

- **Sell Signal:** Issued if none of the above conditions are met.

Sell if else

This methodology uses the Bollinger Bands to assess market conditions and determine the appropriate trading action based on recent price movements and band values.

Taking the mode, or majority vote, from the signals produced by all three models determines the ultimate course of action: buy, sell, or hold. The market trend for the following ten minutes is indicated by this combined signal.

(source: https://tradewithpython.com/generating-buy-sell-signals-using-python)

➢ **Indicator Working**

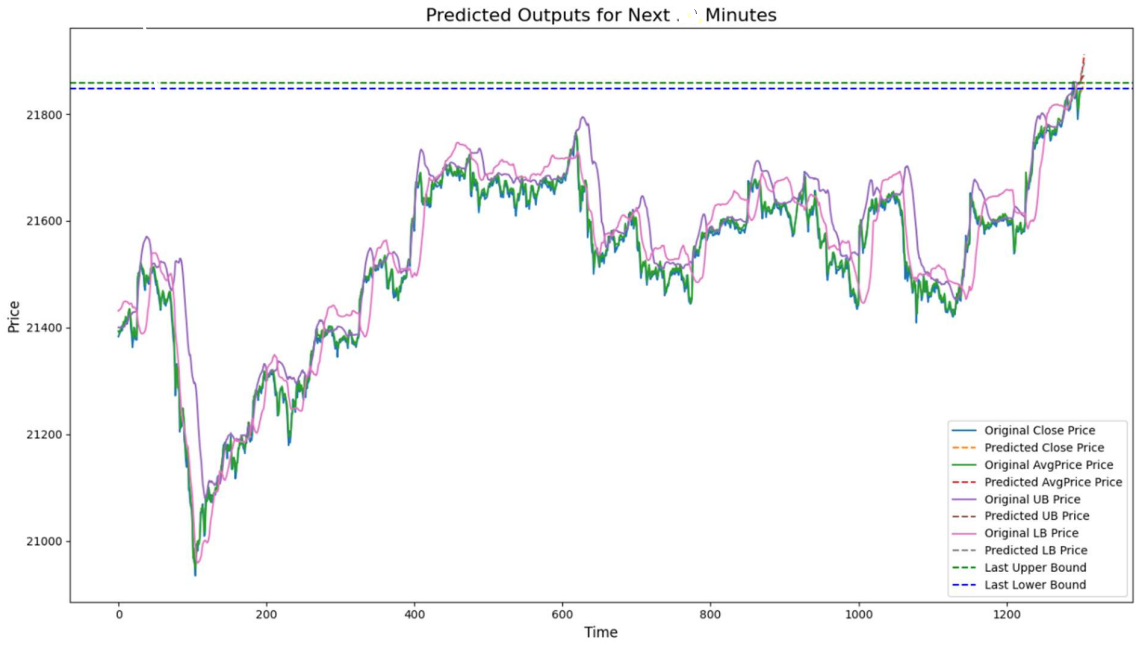

Figure 4 –Internal Structure Hybrid Model.

In this chart, we can see two horizontal lines that help establish support and resistance levels for the close price. Based on these two horizontal lines, the system generates Buy, Sell, and Hold signals.

TRAINING PROCESS:

The training process involves several steps to efficiently train and evaluate multiple models within a specified timeframe. Initially, a subset of the data, spanning 10 to 15 days of 5-minute intervals, is extracted from the full dataset which a length of 543-768. This subset serves as the training data for the models.

1. **Training Data**: 65% of the dataset.
2. **Testing Data**: 35% of the dataset.
3. **Validation Data**: TensorFlow "model.fit" function uses the validation data (validation_data = (X_test, y_test)) from the testing set provided. the validation data in this case is 35% of the total dataset, as it is taken from the test split.

Three different models are trained on this subset of data:

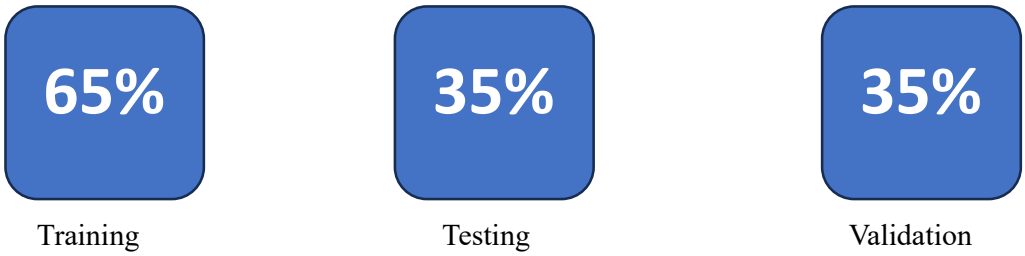

Figure 6: Details Of Dataset.

1. LSTM-Model: To identify long-term dependencies in the data, a Long Short-Term Memory network is trained.

2. GRU-Model: A Gated Recurrent Unit network is utilized to manage vanishing gradient issues and learn temporal patterns.

3. Hybrid-Model: A combination of LSTM and GRU layers is employed to leverage the advantages of both architectures.

Given that each model requires approximately 4 minutes to complete its training process, different machines are used to train the LSTM, GRU, and hybrid models concurrently. This parallel training approach ensures that all models are trained efficiently and within the desired timeframe.

RESULT

Comparison Of 3 Model With Close, Upper And Lower Band:

1. **GRU:**

The results obtained from Nifty 50 dataset were fine on checking several indicators such as closing price, UB and LB through implementing a trained model of GRU. As for the last index, the cost, model's Root Mean Square Error (RMSE) was equaled to 12. Performing the same iterations as those done above, but on the training set, results in cross-validation accuracy of 5 058 on the training set and 14. Equation 3: 791 on the test set and Get MSE similar to; 145 on the test set. 392 and 218. 783 and Mean Absolute error to be 7. 927 and 9. 907 respectively. With RMSEs of 3. 936 and 3. and the corresponding MSEs were 797 for the training set and 802 for the test set while the MSEs are 15. 491 and 14. Foreseen are expenses of 452 thousand USD as well as MAEs of 2 thousand USD. 640 and 2. 549, the upper band errors were less as compare to previous levels. The lower band also showed a fairly good performance with RMSE of 4. 840 and 4. 574, MSEs of 23. 423 and 20. 920 percent and Margin of Earnings per Share of 3 percent. 100 and 3. The Correlation Matrix between the various Terms and variables was computed, with the following values: 0.263 on the training set and testing set Below is evidence of this: These results suggest that the studied GRU model was capable of capturing the structures within the dataset and, therefore, the model successfully predicted the price and Bollinger Band values alike.

Table 9: GRU (RMSE,MSR,MAE)

Category	Data Type	RMSE	MSE	MAE
Close	Train	12.016	144.387	8.197
	Test	14.310	204.777	10.043
UB	Train	3.819	14.587	2.421
	Test	3.679	13.533	2.405
LB	Train	4.556	20.760	3.104
	Test	4.507	20.317	3.275

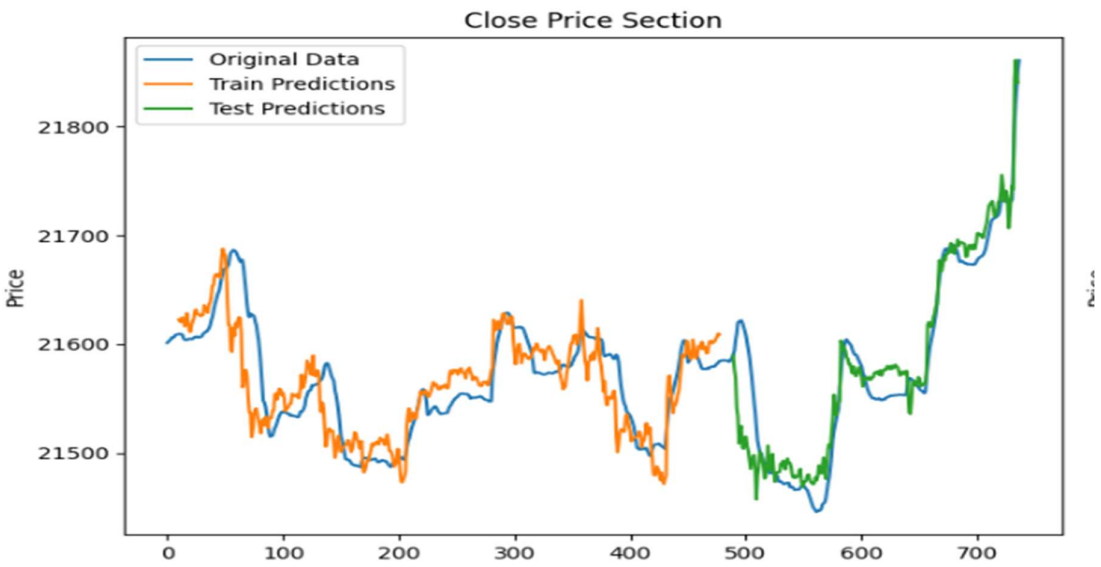

Figure 7: GRU (Training, Testing and Original Data For Close Price)

2 LSTM:

LSTM also yielded accurate results during the training as well as the testing phase in respect of several stock price variables. For the "Close," category During training the model produced error values of 10. The lost students' number is best estimate at about 164 for Mean Squared Error (MSE) or about 6. 158 for Mean Absolute Error (MAE), and 117 respectively. 33 for Coefficient of R-squared (R2), and 164 for Root Mean Square Error (RMSE). undefined defined undefined 926 and Moving Average of Errors (MAE) of 8. Some of the discrepancies might be because of a bit higher error values of the "Close" category which was observed during the testing phase. In testing, these values. to achieve an RMSE of 7 even more often; undefined 940 and MAE of 4. K values: The K values of the upper Bollinger band UB have reached 848 whereas the training value of the K1 and K2 are 4. undefined undefined undefined More specifically the RMSE for their Lower Bollinger Band (LB) category was calculated to be approximately 4 in the training phase. sITS: undefined 971, and MAE of 3. 115, Hence, the testing results of RMSE, MSE and MAE are 5. undefined undefined undefined These measurements suggest that in the training process the LSTM model was more accurate than the testing process with a little disposal of errors when testing.

Table 10: LSTM (RMSE,MSR,MAE)

Category	Data Type	RMSE	MSE	MAE
Close	Train	10.824	117.164	6.982
	Test	13.562	183.924	8.603
UB	Train	4.663	21.746	3.099
	Test	7.612	57.936	4.848
LB	Train	4.356	18.971	3.115
	Test	5.912	34.952	4.075

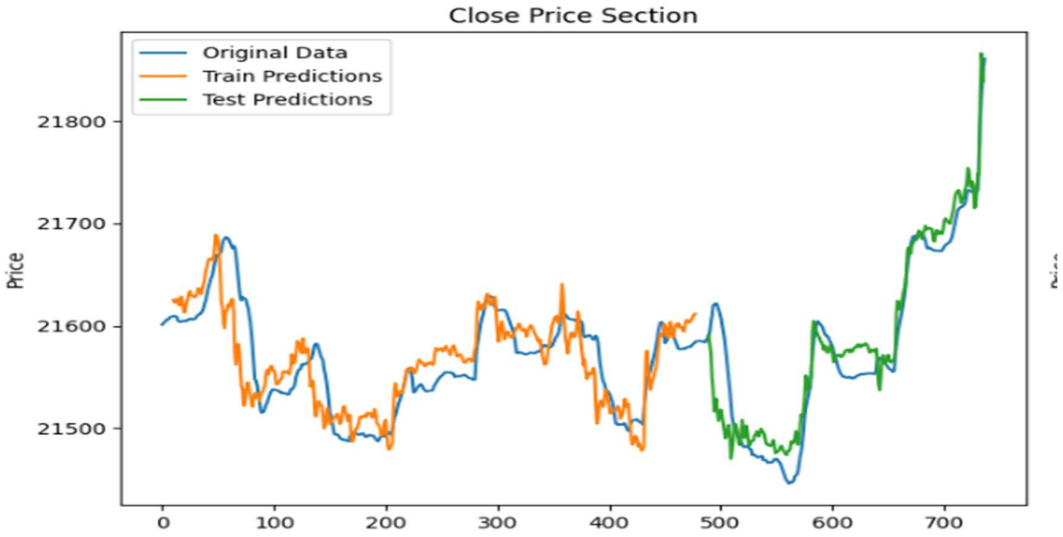

Figure 8: LSTM (Training, Testing and Original Data For Close Price)

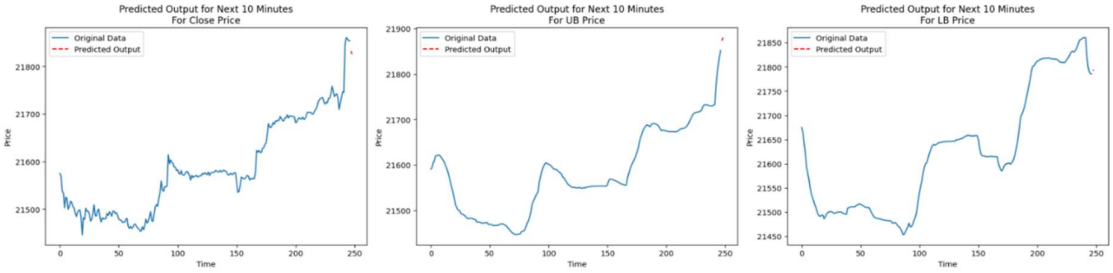

Figure 9: Prediction Point For All Three Price Sections (GRU)

3 GRU + LSTM (Hybrid)

The combination of the GRU and LSTM layers was also evaluated by comparing the accuracy of the stock price indicators during training and testing on the trained model. The overall results of the training of the model for the 'Close' category of the responses produced a Mean-Squared-Error (MSE) of 116. 519, an RMSE of 10. The RMSE of the model is 794 and the Mean-Absolute-Error (MAE) of value 7. 027. The modifications of the Lower Bollinger Band (LB) also librated moderate errors in their training RMSE of 3. 648, MSE of 13. 309 while mean absolute error on atrial fibrillation data was 2. 485, and testing errors of RMSE 5. 016, MSE 25. 156, and MAE 3. 250. From these outcomes, it was noted that the proposed model of combination of GRU and LSTM was effective during the training phase but during the testing phase the accuracy was slightly off especially when predicting the "Close" price.

Table 11: GRU + LSTM (RMSE,MSR,MAE)

Category	Data Type	RMSE	MSE	MAE
Close	Train	10.794	116.519	7.027
	Test	16.020	256.641	10.665
UB	Train	3.216	10.341	2.056
	Test	4.847	23.497	2.895
LB	Train	3.648	13.309	2.485
	Test	5.016	25.156	3.250

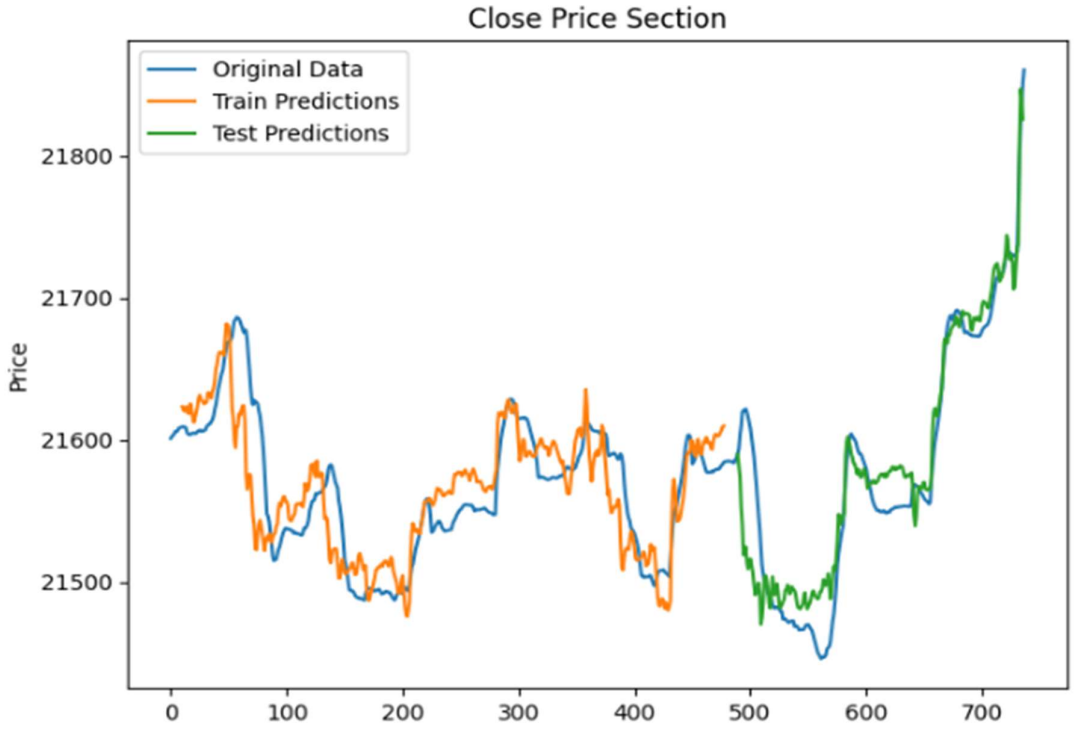

Figure 10: GRU + LSTM (Training, Testing and Original Data For Close Price

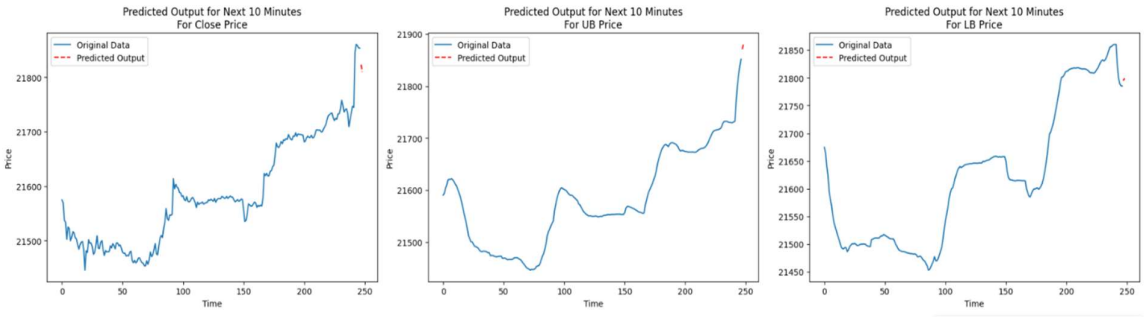

Figure 11: Prediction Point For All Three Price Section (GRU+LSTM)

As a result of developing the hybrid GRU + LSTM model on each of the columns including Close, UB and LB the model is updated every 5 minutes. This means that the model keeps on learning and is always updated with new data so as to predict the possibility of occurrence of the event in the next 10 minutes. All the columns are learned separately so that the model can learn the totally of the trends and values for the given indicator. This often a daily update means that the model remains very sensitive to the current market position hence ideal for intra-day trading decisions.

Table 12: Result Comparison

Actual	LSTM	GRU	LSTM+GRU	Mode OF Prediction	Result
Buy	Buy	Buy	Hold	Buy	Buy
Buy	Hold	Sell	Hold	Hold	Hold
Buy	Buy	Buy	Buy	Buy	Buy
Sell	Buy	Buy	Sell	Buy	X
Buy	Hold	Sell	Sell	Sell	X
Sell	Sell	Sell	Sell	Sell	Sell
Sell	Hold	Hold	Sell	Hold	Hold
Sell	Sell	Sell	Hold	Sell	Sell
Buy	Buy	Buy	Buy	Buy	Buy
Buy	Buy	Buy	Buy	Buy	Buy

The LSTM, GRU and LSTM+GRU models that have been developed for the research purpose portray a strategic concept for decision making in trading environment. Based on the results gathered, the model is efficient in identifying profitable opportunities that it targets considering that it can make correct predictions for actionable trades for 60% of them. This consistencies of wrong predictions is most important in financial markets due to small differential of accuracy that is proportionately varies in big way when the number of trade is high.

Importance of "Hold" Predictions:

The extra 20% of those eighty stocks labelled “Hold” serve a buffer against risk. In this regard, the less trades are made in volatile or domestically turbulent market environment the better for the overall business. This is especially the case when the model gives a “Hold” recommendation, which means the investor does not commit his/her capital to investments that

may result in losses. This adding of safety net makes the model more protective in a way that it will not open the trades unless the signals are clear and consistent.

Impact of Incorrect Predictions:

The 20% of fully incorrect predictions are the price that one must pay for using any crude trading model. In these instances, the predictions lead to decisions that go in the opposite direction of the actual market movements hence incurring a loss. However, the fact that most of these signals are incorrect shows a relatively low frequency of such occurrences meaning that the model in question is more accurate most of the times. The best part of the process is separating the incorrect trades where traders utilizing a strong money management strategy which negates those specific trades where the system is going to be a loser overall but the overall system is going to be profitable.

Strategic Value of the Hybrid Model:

This is made possible by the fact that hybrid models mean making predictions from several models comprising of LSTM, GRU, and LSTM+GRU. When the concept of architecture strengths is used the final prediction is more accurate and contains balanced output. The mode based aggregation method helps to arrive at an aggregate representation from all the models rather than attaining from a single model. This make it easier to generalize and reduce the likelihood of overfitting as various conditions of the market are taken into consideration.

Trading Application and Risk Management:

As far as traders are concerned this 60% accuracy matched with a risk averse "Hold" signal makes the model particularly appropriate for the conservative or long-term approach. It provides an opportunity to remain connected with the market without high risks as to unnecessary activities. This breakdown between Buy/Sell and Hold signals may be useful in reducing confusion and helping traders stick to a strict plan.

Moreover, the application of this model into an algorithmic trading platform can only improve its performance by diminishing the impact of feelings' input. As the model suggests, if the user unwaveringly sticks to its guidance, even with the help of the "Hold" recommendation, which is an integrated safety measure – the returns will be less volatile.

Thus, the integration of the three models shows an effective scenario of trading with a high accuracy of buy and sell signals but at the same time, there are avoidance of risky situations as well. Predicted with a 60% accuracy the actionable positive predictions illustrated the model's potential of identifying good profitable outcomes; the 20 percent 'Hold' potential avails the model's ability to avoid unprofitable risks. While a fair mileage of twenty percent of fully incorrect predictions has its issues, the general performance of the hybrid model exemplifies robust enough to be employed for actual trading purposes With adequate risk control and money management strategies. This mixed structure improves decision-making and helps the traders to be in a better position to effectively handle challenging market situations.

DISCUSSION

We showed that our model can be effective for stock price prediction during its evaluation within the framework. for short term price changes, the forecasting ability is fairly good. However, whenever these predictions are extended as compared to options trading on the long-run, the model had some problems with meeting this point. mainly because of high volatility and the fact of complex pricing for options.

In order to solve these problems, we devoted our efforts to improve the strategy for short-term prediction of prices. To schedule the message for a delivery intended to reach a recipient at a certain time of the day, the time duration to be specified would be 5 min, for instance. In regard to the strategy that was applied, it entailed the use of three different types of messaging. models: The literature review is presented on Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) and a combined. LSTM+GRU model. Such combination was adopted in order to maximize on the strengths each held on the other: shown the efficiency of the mentioned techniques in the short-term movements of prices.

Since options trading are highly unpredictable, timely predictions are inevitable for success. To make sure our models can give results within 5 minute time period, we then adopted a very strict training procedure with a restriction in the population sample. Specifically, the model training a break which had a maximum of five minutes, so that the probability forecasts could be made in good time. This necessitated reducing the data sample from the larger 30 days look into a more detailed 10 days look into meet the fast training needs all the same.

In practice, the models are trained on different machines so as to enhance the process of making predictions. All three models were trained separately with LSTM, GRU, and LSTM+GRU as different models to be tested. For each day there were several forecasts produced from these models and these were combined. The last of these signals headed a global stock trading recommendation—whether to buy or sell, or hold—was calculated by finding the central value— the mode—of the signals produced by the three models. This approach made it possible to optimize the training and execution of the models in the system. required time period so that traders can get alerts at the earliest without any time lag.

In order to obtain such a result, our model concentrated on short-term forecasts and minimized the data samples. achieved a good training speed that is paralleled by the accuracy of the predictions made. This setup offered the traders timely low-risk information which was critical to their business. signals; to strengthen the decision-making capacity and possibly the quality of decisions in an organisation’s profitability.

CONCLUSION

For this purpose, three models have been proposed and tested which are GRU, LSTM, and Hybrid GRU + LSTM to forecast 5-minute interval Nifty-50 index price fluctuation during the next interval. By combining these models through the model-based approach, it was possible to arrive at an average accuracy of 60%, meaning that we were able to forecast the movements of the market 6 out of 10 times. This is shown by the ability of the model to be applied in the real-time trading environment whereby the early indications will assist traders in making proper decisions before the actual change in the prices occurs hence making money out of it.

A foremost advantage of our model is that it contains a "Hold" signal which constitutes 20% of all signals; the component that allows for risk exposure. During volatile market conditions, they help in avoiding unnecessary trading hence keeping capital intact and reducing on possible losses. This conservative approach helps the model eliminate the pushing of trades when the market signals are tangled, which simply places a buffer against making bad decisions.

Therefore, the utilization of both GRU, LSTM and Hybrid GRU + LSTM results brings out a strong and stable trading system. Though it is important to note that participating in the betting system at least 20% of the time will yield fully incorrect predictions, nonetheless, the strategy of the system when combined with good money management skills will always prove to be profitable. The model's capacity to provide stable actionable signals and integrate "Hold" recommendations more makes the model likely for real-trading utilization. Subsequent initiatives will include live prototyping of the model and optimizing the model to enhance its performance in various markets and increase both the effectiveness and efficiency of the model and its results.

REFERENCES

[1] (Islam, & Hossain, 2021). Foreign exchange currency rate prediction using a GRU-LSTM hybrid network. *Soft Computing Letters*, *3*, 100009. https://doi.org/10.1016/j.socl.2020.100009

[2] (Mehtab et al, 2021). Stock price prediction using machine learning and LSTM-Based deep learning models. In Communications in computer and information science (pp. 88–106). https://doi.org/10.1007/978-981-16-0419-5_8

[3] (Jafar et al, 2023). Forecasting of NIFTY 50 Index Price by Using Backward Elimination with an LSTM Model. Journal of Risk and Financial Management, 16(10), 423. https://doi.org/10.3390/jrfm16100423

[4] (Chauhan et al, 2023). Stock price forecasting using PSO hypertuned neural nets and ensembling. Applied Soft Computing, 147, 110835. https://doi.org/10.1016/j.asoc.2023.110835

[5] (Fathali et al, 2022). Stock Market Prediction of NIFTY 50 Index Applying Machine Learning Techniques. Applied Artificial Intelligence, 36(1). https://doi.org/10.1080/08839514.2022.2111134

[6] (Chowdhury et al, 2024). Deep Learning Models for Stock Market Forecasting: A Comprehensive Comparative Analysis. Journal of Business and Management Studies, 6(2), 95–99. https://doi.org/10.32996/jbms.2024.6.2.9

[7] (Ghosh et al, 2024). Deep Learning in Stock Market Forecasting: Comparative analysis of neural network architectures across NSE and NYSE. Journal of Computer Science and Technology Studies, 6(1), 68–75. https://doi.org/10.32996/jcsts.2024.6.1.8

[8] (Manu et al, 2020). Stock index prediction using artificial neural network and econometric model: The case of Nifty 50. *International Journal of Advanced Science and Technology*, 29(5), 3425-3437. https://www.researchgate.net/publication/350037462_Stock_Index_Prediction_using_Artificial_Neural_Network_and_Econometric_Model_The_case_of_Nifty_50

[9] Code/Data Link: https://www.kaggle.com/code/praveenchoudhary1217/stock-option-indicator/edit

Appendix

Create A signals On Original Data

```python
# Load data
file_path = "/kaggle/input/dataset/NIFTY 50.csv"
df1 = pd.read_csv(file_path)

end = '2024-01-10 10:55:00'

# Convert the date column to datetime and floor it to 5 minutes
df1['date'] = pd.to_datetime(df1['date'])
df1['date'] = df1['date'].dt.floor('5min')

# Resample the data to get OHLC values
df_resampled1 = df1.groupby('date').agg({
    'open': 'first',
    'high': 'max',
    'low': 'min',
    'close': 'last'
}).dropna()

df_resampled1 = df_resampled1.reset_index()
df_resampled1 = df_resampled1[(df_resampled1['date'] >= start) & (df_resampled1['date'] <= end)]

# Calculate Moving Averages
df_resampled1['short_ma'] = df_resampled1['close'].rolling(window=5).mean()
df_resampled1['long_ma'] = df_resampled1['close'].rolling(window=20).mean()

# Calculate RSI
window_length = 14
delta = df_resampled1['close'].diff(1)
gain = (delta.where(delta > 0, 0)).rolling(window=window_length).mean()
loss = (-delta.where(delta < 0, 0)).rolling(window=window_length).mean()
rs = gain / loss
df_resampled1['rsi'] = 100 - (100 / (1 + rs))

# Calculate MACD
short_ema = df_resampled1['close'].ewm(span=12, adjust=False).mean()
long_ema = df_resampled1['close'].ewm(span=26, adjust=False).mean()
df_resampled1['macd'] = short_ema - long_ema
df_resampled1['macd_signal'] = df_resampled1['macd'].ewm(span=9, adjust=False).mean()
```

[14]

```python
# Improved Signal Generation Logic
def generate_signal(row):
    # High priority: RSI thresholds for oversold/overbought conditions
    if row['rsi'] < 50:
        return 'sell'
    elif row['rsi'] > 50:
        return 'buy'

    # Medium priority: MACD crossovers for momentum detection
    if row['macd'] > row['macd_signal']:
        return 'buy'
    elif row['macd'] < row['macd_signal']:
        return 'sell'

    # Low priority: Moving average crossovers as a fallback
    if row['short_ma'] > row['long_ma']:
        return 'buy'
    else:
        return 'sell'

df_resampled1['signal'] = df_resampled1.apply(generate_signal, axis=1)

# Count the "buy" and "sell" signals only
signal_counts = df_resampled1['signal'].value_counts()
print(signal_counts)
```

[14] Python

```
signal
buy     275
sell    271
Name: count, dtype: int64
```

Create Bolinger Band

```python
import pandas as pd
# Define the window size for rolling calculations
window = 20  # Example window size, adjust as needed

# Calculate the moving average (Middle Band)
df_resampled['Middle Band'] = df_resampled['close'].rolling(window=window).mean()

# Calculate the rolling standard deviation
df_resampled['STD'] = df_resampled['close'].rolling(window=window).std()

# Calculate the Upper Band
df_resampled['Upper Band'] = df_resampled['Middle Band'] + (df_resampled['STD'] * 2)

# Calculate the Lower Band
df_resampled['Lower Band'] = df_resampled['Middle Band'] - (df_resampled['STD'] * 2)


```

[21]

```
df1=df_resampled.reset_index()['close']
df7=df_resampled.reset_index()['Upper Band']
df8=df_resampled.reset_index()['Lower Band']
```

[23]

```
# deleting date column and normalizing using MinMax Scaler
from sklearn.preprocessing import MinMaxScaler
del df_resampled['date']
scaler=MinMaxScaler(feature_range=(0,1))
closedf=scaler.fit_transform(np.array(df_resampled).reshape(-1,1))
print(closedf.shape)
```

[24]

```
(4200, 1)
```

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
df1=scaler.fit_transform(np.array(df1).reshape(-1,1))
df7=scaler.fit_transform(np.array(df7).reshape(-1,1))
df8=scaler.fit_transform(np.array(df8).reshape(-1,1))
```

[25]

```
##splitting dataset into train and test split
training_percentage = 0.65
training_size=int(len(df1)*training_percentage)
test_size=len(df1)-training_size
train_data_df1,test_data_df1=df1[0:training_size,:],df1[training_size:len(df1),:1]
train_data_df7,test_data_df7=df7[0:training_size,:],df7[training_size:len(df7),:1]
train_data_df8,test_data_df8=df8[0:training_size,:],df8[training_size:len(df8),:1]
```

26]

```
import numpy
# convert an array of values into a dataset matrix
def create_dataset(dataset, time_step=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-time_step-1):
        a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return numpy.array(dataX), numpy.array(dataY)
```

27]

```
# reshape into X=t,t+1,t+2,t+3 and Y=t+4
time_step = 10
X_train_df1, y_train_df1 = create_dataset(train_data_df1, time_step)
X_test_df1, y_test_df1 = create_dataset(test_data_df1, time_step)
X_train_df7, y_train_df7 = create_dataset(train_data_df7, time_step)
X_test_df7, y_test_df7 = create_dataset(test_data_df7, time_step)
X_train_df8, y_train_df8 = create_dataset(train_data_df8, time_step)
X_test_df8, y_test_df8 = create_dataset(test_data_df8, time_step)
```

28]

+ Code + Markdown

```
# reshape input to be [samples, time steps, features] which is required for LSTM
X_train_df1 =X_train_df1.reshape(X_train_df1.shape[0],X_train_df1.shape[1] , 1)
X_test_df1 = X_test_df1.reshape(X_test_df1.shape[0],X_test_df1.shape[1] , 1)
X_train_df7 =X_train_df7.reshape(X_train_df7.shape[0],X_train_df7.shape[1] , 1)
X_test_df7 = X_test_df7.reshape(X_test_df7.shape[0],X_test_df7.shape[1] , 1)
X_train_df8 =X_train_df8.reshape(X_train_df8.shape[0],X_train_df8.shape[1] , 1)
X_test_df8 = X_test_df8.reshape(X_test_df8.shape[0],X_test_df8.shape[1] , 1)
```

29]

LSTM, GRU and Hybrid (LSTM+GRU) MODEL

```
import time

# Start the timer
start_time = time.time()

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GRU, Dropout, Dense
from tensorflow.keras.callbacks import EarlyStopping

# Function to create and train the GRU model
def create_and_train_model(X_train, y_train, X_test, y_test, batch_size):
    # EarlyStopping Callback
    callback = EarlyStopping(
        monitor="val_loss",
        min_delta=0.000001,
        patience=10,
        verbose=1,
        mode="auto",
        restore_best_weights=True
    )

    model = Sequential()
    model.add(GRU(256, input_shape=(time_step, 1), return_sequences=True))
    model.add(Dropout(0.1))
    model.add(GRU(128))
    model.add(Dropout(0.1))
    model.add(Dense(32, activation='relu'))
    model.add(Dense(1, activation='linear'))  # Use 'relu' if needed
    model.compile(optimizer='adam', loss='mse', metrics=['mse', 'mae'])
```

```
#     model = Sequential()
#     model.add(GRU(128, input_shape=(time_step,1), return_sequences=True))
#     model.add(Dropout(0.1))
#     model.add(LSTM(128))
#     model.add(Dropout(0.1))
#     model.add(Dense(64, activation='relu'))
#     model.add(Dense(1, activation='linear'))
#     model.compile(optimizer='adam', loss='mse', metrics=['mse','mae'])

#     model = Sequential()
#     model.add(LSTM(256, input_shape=(time_step, 1), return_sequences=True))
#     model.add(Dropout(0.1))
#     model.add(LSTM(128))
#     model.add(Dropout(0.1))
#     model.add(Dense(64, activation='relu'))
#     model.add(Dense(1, activation='linear'))  # Use 'relu' if needed
#     model.compile(optimizer='adam', loss='mse', metrics=['mse', 'mae'])

    history = model.fit(X_train, y_train, epochs=3000, batch_size=batch_size,
                        validation_data=(X_test, y_test), callbacks=[callback])
    return model, history

# Train model1
model1, history_1 = create_and_train_model(X_train_df1, y_train_df1, X_test_df1, y_test_df1, batch_size=64)

# Train model7
model7, history_7 = create_and_train_model(X_train_df7, y_train_df7, X_test_df7, y_test_df7, batch_size=64)

# Train model_8
model8, history_8 = create_and_train_model(X_train_df8, y_train_df8, X_test_df8, y_test_df8, batch_size=64)
```

[43]

```
Epoch 1/200
6/6 ━━━━━━━━━━━━━━━━━━━━ 2s 65ms/step - loss: 0.1424 - mae: 0.3144 - mse: 0.1424 - val_loss: 0.0255 - val_mae: 0.1451 - val_mse: 0.0255
Epoch 2/200
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0149 - mae: 0.0997 - mse: 0.0149 - val_loss: 0.0100 - val_mae: 0.0800 - val_mse: 0.0100
Epoch 3/200
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0090 - mae: 0.0750 - mse: 0.0090 - val_loss: 0.0126 - val_mae: 0.0932 - val_mse: 0.0126
Epoch 4/200
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0071 - mae: 0.0618 - mse: 0.0071 - val_loss: 0.0078 - val_mae: 0.0689 - val_mse: 0.0078
```

Signal Predition

```
import numpy as np
def predict_future(test_data, time_step, n_steps, model, time_interval, total_time):
    lst_output = []
    i = 0

    x_input = test_data[len(test_data) - time_step:].reshape(1, -1)
    temp_input = list(x_input[0])

    while i < total_time:
        if len(temp_input) > time_step:
            x_input = np.array(temp_input[1:])
            x_input = x_input.reshape((1, n_steps, 1))

            yhat = model.predict(x_input, verbose=0)
            temp_input.extend(yhat[0].tolist())
            temp_input = temp_input[1:]

            lst_output.extend(yhat.tolist())
        else:
            x_input = x_input.reshape((1, n_steps, 1))
            yhat = model.predict(x_input, verbose=0)
            temp_input.extend(yhat[0].tolist())

            lst_output.extend(yhat.tolist())

        i += time_interval  # Increment by the specified time interval

    return lst_output

def make_predictions_and_recommendation(model1, model7, model8, X_train, y_train, X_test, y_test, scaler, test_data, time_step, n_steps, time_interval, total_time):
    # Make predictions
    train_predict_df1 = model1.predict(X_train)
    test_predict_df1 = model1.predict(X_test)
    train_predict_df7 = model7.predict(X_train)
    test_predict_df7 = model7.predict(X_test)
    train_predict_df8 = model8.predict(X_train)
    test_predict_df8 = model8.predict(X_test)
```

[44]

```
    # Transform back to original form
    train_predict_df1 = scaler.inverse_transform(train_predict_df1)
    test_predict_df1 = scaler.inverse_transform(test_predict_df1)
    original_ytrain_df1 = scaler.inverse_transform(y_train_df1.reshape(-1,1))
    original_ytest_df1 = scaler.inverse_transform(y_test_df1.reshape(-1,1))

    train_predict_df7 = scaler.inverse_transform(train_predict_df7)
    test_predict_df7 = scaler.inverse_transform(test_predict_df7)
    original_ytrain_df7 = scaler.inverse_transform(y_train_df7.reshape(-1,1))
    original_ytest_df7 = scaler.inverse_transform(y_test_df7.reshape(-1,1))

    train_predict_df8 = scaler.inverse_transform(train_predict_df8)
    test_predict_df8 = scaler.inverse_transform(test_predict_df8)
    original_ytrain_df8 = scaler.inverse_transform(y_train_df8.reshape(-1,1))
    original_ytest_df8 = scaler.inverse_transform(y_test_df8.reshape(-1,1))


    # Predict future values for different datasets
    output_df1 = predict_future(test_data_df1, time_step, n_steps, model1, time_interval, total_time)
    output_df7 = predict_future(test_data_df7, time_step, n_steps, model7, time_interval, total_time)
    output_df8 = predict_future(test_data_df8, time_step, n_steps, model8, time_interval, total_time)

    # Transform future predictions back to original scale
    close = scaler.inverse_transform(np.array(output_df1).reshape(-1, 1))
    ub = scaler.inverse_transform(np.array(output_df7).reshape(-1, 1))
    lb = scaler.inverse_transform(np.array(output_df8).reshape(-1, 1))

    # Generate buy/sell/hold recommendation
    buy = 0
    sell = 0
    hold = 0

    if np.mean(train_predict_df1[-20:]) <= close[1]:
        recommendation = 'Buy'
        buy = 1
    elif np.mean(train_predict_df1[-20:]) > close[1] and np.mean(train_predict_df1[-20:]) < close[1]:
        recommendation = 'Hold'
        hold = 1
    else:
        recommendation = 'Sell'
        sell = 1
```

```
    print(f"Recommendation: {recommendation}")
    return recommendation, buy, sell, hold

# Assuming model and test_data_df1, test_data_df2 are defined elsewhere
time_step = 10
n_steps = time_step
time_interval = 5  # User-specified time interval in minutes
num_intervals = 2  # Number of predictions you want
# Total time for predictions
total_time = num_intervals * time_interval

# Example usage of the function
recommendation, buy, sell, hold = make_predictions_and_recommendation(
    model1, model7, model8,
    X_train_df1, y_train_df1, X_test_df1, y_test_df1,
    scaler, test_data_df1, time_step, n_steps,
    time_interval, total_time
)
print(f"Buy: {buy}, Sell: {sell}, Hold: {hold}")
# End the timer
end_time = time.time()
# Calculate the time taken in seconds
time_taken = end_time - start_time
# Convert time taken to minutes and seconds
minutes = int(time_taken // 60)
seconds = time_taken % 60
# Output the result and time taken
print(f"Time taken: {minutes}:{seconds:.2f} (min:sec)")
```

[44]

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Recommendation: Sell
Buy: 0, Sell: 1, Hold: 0
Time taken: 2:46.97 (min:sec)
```

Proforma

Undertaking from the PG student while submitting his/her final dissertation to his respective institute

Ref. No. 23070243065

I , the following student

Sr. No.	Sequence of students names on a dissertation	Students name	Name of the Institute & Place	Email & Mobile
1.	First Author	Praveen Choudhary	SIG	Email: 23070243065@sig.ac.in Mobile: 8619714798

Note: Put additional rows in case of more number of students

hereby give an undertaking that the dissertation **Enhancing Option Trading Predictions on Nifty-50 Using Ensemble Technique** been checked for its Similarity Index/Plagiarism through Turnitin software tool; and that the document has been prepared by me and it is my original work and free of any plagiarism. It was found that:

1.	The Similarity Index (SI) was: *(Note: SI range: 0 to 10%; if SI is >10%, then authors cannot communicate ms;* ***attachment of SI report is mandatory)***	8%
2.	The ethical clearance for research work conducted obtained from: *(Note: Name the consent obtaining body; if 'not appliable' then write so)*	NA
3.	The source of funding for research was: *(Note: Name the funding agency; or write 'self' if no funding source is involved)*	Self
4.	Conflict of interest: *(Note: Tick √ whichever is applicable)*	No
5.	The material (adopted text, tables, figures, graphs, etc.) as has been obtained from other sources, has been duly acknowledged in the manuscript: *(Note: Tick √ whichever is applicable)*	Yes

In case if any of the above-furnished information is found false at any point in time, then the University authorities can take action as deemed fit against all of us.

Full Name &
Signature of the student

Name &
Signature of SIU Guide/Mentor

Date: 13 September 2024

Endorsement by
Academic Integrity Committee (AIC)

Place: Pune

Note: It is mandatory that the Similarity Index report of plagiarism (only first page) should be appended to the UG/PG dissertation.

Turnitin Originality Report

Processed on: 04-Sep-2024 16:10 IST
ID: 2438315078
Word Count: 8810
Submitted: 3

V04 By Praveen Choudhary

Similarity Index	Similarity by Source
8%	Internet Sources: 6% Publications: 6% Student Papers: 3%

1% match (Himanshu Goel, Bhupender Kumar Som. "Stock market prediction, COVID-19 pandemic and neuralnetworks: an SCG algorithm application", EconomiA, 2023)
Himanshu Goel, Bhupender Kumar Som. "Stock market prediction, COVID-19 pandemic and neuralnetworks: an SCG algorithm application", EconomiA, 2023

1% match (Internet from 20-Mar-2024)
https://www.al-kindipublisher.com/index.php/jcsts/article/view/6643

< 1% match (Internet from 09-May-2023)
https://www.researchgate.net/profile/Zahra-Kodia-Aouina/publication/363632317_Stock_Market_Prediction_of_NIFTY_50_Index_Applying_Machine_Learning_Techniques/links/6368fd752f4bca7fd03d6bcc/Sto
Market-Prediction-of-NIFTY-50-Index-Applying-Machine-Learning-Techniques.pdf?origin=publication_detail

< 1% match (Internet from 17-Mar-2023)
https://www.researchgate.net/publication/352803474_The_Deep_Learning_LSTM_and_MTD_Models_Best_Predict_Acute_Respiratory_Infection_among_Und
Five-Year_Old_Children_in_Somaliland

< 1% match (Internet from 06-Feb-2023)
https://www.researchgate.net/publication/340289689_Machine_Learning_Approach_for_Confirmation_of_COVID-19_Cases_Positive_Negative_Death_and_Release

< 1% match (Internet from 04-Feb-2023)
https://www.researchgate.net/publication/227359841_Modelling_the_asymmetry_of_stock_market_volatility

< 1% match (Internet from 18-Sep-2021)
https://www.researchgate.net/publication/349066693_Stock_Price_Prediction_Using_Machine_Learning_and_LSTM-Based_Deep_Learning_Models

< 1% match (student papers from 14-Jun-2024)
Submitted to University of Hertfordshire on 2024-06-14

< 1% match (student papers from 09-Mar-2024)
Submitted to University of East London on 2024-03-09

< 1% match (Internet from 17-Jun-2024)
https://ojs.amhinternational.com/index.php/imbr/issue/download/313/16%282%29SI

< 1% match (student papers from 18-Mar-2024)
Submitted to Bahrain Polytechnic on 2024-03-18

< 1% match (student papers from 17-May-2024)
Submitted to Manchester Metropolitan University on 2024-05-17

< 1% match (Internet from 02-Oct-2021)
http://www.sersc.org/journals/index.php/IJAST/citationstylelanguage/get/apa?submissionId=12034

< 1% match (student papers from 16-May-2024)
Submitted to Universidad de Monterrey on 2024-05-16

< 1% match (Internet from 28-Apr-2023)
https://ncr.christuniversity.in/academic-publication/main%20campus/school%20of%20humanities%20and%20social%20sciences/MA==

< 1% match (student papers from 29-Mar-2022)
Submitted to University of Leeds on 2022-03-29

< 1% match (publications)
Sujata Dash, Subhendu Kumar Pani, Joel J. P. C. Rodrigues, Babita Majhi. "Deep Learning, Machine Learning and IoT in Biomedical and Health Informatics - Techniques and Applications", CRC Press, 2022

< 1% match (publications)
Aakanksha Sharaff, G. R. Sinha. "Data Science and Its Applications", CRC Press, 2021

< 1% match (Vintha, Nikhila. "Comparative Analysis of Deep Learning Approaches for Analysis and Prediction of Multivariate Time Series Data", The University of Toledo, 2024)
Vintha, Nikhila. "Comparative Analysis of Deep Learning Approaches for Analysis and Prediction of Multivariate Time Series Data", The University of Toledo, 2024

< 1% match (Internet from 15-Dec-2022)
https://assets.researchsquare.com/files/rs-27216/v1/0caf8163-8e41-4516-aacf-809a95fef9b4.pdf?c=1631841476

< 1% match (student papers from 02-Sep-2024)
Submitted to Cardiff University on 2024-09-02

< 1% match (student papers from 29-Jun-2023)
Submitted to Nelson Marlborough Institute of Technology on 2023-06-29

< 1% match (Internet from 19-Aug-2022)