SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Transformer Based Nifty-50 Stock Index Close Price Prediction

Krishnapriya C A[1], Ajay James[2], Asha J[3], Manju B[4]

[1]M Tech Student, Department of CSE, Government Engineering College Thrissur, India. E-mail: krishnapriyaca@gmail.com
[2]Associate Professor, Department of CSE, Government Engineering College, Idukki, India. E-mail: ajay@gectcr.ac.in
[3]Associate Professor, Department of ECE, Government Engineering College, Thrissur, India. E-mail: ashaj@gectcr.ac.in
[4]Professor, Department of EEE, Government Engineering College, Thrissur, India. E-mail: manjub@gectcr.ac.in

ABSTRACT

Stock market investment has emerged as a high-growth financial opportunity in the modern economy. The inherently volatile and non-linear nature of equity markets, however, introduces substantial financial risk, making accurate price prediction a challenging yet critically important task. The rapid advancement of artificial intelligence (AI), coupled with the exponential growth in financial data availability and improved computational capabilities, has enabled new frontiers for developing robust and efficient stock price prediction models.

The Nifty 50 index, comprising the fifty most liquid and actively traded stocks listed on the National Stock Exchange (NSE) of India, serves as the benchmark indicator of the Indian equity market and a key barometer of the nation's macroeconomic health. Accurate forecasting of the Nifty 50 closing price, therefore, holds significant value for investors, financial analysts, and policymakers alike.

This study proposes a Multivariate Transformer-based deep learning model for one-step-ahead prediction of the Nifty 50 stock index closing price Nifty 50 stock index historical price data is used as the dataset. Input parameters contain historical prices and technical parameters. To validate the superiority of the proposed approach, its performance is benchmarked against a well-established Long Short-Term Memory (LSTM)-based model. Additionally, an ablation study is conducted to assess the model's predictive efficiency when the input feature set is restricted solely to the four primary price parameters — Open, High, Low, and Close (OHLC). Model performance is evaluated using three standard regression metrics: Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and the Coefficient of Determination (R^2). Experimental results demonstrate that the proposed Multivariate Transformer model significantly outperforms the LSTM-based baseline across all evaluation metrics, establishing its effectiveness as a reliable tool for stock index forecasting.

Keywords: Nifty 50, Stock Price Prediction, Close Price, Transformer, Lstm

Introduction

Prediction is among the most challenging subjects in research. When forecasts prove accurate, they have the potential to simplify decision-making and improve people's lives significantly. Forecasting the stock market is particularly difficult due to its non-linear behaviour, enormous volume of transactional data, and highly dynamic nature. Local and global events further compound this complexity by exerting considerable influence on stock prices. If reliable stock market predictions can be made, regulators and investors stand to benefit greatly, as such predictions enable the formulation of better trading strategies, reduction of investment risk, and enhancement of overall profitability.

Stock market prediction broadly encompasses four categories: stock trend prediction, stock price prediction, stock index prediction, and stock crisis prediction. Stock trend prediction involves forecasting whether the share price of a company will rise or fall in the future. Stock price prediction, on the other hand, refers to estimating the future price of an individual company's stock. Similarly, stock index prediction focuses on forecasting the future values of broader market indices, such as the Nifty 50. Finally, stock crisis prediction pertains to anticipating sharp market downturns — specifically, situations where stock prices decline by more than 10% within a short period due to a sudden surge in selling activity. Such predictions are particularly valuable as they aid investors in timing their market exit strategically. The following section presents a few typical applications of stock price forecasting. Investment Decision Making: Stock price prediction assists investors and traders in making informed decisions regarding the purchase, sale, or retention of stocks. By forecasting future stock values, investors can identify potential market opportunities and assess associated risks in a timely manner.

Portfolio Management: Price prediction plays a pivotal role in building and managing investment

portfolios. By identifying stocks with higher predicted returns or lower volatility, it facilitates effective asset allocation, portfolio diversification, and risk management, thereby enabling investors to optimise their overall portfolio performance.
Risk Assessment: Stock price predictions contribute significantly to evaluating the risk associated with individual stocks or an entire portfolio. By understanding anticipated price volatility, investors can assess the risk-return trade-off more accurately and adjust their investment strategies accordingly to align with their financial objectives.

Algorithmic Trading: Stock price forecasting constitutes a critical component of algorithmic trading, wherein computer algorithms execute trades automatically in accordance with pre-defined rules and conditions. Predictive models are integrated into these algorithms to support real-time decision-making and enable swift, data-driven responses to evolving market conditions. In olden days, the share market prediction had been done mainly through fundamental and technical analysis. Fundamental analysis [1][2] is mainly done for long term investments. It uses company's past performance and economic and financial information of the company for the prediction. Technical analysis is done based on the stock prices using candlestick charts etc. But, now a days due to emergence of artificial intelligence many technological methods are introduced [3][4]. These methods use machine learning, deep learning and sentimental analysis techniques. Time series analysis is yet other techniques used for the stock price forecasting.
Introduction of the artificial intelligence methods improved the accuracy of prediction. When statistical techniques and machine learning methods are compared machine learning models found to be more accurate [5]. On comparing the effectiveness of machine learning and deep learning technologies for stock price forecasting, deep learning methods found to have better prediction accuracy [6]. Among the deep learning models like Recurrent Neural Network (RNN) and LSTM, LSTM is found to have better prediction since RNN has vanishing gradient problem. Even if the aforementioned strategies produce superior outcomes, stock data can occasionally be exceedingly stochastic and erratic. Therefore, their forecasts remain challenging. Nifty 50 is one of the stock indices in India. The Nifty 50 represents the performance of the 50 biggest and most liquid stocks listed on the National Stock Exchange (NSE). It serves as a benchmark for investment performance and is one of the most extensively used stock indices in India. This work aims to implement a deep learning transformer-based model for predicting Nifty 50 stock index close price of N+1th day using N days stock data. This model's error rate is compared to that of an LSTM-based model to demonstrate how well it performs in comparison to other models.
There are different methods ranging from statistical techniques to more advanced deep learning techniques to perform stock market predictions. Currently, deep learning-based hybrid models is mainly used for the prediction because of its improved performance [19]. Some of these techniques are discussed below.

1.2 Stock Prediction Using Multivariate Models

The methods discussed above are univariate models, wherein only a single parameter is used as the input variable. However, relying on a single variable is insufficient, given that stock market data is highly unpredictable, dynamic, and intricately dependent on broader economic conditions. Consequently, multivariate analysis is expected to yield superior results compared to univariate analysis. To validate this, a CNN model was applied to both univariate and multivariate inputs for stock price prediction [20]. In the univariate setting, only the closing price was used as the input variable, whereas the multivariate setting incorporated Open, High, Low, and Close prices. The findings confirmed that multivariate inputs are preferable to univariate inputs for stock price prediction. Additionally, multistep prediction was found to reduce processing complexity.
It is worth noting that conventional decomposition methods are not directly applicable to multivariate stock data. To address this, Multivariate Empirical Mode Decomposition (MEMD) can be employed to decompose stock data containing multiple variables. Multistep forward prediction can further be accomplished using the Multiple Input Multiple Output (MIMO) forecasting strategy. Adopting this approach, the MEMD-LSTM model [12] predicts the closing price of a stock index over multiple days using daily stock price data with input variables comprising Open, High, Low, Close, and Volume values.
Decomposed sub-sequences may, however, exhibit instability and noise. To address this limitation, secondary decomposition is applied to extract additional non-linear features and improve noise filtering [13]. The decomposition techniques employed include Improved Variational Mode Decomposition (IVMD) and Improved Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (ICEEMDAN), both of which terminate decomposition upon the generation of similar signals. In this framework, IVMD is first applied to decompose the initial input. An attention layer is subsequently incorporated into the LSTM to amplify the weights assigned to critical information during prediction. Low-frequency Intrinsic Mode Functions (IMFs) derived from the decomposed output are then fed into an Attention-based LSTM (ALSTM), while ICEEMDAN is used to aggregate and further decompose the high-frequency IMFs.
Only those IMFs that pass both the F-test and Mutual Information (MI) test are forwarded to the ALSTM. Furthermore, the incorporation of multi-factor analysis alongside basic stock price data can enhance

prediction accuracy. In this multi-factor analysis, additional variables are screened using the F-test and MI test, and those that pass are encoded using an autoencoder. These encoded variables, in combination with the selected IMFs, are then utilised for prediction via the ALSTM. The accuracy of this hybrid model was found to be superior to that of univariate models.
The CNN-BiLSTM-AM model [14] predicts the one-step-ahead closing price of a stock using input variables comprising Open, High, Low, Close, Volume, Turnover, Ups and Downs, and Change. CNN is employed for feature extraction, while Bidirectional LSTM (BiLSTM) is used for forecasting. The incorporation of an attention mechanism further enhances prediction accuracy. This model demonstrates superior performance compared to both the CNN-LSTM and CNN-BiLSTM models.

1.3 Stock Prediction Using Transformer

The Transformer model was originally introduced for Natural Language Processing (NLP) tasks and has since been successfully adapted for time series prediction. The Deep Transformer Model [15] predicts the closing price of the following trading day using daily stock index closing price data. The model employs a positional encoding and embedding layer, a multi-head self-attention mechanism, and an encoder-decoder architecture, making its design largely consistent with that of the original Transformer model [16]. The ability of Transformers to capture long-range dependencies is largely governed by the number of attention heads employed [17]. Time2Vec has also been proposed as an effective alternative for positional encoding [18]. Multiple attention heads enable the model to capture critical temporal information while simultaneously filtering out irrelevant background noise, and this multi-head attention mechanism is widely regarded as the primary driver of the Transformer's exceptional performance.
The Transformer architecture consists of N stacked encoder and decoder layers. The encoder compresses relevant information from the input sequence into a fixed-length vector representation, which the decoder subsequently transforms into the desired output. This encoder-decoder architecture provides an effective solution for handling lengthy sequential data. Each encoder layer comprises two sub-layers: a multi-head self-attention layer and a fully connected feed-forward neural network. Residual connections and layer normalisation are applied after each sub-layer to enhance training stability and overall performance. The decoder follows a similar structure, with the key distinction being the presence of two multi-head self-attention layers. Since all inputs to the decoder consist of observed historical data without any future information, the masked attention technique employed in the original decoder is not applied here. This univariate Transformer model has been shown to outperform CNN, RNN, and LSTM-based models.

1.4 Nifty 50 Stock Index Prediction

The majority of the methods discussed above were evaluated on datasets derived from foreign stock indices. In the Indian context, the Nifty 50 is one of the most prominent and widely followed stock exchanges. However, relatively few studies have been conducted specifically on the Nifty 50 index dataset. One such study compared the predictive performance of eight machine learning models, namely Linear Regression, Artificial Neural Network (ANN), Support Vector Machine (SVM), Stochastic Gradient Descent (SGD), K-Nearest Neighbour (KNN), AdaBoost, Decision Tree, and Random Forest [21]. Among these, Linear Regression and ANN demonstrated the best performance. Another study predicted the Nifty 50 price using CNN, RNN, and LSTM models [22], with LSTM yielding the most accurate predictions. This study also demonstrated that the input combination of Open, Low, High, and Close prices results in lower prediction errors. A subsequent comparison between LSTM and Linear Regression on the Nifty 50 dataset confirmed that LSTM delivers significantly more accurate predictions [23], with Linear Regression performing poorly. As this study utilised only the closing price as input, a further investigation compared LSTM performance across different input parameter sets [24] — one comprising only global indices, another consisting solely of technical parameters, and a third combining both. The LSTM model trained exclusively on technical parameters produced the most accurate results.
Despite these efforts, very few studies have applied deep learning techniques to the Nifty 50 dataset, and the majority of existing research has relied on univariate models. Multivariate models, however, have consistently demonstrated superior predictive performance. Furthermore, while Transformer-based models have been shown to outperform LSTM, no prior study has applied a Transformer-based model to the Nifty 50 dataset. To address this gap, the present work aims to predict the next day's closing price of the Nifty 50 index using a multivariate Transformer-based model. A comprehensive comparative study on the Nifty 50 dataset evaluated a wide range of architectures including Linear Regression, LSTM, GRU, CNN, RNN, Temporal Convolutional Network (TCN), and hybrid combinations such as LSTM+GRU, CNN+RNN, CNN+TCN, and LSTM+TCN. The results showed that deep learning models such as LSTM, GRU, CNN, RNN, and TCN exhibited superior performance over linear regression in capturing the nonlinear and time-varying nature of stock market data, with hybrid architectures demonstrating further improvements by leveraging the complementary strengths of individual models. [25].
More recently, ANN-based models were applied to Nifty 50 forecasting with technical indicators as inputs. Daily Nifty 50 data from 2020 to 2024 was used to compute technical indicators such as RSI, Exponential

Moving Average (EMA), and MACD as input features for a feedforward ANN trained using backpropagation for short-term return forecasting. [26].
Despite these advances, transformer-based models — which have demonstrated superior performance over LSTM in general time series forecasting — have not yet been applied to the Nifty 50 dataset. Furthermore, most existing studies rely on univariate models, despite multivariate models consistently showing better predictive performance. To address this gap, the present work proposes a multivariate Transformer-based model for one-step-ahead prediction of the Nifty 50 closing price.

Research Method

This work mainly aims to predict Tth day's Close price of Nifty 50 index using T-M days Nifty 50 stock data using multivariate transformer-based model. M is the window size. The model is tested and trained using the siding window method. The dataset consists of historical prices and technical indicators. Maximum 21 input parameters are used. Figure 1 depicts the model's overall architecture. N layers of encoder and decoder modules make up a transformer. First, linear feed forward layer processes the input of encoder and decoder before it is sent to the transformer encoder and decoder. The encoder module receives the linear feed forward layer's output. A feed forward network and a multi-head attention layer make up the encoder module. Two multi-head attention layers and a feed forward network make up the decoder. The first multi-head attention layer in the decoder module receives its input from the output of the linear feed forward layer on the decoder side. The second multi-head attention layer, which is followed by the feed forward network, receives the output of the first multi-head attention layer as well as the output from the stacked encoder layers. Addition and normalisation operations are carried out in between every layer of encoder and decoder modules. The last decoder module's output is routed through a linear feed forward layer to obtain the close price for Tth day [19]. Figure 2 shows the work flow. This section describes the data collection process, data preprocessing techniques and the layers used in the proposed model.

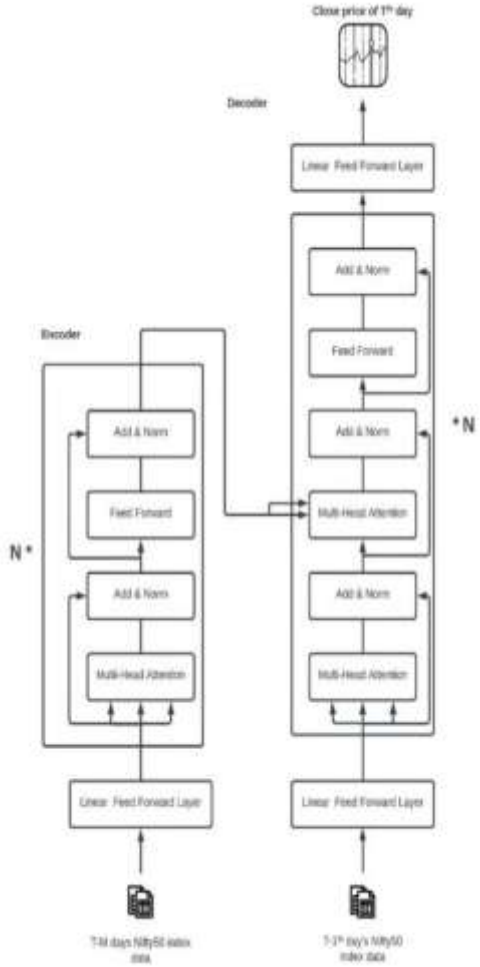

Figure 1: Design of the suggested system

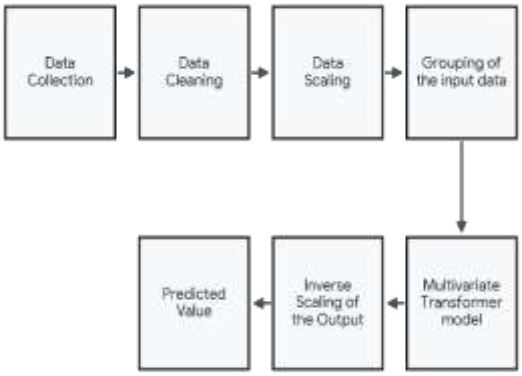

Figure 2: Overall workflow

Data Collection

The Nifty 50 stock price dataset is utilised as the primary data source for this study. A total of 21 input parameters are considered, all of which are listed in Table 1 along with their respective collection sources. The historical price values of the Nifty 50 index — namely Open, High, Low, and Close — are collected using the yfinance API from Yahoo Finance. Four additional derived features are subsequently computed from these historical price values: Daily Return, Cumulative Daily Return, the Difference between High and Low prices, and the Difference between Close and Open prices. The remaining input parameters are computed using the TA-Lib API, a widely used technical analysis library. As part of the data preprocessing and cleaning process, all rows containing NULL values are removed to ensure data integrity and model reliability.

The datasets considered for the experiments are presented in Table 2, along with their respective purposes and associated input parameters. A sample screenshot of the dataset is provided in Figure 3. Detailed explanations of each input variable are presented in the subsequent subsections.

Table 1: Input Parameters Description

Variable	Source
Open	yfinance
Close	yfinance
High	yfinance
Low	yfinance
Daily Return	Derived
Cumulative Daily Return	Derived
High-Low	Derived
Close-Open	Derived
MA with 7 days period	TA-Lib
MA with 21 days period	TA-Lib
SMA with 10 days period	TA-Lib
SMA with 50 days period	TA-Lib
SMA with 500 days period	TA-Lib
EMA with 12 days period	TA-Lib
EMA with 26 days period	TA-Lib
EMA with 64 days period	TA-Lib
RSI with 14 days period	TA-Lib
MACD	TA-Lib
BB High	TA-Lib
BB Low	TTA-Lib

Momentum with 4 days period	TA-Lib

Open: Price at which Nifty 50 started the trading on that particular day.
Close: Last traded price before the market close on that particular day.
High: Highest price at which Nifty 50 traded during a specific trading day.
Low: Lowest price at which Nifty 50 traded during a specific trading day.
Daily Return: It calculates the percentage change in the stock's price from one trading day to the next. It measures the gain or loss the stock has incurred in a particular day.

$$\text{Daily Return} = \frac{T-Y}{T} \qquad (1)$$

T is the closing price of today. Y is the closing price of yesterday.

Cumulative Daily Return: It indicates the investment's overall return over a given time frame, accounting for daily returns over that time frame. Multiply the daily returns for each trading day consecutively in order to determine the cumulative daily return. The equation is as follows:

$$CDR = (1 + DR_1) * (1 + DR_2) * \ldots * (1 + DR_n) \qquad (2)$$

In this formula, CDR represents Cumulative Daily Return, DR_1 represents the first trading day's daily return, DR_2 represents the second trading day's daily return, and so on, up to DR_n, which represents the nth trading day's daily return.
High – Low: Price difference between high and low.

Table 2: Datasets

Dataset	Date Range	Training or Testing	Parameters
Dataset 1	07/10/1999 to 03/10/2016	Training	Historical Prices and Technical Indicators
Dataset 2	04/10/2016 to 09/04/2021	Testing	Historical Prices and Technical Indicators
Dataset 3	07/10/1999 to 03/10/2016	Training	Historical Prices
Dataset 4	04/10/2016 to 09/04/2021	Testing	Historical Prices
Dataset 5	01/01/2014 to 31/12/2017	Training	Historical Prices
Dataset 6	12/04/2021 to 09/06/2023	Testing	Historical Prices

Close – Open: Price difference between close and open.
Simple Moving Average (SMA): To get the average price over a certain number of periods, the simple moving average adds up the prices and divides the total by the number of periods. As an illustration, a 20-day simple moving average determines the average of the last 20 closing prices.

Exponential Moving Average (EMA): Similar to the SMA, the exponential moving average gives more weight to recent prices. It uses a smoothing factor or weightage to give more importance to recent data points. As a result, compared to the SMA, the EMA responds to price fluctuations more swiftly.

Relative Strength Index (RSI): It evaluates a price trend's strength and momentum. The RSI is represented as a bounded oscillator that fluctuates between 0 and 100. To compute RSI, first determine the average gain and loss for the given time period. Then, by dividing the average gain by the average loss, relative strength (RS) can be calculated. Apply the formula to determine the RSI:

$$\text{RSI} = 100 - \frac{100}{1+\text{RS}} \qquad (3)$$

Moving Average Convergence Divergence (MACD): It is a lagging oscillator that displays the correlation between the exponential moving averages (EMA) of short and long periods. The EMAs' convergence and divergence show the asset's trend's strength and weakness, respectively. These EMAs cross each other when the trend is about to reverse. A positive MACD number indicates strength, whereas a negative value indicates weakness in the underlying security.

$$\text{MACD} = \text{EMA}_{12} - \text{EMA}_{26} \qquad (4)$$

EMA12 indicates EMA with 12 days period and EMA26 indicates EMA with 26 days period.

Bollinger Bands: It displays the higher and lower price ranges that a stock is most likely to trade using moving average and standard deviation.
Momentum: It is determined utilising the price difference for a specific security over a given length of time. The rate at which a security's price rises or falls is used to assess a security's strength or weakness.

Dataset Preprocessing
The input dataset is scaled to the range of 0 to 1 for faster convergence. For that MinMaxScaler of sklearn library is used. Inverse transform is applied to predicted result before evaluating the performance metrics. Equation 5 shows the formula for Min-max scaling.

$$\text{xi scaled} = \frac{\text{xi}-\min(\text{x})}{\max(\text{x})-\min(\text{x})} \qquad (5)$$

Training datasets and Test datasets are grouped using sliding window mechanism. In sliding window Technique, M+1 days data constitute a window. In a window M days data is used as input to the prediction model to predict M+1 day's close price. The window is moved one step ahead to form the next window. For example, if first window ranges from 0 to 21 then second window will range from 1 to 22. Now the dataset is ready for testing and training.

Linear Forward Layer
Linear transformation of the input is done by Linear Forward Layer. Linear transformation equation is shown in equation 6.

$$\text{Y} = \text{A}^{\text{T}}\text{X} + \text{B} \qquad (6)$$

In equation 6, B is the bias, A is the weight matrix, X is the input and Y is the Output. This layer is used as Embedding layer on encoder and decoder input to map the input dimension from 21 to Embedding size. Here, embedding size is set as 256. This layer used after decoder output is used to map the dimension of the result from Embedding size to 1 to get the predicted close price.

2.4 Transformer Encoder and Decoder Layers
The attention mechanism is the basis for the encoder and decoder layers of transformers. When making predictions or producing outputs, the model's attention mechanism enables it to concentrate on particular areas of the incoming data. The attention mechanism's goal is to identify significant dependencies or links among the various pieces of the input sequence. Self-attention is a particular kind of attention mechanism that is applied to the same input or sequence. A model can pay attention to several points within a same sequence using self-attention. Multi-head attention is an advancement of the self-attention mechanism frequently employed in transformer-based models. By enabling the model to jointly attend to various features or positions of the input sequence, it improves the modelling capacity and flexibility of self-attention. Feed Forward network constitutes linear layer and dropout layer. Dropout layer makes some fraction of neurons inactive in each iteration. Dropout (0.1) means 10% of neurons are inactive. Add and Normalization operation combines input and output of its previous layer then performs layer normalization. If Number of layers and number of heads is set as 2 and window size is 60 then Table 3 shows input and output shape of each layer for one sample.

Table 3: Input and output shape in each layer

Layer (type)	Input Shape	Output Shape

Linear-1	[60, 21]	[60, 256]
TransformerEncoderLayer-2	[60, 256]	[60, 256]
TransformerEncoderLayer-3	[60, 256]	[60, 256]
Linear-4	[1, 21]	[1, 256]
TransformerDecoderLayer-5		[1, 256]
TransformerEncoderLayer-6		[1, 256]
Linear-7	[1, 256]	[1, 1]

2.5 Multi-Head Attention

In the context of the Transformer architecture, multi-head attention is used in both the encoder and decoder components. Instead of relying on a single attention mechanism, the input sequence is processed through multiple parallel attention heads, each learning its own attention weights and generating its own output representation. The choice of the number of attention heads is a hyper parameter, and it can be adjusted based on the complexity of the task and the available computational resources. The multi-head attention mechanism works as follows:

Input Transformation: The input sequence is transformed into separate Query, Key, and Value matrices for each attention head. These transformations are usually linear projections of the input embeddings.

Parallel Attention Computations: Each head independently performs the self-attention calculation. For each head, attention scores are computed by taking the dot product between its Query matrix and the Key matrix of all positions in the sequence.

Attention Weights: The attention scores for each head are scaled and normalized using softmax, producing attention weights specific to that head. Each position's relevance or significance for the corresponding head is determined by these attention weights.

Weighted Sum: For each head, the attention weights are applied to the Value vectors of all positions, resulting in a weighted sum.

Concatenation: The outputs from all heads are concatenated or linearly combined to form the final output representation. This allows the model to leverage multiple perspectives or aspects of the input sequence, capturing a more comprehensive understanding of the relationships and dependencies within the data.

Linear Transformation: The concatenated output is typically passed through a linear transformation layer, which maps it to the desired output dimensionality.

By employing multiple heads, each attending to different aspects of the input sequence, multi-head attention allows the model to capture and process information at different levels of granularity. It can learn different types of attention patterns or relationships within the data, enhancing its capacity to model complex interactions and dependencies. This enhances the expressiveness and effectiveness of the self-attention mechanism, leading to improved performance.

In the encoder module Query, Key, Value vectors are linear transformation of the embedded input. In decoder module, Query is the linear transformation of decoder input. Key and value vectors are output of the last encoder module. Input sequence length of the decoder must be equal to output window size. In this model output window size is one since only one step ahead prediction is needed. so input sequence length of decoder for a sample must be equal to one. Next chapter discusses the performance evaluation of our model in various conditions.

2.6 Evaluation Criteria

To know the correctness of the model, proper evaluation methods is needed. Commonly used evaluation metrics are RMSE, MAE, and R2 score for stock price prediction. So, the model's efficiency is evaluated using these metrics.

Root Mean Square Error (RMSE): RMSE is used to calculate the discrepancy between actual and projected outcomes. The square root of the mean of the squared differences between the predicted results and the actual values is calculated as part of the RMSE. It is an indicator of a regression model's general accuracy or goodness-of-fit. Small values of RMSE implies less error.

Mean Absolute Error(MAE): The mean of the absolute differences between the expected and observed

values is determined by MAE. Regardless of the direction of the mistakes, it provides a measurement of the average difference between the expected and observed values. Smaller the value of MAE, better the prediction. Unlike RMSE, MAE does not involve squaring the differences, which makes it less sensitive to outliers.

Coefficient of Determination(R2): It shows how effectively the regression model fits the actual data. The model's quality of fit is gauged by the R2 score. The R2 score is a number between 0 and 1, where 0 means that the model is worst fit and 1 means that the model is the best fit. So, the value close to 1 gives better performance. The RMSE, MAE, and R2 score formulas are provided in Equations 7, 8, and 9, respectively. The actual value is represented here by ai, pi stands for the predicted value, n for the total number of data points, and m for the mean of the actual values.

$$\text{RSME} = \sqrt{\left(\frac{1}{n}\right)\sum_{i=1}^{n}(ai - pi)^2} \quad (7)$$

$$\text{MAE} = \left(\frac{1}{n}\right)\sum_{i=1}^{n}|ai - pi| \quad (8)$$

$$\text{R2} = 1 - \frac{\sum_{i=1}^{n}(ai-pi)^2}{\sum_{i=1}^{n}|ai-pi|} \quad (9)$$

Results And Discussion

The proposed model for predicting Nifty 50's next day's closing price is compared with TechLSTM [24]. TechLSTM model incorporates technical parameters also in addition to historical prices as input parameters that's why it is called as TechLSTM. The proposed model uses the same training and testing data as TechLSTM for its training and testing. TechLSTM uses window size of 60. So, same window size is used for the comparison. Adam optimizer with MSE as the loss function is used for model training. Training batch size is set as 1024 and learning rate as 0.0001. Experiment is carried out in python using PyTorch library. Google Colab is used as experiment environment. In Google Colab GPU enabled configuration is used.

Hyper-parameter Tuning

Hyper-parameters of the suggested model are Embedding size, Number of layers, Number of heads and dropout rate. Dropout is set as 10%. Training and testing are done using dataset 1 and 2, respectively. Tab 4 shows error rate for different values of embedding size. When embedding size is set as 256, error rate is low. Table 5 shows error rate for different combination of number of heads and number of layers. When number of heads and number of layers is set as 2, error rate is less compared to others. So, less error rate configuration is used for comparison of the models.

Table 4: Embedding gsize comparison

Embedding size	RMSE	MAE	R2 Score
128	312	213	0.95
256	242	158	0.97
512	287	189	0.95
1024	483	400	0.88

Table 5: Number of heads and Number of layers comparison

Number of heads	Number of layers	RMSE	MAE	R2 Score
2	2	242	158	0.97
2	4	442	282	0.90
4	2	342	213	0.94
4	4	455	328	89

Model Comparison

TechLSTM model has two LSTM layers with neurons 128 and 64. Two dense layers having 25 neurons and one neuron follows this. Here for comparison, look back period is set as 60 and epoch as 100. Performance comparison of our proposed model and TechLSTM is given in Table 6. Figures 3 and 4 show predictions of the models and original Nifty 50 close price. Proposed model has lowest MAE and best R2 score. So, Proposed model outperforms LSTM based model.

Table 6: Comparison of Performance of the Models

Model	MAE	R2 Score
Multivariate Transformer	158	0.97
TechLSTM	236	0.94

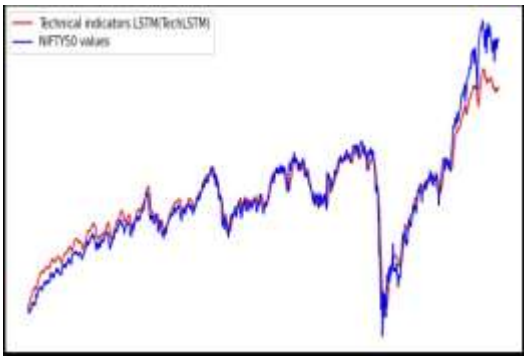

Figure 3. TechLSTM vs Nifty50

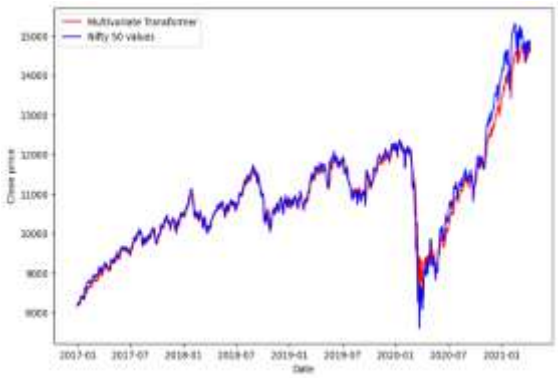

Figure 4: Multivariate Transformer vs Nifty 50

Performance Evaluation of the Model with Various Input Parameters

The model performance is compared with different input parameters. Suggested model is again trained using only Open, Low, High and Close price as input parameters. This model's performance is contrasted with that of the preceding model using the evaluation metrics RMSE, MAE and R2 score. Here, dataset 3 is used for training and dataset 4 is used for testing.

The performance comparison of these models is as shown in the Table 7. Multivariate transformer model trained with only historical prices have less error rate and best R2 score then the other model. Figure 5 shows Predicted close price using multivariate transformer model trained with only historical prices plotted against Nifty 50 actual close price.

Table 7: Comparison of performance of Multivariate Transformer with and without technical parameter

Model	No. of input	RSME	MAE	R2 Score

	parameters			
Multivariate Transformer including technical parameters	21	242	158	0.971
Multivariate Transformer including only historical prices	4	194	132	0.986

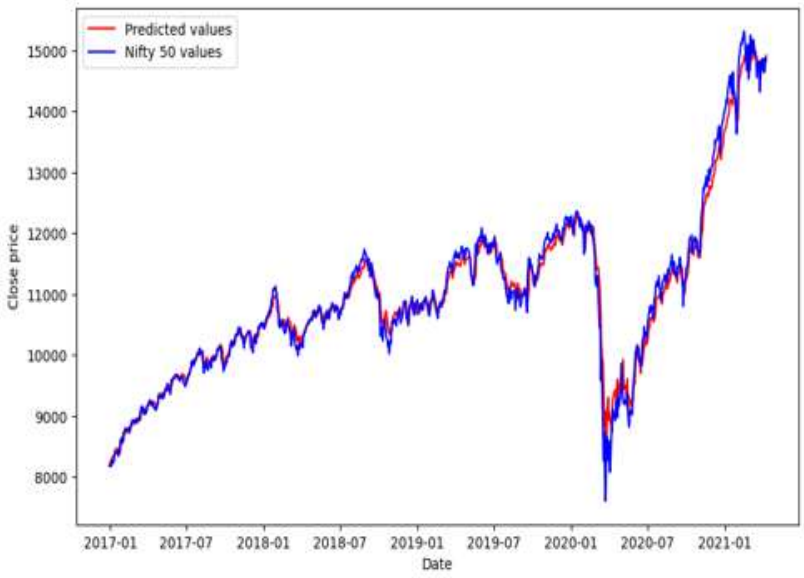

Figure 5: Multivariate Transformer including only historical prices Vs Nifty 50

3.4 Comparison of the Model's Performance for Various Look-Back Periods

To identify which window size gives better result, tested the proposed model after training with different window sizes. As input parameters, only the Open, High, Low, and Close prices are used. For testing, same number of data points is considered for all window sizes. Close price of trading days from 13/02/2017 to 09/04/2021 is predicted with different window size. It contains 1029 data points. Table 8 shows the analysis. From the analysis, it is found that input window size of 9 gives less error. Figure 6 shows its plot.

Table 8: Window size comparison

Window Size	RSME	MAE	R2 Score
7	249	214	0.968
8	180	139	0.983
9	177	129	0.984
10	243	203	0.969
20	231	182	0.972
30	198	144	0.979
40	229	181	0.972
50	236	186	0.970
60	197	134	0.979
70	359	329	0.930

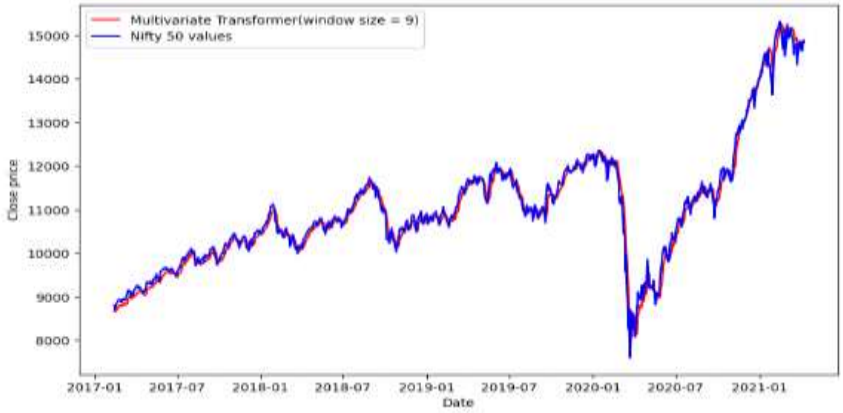

Figure 6. Predicted values of model with window size = 9 Vs Nifty 50 actual values

3.5 Training and Testing with Latest Dataset

In above cases, around 20 years of data is used to train the model. To measure effectiveness of the suggested model if trained with lesser amount of data, dataset 5 is used for training. It contains only 3 years of data. For training, 1000 epochs and batch size of 64 is used. Input window size is set as 9. Dataset 6 which includes latest stock prices is used for testing the performance. Evaluation metrics value for 528 days is given in the Table 9 and the predicted values are plotted in Figure 7. Table 10 shows evaluation metrics of 50, 100 and 200 days prediction. Figure 8,9 and 10 shows predictions for 50, 100 and 200 days plotted with Nifty 50 actual values. Even if training dataset size is less, correctness of the model is better.

From all the analysis carried out it is evident that proposed system gives acceptable predictions. The suggested model clearly outperforms all other models based on the Nifty 50 dataset.

Table 9: Error Rate

Evaluation Metrics	Value
RSME	173
MAE	131
R^2 Score	0.967

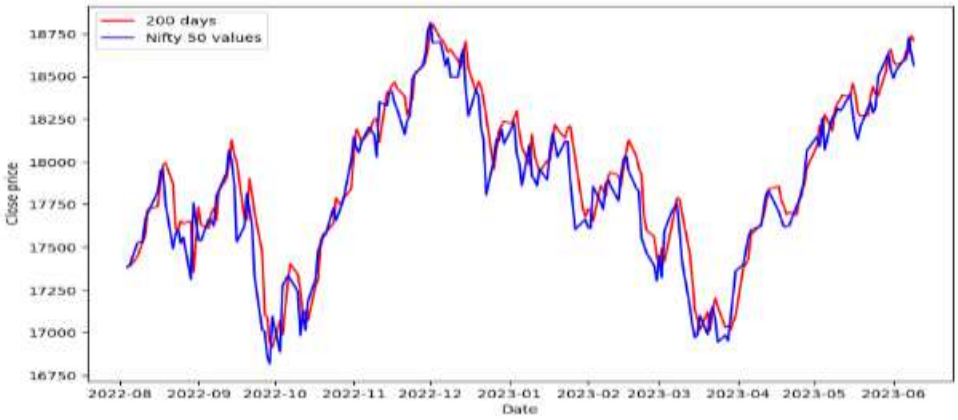

Figure 7: Predicted values Vs Nifty 50 actual values

Table 10: Error rate for different days of prediction

No. of Days	RSME	MAE	R^2 Score
50	108	89	0.961
100	137	110	0.903
200	137	114	0.899

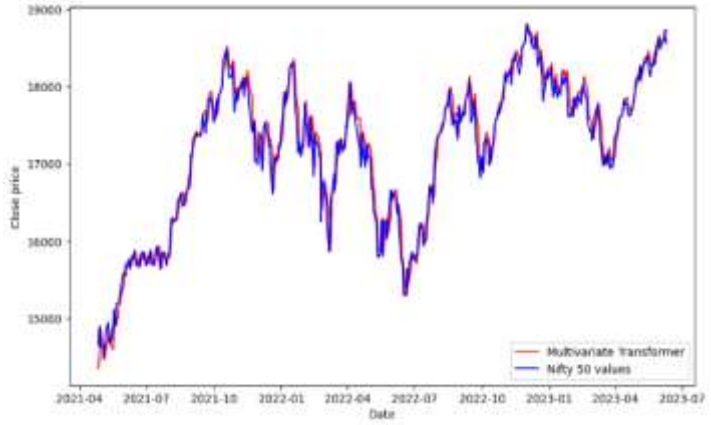

Figure 8: 50 days Prediction

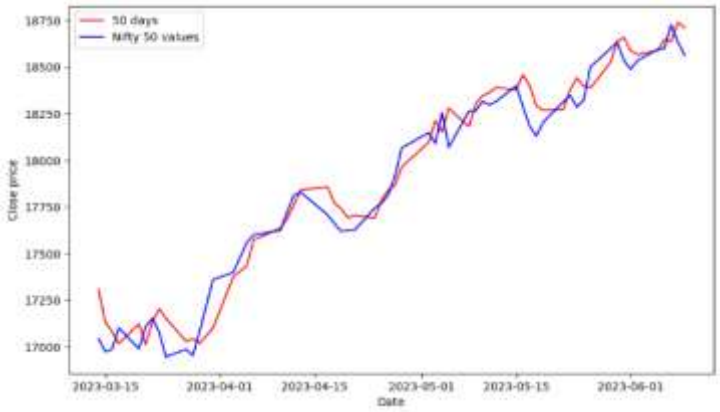

Figure 9: 100 days Prediction

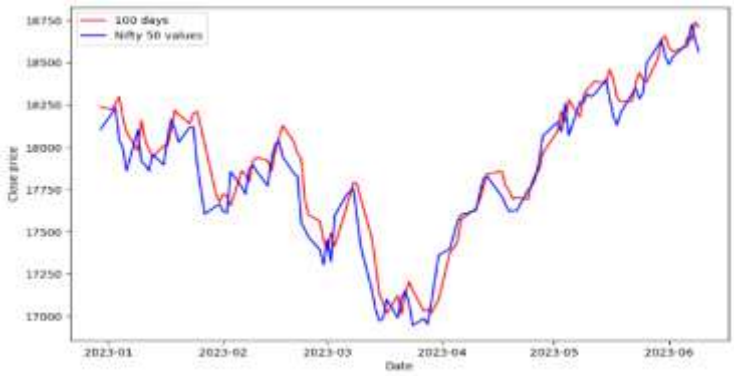

Figure 10: 200 days Prediction

Conclusion

Developing predictive models for dynamic and non-linear data such as stock market prices remains one of the most challenging problems in financial forecasting. Nevertheless, the successful development of such models holds significant value for investors seeking to maximise returns, for algorithmic trading systems requiring real-time decision support, and for policymakers engaged in economic planning, given that stock indices serve as widely accepted indicators of macroeconomic growth.

In this work, a Transformer-based deep learning model was developed to forecast the expected next-day closing price of the Nifty 50 index using historical stock price data spanning the previous M days. The proposed model incorporates multiple input variables, enabling it to capture complex temporal dependencies and market dynamics more effectively than univariate approaches. The prediction error of the proposed model was benchmarked against that of the TechLSTM model [24], and the results confirm that the proposed model outperforms the existing baseline. Notably, the proposed model demonstrated superior performance even when the input feature set was restricted to only the four primary price parameters — Open, High, Low, and Close — establishing its robustness under limited feature availability. Furthermore, the model achieved competitive results with smaller window sizes, indicating that it requires only a relatively small historical dataset for effective prediction, which further enhances its practical applicability.

References

1. P. M. Dechow, A. P. Hutton, and R. G. Sloan, "Short-sellers, fundamental analysis, and stock returns," Journal of Financial Economics, vol. 61, no. 1, pp. 77–106, 2001.
2. K.-Y. Shen and G.-H. Tzeng, "Combined soft computing model for value stock selection based on fundamental analysis," Applied Soft Computing, vol. 37, pp. 142–155, 2015.
3. H. Mizuno, M. Kosaka, H. Yajima, and N. Komoda, "Application of neural network to technical analysis of stock market prediction," Studies in Informatics and Control, vol. 7, no. 3, pp. 111–120, 1998.
4. T. Chenoweth, Z. Obradovic, and S. S. Lee, "Embedding technical analysis into neural network based trading systems," Applied Artificial Intelligence, vol. 10, no. 6, pp. 523–542, 1996.
5. I. Bhattacharjee and P. Bhattacharja, "Stock price prediction: A comparative study between traditional statistical approach and machine learning approach," in Proc. 4th Int. Conf. Electrical Information and Communication Technology (EICT), IEEE, 2019, pp. 1–6.
6. M. Nabipour et al., "Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis," IEEE Access, vol. 8, pp. 150199–150212, 2020.
7. H. Rezaei, H. Faaljou, and G. Mansourfar, "Stock price prediction using deep learning and frequency decomposition," Expert Systems with Applications, vol. 169, p. 114332, 2021.
8. Y. Lin et al., "Forecasting stock index price using the CEEMDAN-LSTM model," The North American Journal of Economics and Finance, vol. 57, p. 101421, 2021.
9. P. Lv et al., "Modal decomposition-based hybrid model for stock index prediction," Expert Systems with Applications, vol. 202, p. 117252, 2022.
10. H. Niu, K. Xu, and W. Wang, "A hybrid stock price index forecasting model based on variational mode decomposition and LSTM network," Applied Intelligence, vol. 50, pp. 4296–4309, 2020.
11. R. Bisoi, P. K. Dash, and A. K. Parida, "Hybrid variational mode decomposition and evolutionary robust kernel extreme learning machine for stock price and movement prediction on daily basis," Applied Soft Computing, vol. 74, pp. 652–678, 2019.
12. C. Deng et al., "Multi-step-ahead stock price index forecasting using long short-term memory model with multivariate empirical mode decomposition," Information Sciences, vol. 607, pp. 297–321, 2022.
13. J. Wang et al., "Asian stock markets closing index forecast based on secondary decomposition, multi-factor analysis and attention-based LSTM model," Engineering Applications of Artificial Intelligence, vol. 113, p. 104908, 2022.
14. W. Lu et al., "A CNN-BiLSTM-AM method for stock price prediction," Neural Computing and Applications, vol. 33, pp. 4741–4753, 2021.
15. C. Wang et al., "Stock market index prediction using deep Transformer model," Expert Systems with Applications, vol. 208, p. 118128, 2022.
16. A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, 2017.
17. G. Tang et al., "Why self-attention? A targeted evaluation of neural machine translation architectures," arXiv preprint arXiv:1808.08946, 2018.
18. S. M. Kazemi et al., "Time2Vec: Learning a vector representation of time," arXiv preprint arXiv:1907.05321, 2019.
19. C. A. Krishnapriya and A. James, "A survey on stock market prediction techniques," in Proc. Int. Conf. Power, Instrumentation, Control and Computing (PICC), IEEE, 2023, pp. 1–6.
20. S. Mehtab and J. Sen, "Stock price prediction using convolutional neural networks on a multivariate timeseries," arXiv preprint arXiv:2001.09769, 2020.
21. G. Singh, "Machine learning models in stock market prediction," arXiv preprint arXiv:2202.09735, 2022.
22. Z. Fathali, Z. Kodia, and L. B. Said, "Stock market prediction of NIFTY 50 index applying machine learning techniques," Applied Artificial Intelligence, vol. 36, no. 1, p. 2111134, 2022.
23. R. Shah et al., "Linear regression vs LSTM for time series data," in Proc. IEEE World Conf. Applied Intelligence and Computing (AIC), IEEE, 2022, pp. 1–6.
24. V. Jain et al., "HybridLSTM for NIFTY50 prediction using global indices and technical indicators," in Proc. 12th Int. Conf. Computing Communication and Networking Technologies (ICCCNT), IEEE, 2021, pp. 1–6.
25. S. P. Kallimath, N. Darapaneni, and A. R. Paduri, "Deep learning approaches for stock price prediction: A comparative study on Nifty 50 dataset," EAI Endorsed Transactions on Intelligent Systems and Machine Learning Applications, vol. 1, Feb. 2025.
26. S. Khan, D. Khandelwal, K. R. Khan, P. Sharma, and N. Sharma, "Stock price forecasting of the NIFTY-50 index using feedforward artificial neural networks with technical indicators," Journal of Interdisciplinary Knowledge, vol. 8, 2025, doi: 10.37497/jik.v8iknowledge.1653.