Stock Price Prediction Using Transformer-Based Model

Sparsh Aggarwal[1*]
Galgotias University,
Greater Noida,U.P., India
Sparshaggarwal561@gmail.com

Mr. Shashikant Sharma[2]
Galgotias University,
Greater Noida,U.P., India
Shahikant.sharma@galgotiasuniversity.edu.in

Shashwat Singh[1]
Galgotias University,
Greater Noida,U.P., India
Shashwat0923@gmail.com

Abstract - Stock price forecasting has been an important area of research in financial markets for decades owing to the non-linear, volatile nature of stock price movement. In this paper, the application of a transformer deep learning model to predict short-term stock prices from high-frequency data and technical indicators is explored. The data used here in this study is the Reliance Industries stock price history, sampled with a frequency of five minutes over one month duration. Then closing prices are used to calculate various features such as Simple Moving Average (SMA), Moving Average Convergence Divergence (MACD), and Relative Strength Index (RSI) which improve the predictability of the model. For verification of the model, it is separated into training subsets and test subsets and then normalized to overcome the problems of varying scales.

The proposed model is a position embedding multi-layer transformer model best for acquiring long-range correlation and temporal relation within the sequence of data. The model by means of operation in feed-forward layers and multi-head attention becomes the best suited for identifying sophisticated patterns that decide stock price movement. In addition, learning has been simplified utilizing advanced methods including model checkpoint, learning rate scheduling depending on a person's demand, and early stopping. These techniques provide faster convergence with improved generalization of the model.

Low Mean Absolute Error ensures the high predictive ability of the transformer-based model. A qualitative and graphical prediction of the stock price versus the actual stock price once again ensures the model's reliability for prediction of the short-term trend.

This paper brings into perspective the cross-domain transferability of the transformer models, initially intended for tasks of natural language processing for financial time-series forecasting.

The research justifies the potential of such models to be used in assisting algorithmic trading, portfolio management, and risk management practices. The study is a credible argument for further research into using other technical indicators as well as mixed models in furthering the precision of financial market forecasting.

1. Introduction

Stock price prediction is today a staple of financial research, with tremendous scope to enhance investment planning, specify algorithmic models of trading, and optimize risk management frameworks. However, accurate stock price prediction is a challenging endeavor, as stock prices follow extremely volatile and non-linear patterns. The interactive dynamics between macroeconomic variables, firm news, market sentiment, and investor psychology render the system extremely complex, with the price action exhibiting chaotic and intractable behavior. This complexity highlights the imperative for advanced predictive models that can detect advanced patterns in fiscal data..

For time-series forecasting of financial markets, conventional statistical models like Generalized Autoregressive Conditional Heteroskedasticity (GARCH) and Autoregressive Integrated Moving Average (ARIMA) have extensively been applied. Even if these methods are useful in capturing linear

relationships and short-run dependencies, they tend to be insufficient in capturing non-linear and long-run interdependencies that cause stock price movements. Due to their more general-purpose models for representing complex patterns in data, machine learning and deep learning techniques are increasingly popular as a result of this limitation.

Deep learning techniques are increasingly utilized in financial time series forecasting in the last two years, particularly in the cases of LSTM networks and CNNs. These models can capture more complex temporal and spatial dependencies than the standard approach and have much improved predictions. Models like LSTM, which are recurrent, are computationally costly and find it difficult to scale when dealing with large datasets or extremely long sequences. Moreover, their dependence on fixed-size windows tends to render them unable to capture global dependencies effectively.

Deep learning models, Long Short-Term Memory (LSTM) networks, and Convolutional Neural Networks (CNNs), have been in vogue for forecasting financial time series. These models capture temporal and geographical patterns in the data and are thus more precise than conventional methods. However, recurrent models like LSTM are computationally costly and barely scalable for large datasets or long sequences. They also suffer from inefficient modeling of global dependencies using fixed-size windows.

The objective of this paper is to predict the short-term closing prices of the top stock, Reliance Industries, of the Indian stock market using high-frequency data with a 5-minute sampling interval for one month. To improve the predictive ability of our model, some technical indicators, including the Simple Moving Average (SMA), Moving Average Convergence Divergence (MACD), and Relative Strength Index (RSI), will be derived from the raw data. The chosen indicators are widely applied in technical analysis to derive patterns and momentum in the market and are found to have valuable forecasting information. The research aims to forecast the closing prices for short-run horizons of Reliance Industries, an Indian blue-chip firm, using high-frequency data collected with a time interval of 5 minutes over a period of one month. To enhance the forecasting capability of the model, technical indicators like the Simple Moving Average (SMA), Moving Average Convergence Divergence (MACD), and Relative Strength Index (RSI) are calculated from raw data. Technical indicators are conventionally used in technical analysis to identify market trends and momentum, among others, and acquire useful forecasting information.

This research demonstrates the capability of transformer models to predict stock prices precisely, even considering dominant market drivers. This research uncovers the game-changing capability of modern machine learning techniques in finance by employing state-of-the-art deep learning models for financial time-series forecasting. Additionally, it opens up opportunities for further research with hybrid models and feature engineering, which all contribute to the ongoing upgrading of the solidity and reliability of predictors in finance.

2. Literature Review

Stock price prediction has been one of the largest research fields in computer science and finance, and of keen interest to scientists. Many statistical and machine learning methods have been experimented with and found helpful for enhancing such predictions.[8] How some of these have evolved over time, from the ancient statistical models to newer deep learning-based methods, like transformers, is what is being discussed here.

ARIMA and GARCH are two classical models that have been extensively utilized to model financial time-series data. ARIMA is able to model linear trends and short-run relationships very well, while GARCH has been extensively utilized in volatility forecasting. While useful, they are unable to model complex non-linear trends that govern stock prices as well as being unable to model long-run relationships very well [1].

Machine learning algorithms such as Decision Trees and Support Vector Machines (SVM) introduced the capability to learn non-linear patterns. SVMs were appropriate for small data sets and were outlier resistant. Decision Trees, because of an interpretability benefit, were not accurate when they worked on noisy data [2]. These were improvements over conventional methods but didn't scale and suffered with large data

Deep learning arrived eventually and truly changed, and architectures such as LSTMs and GRUs became the highlight. LSTMs solve the issue of vanishing gradients and can learn long-term relationships between things in an order. They have been very successful in stock price forecasting [3]. Their computationally costly nature and woeful poor performance when handling very, very long sequences has been their weak spot. CNNs have also been applied to learn neighborhood patterns from the stock data to improve the sequence model knowledge [4].

Transformers are also becoming an enormously powerful tool in financial prediction over the last few years. Initially proposed to be applied in natural language processing, transformers employ self-attention mechanisms for learning sequence dependencies. Transformers can learn long-range dependencies effectively without the constraint placed by recurrent models. Transformers have demonstrated extremely promising time-series application for predicting stock prices [7]. Their capacity to learn global dependencies makes them an early choice of application in this field [5].

Technical indicator usage such as SMA, MACD, and RSI has also been extensively researched in stock forecasting. The indicators decide underlying direction and trend momentum of movement of the markets and enable the models to forecast more accurately when used along with raw prices [6].

This research encases the time that has lapsed since techniques of share price prediction have evolved from traditional statistical methods to deep learning of the contemporary age. Transforms as a method of representing intricate dependences are some such advancements. Integration of such advancements into technical indicators is a research agenda of the future.

3. Methodology

The method for this project involves a few important steps to offer a just and reasonable way of predicting stock prices. We started by gathering data, next cleaned it to generate useful features, built a machine learning model, and experimented with its performance. Here is how each was performed:

3.1. Data Collection

The first part was obtaining quality stock market information. We utilized the Yahoo Finance API to obtain the historical data. The information included five-minute interval data spanning a month duration, including most important indicators such as Open, High, Low, Close, and Volume. This level of detail offered the fine degree of detail necessary to build a short-term prediction mode

3.2. Data Pre-Processing

The most crucial step in any machine learning model development is the preprocessing of the data. We were working on stock market data that had to be cleaned, transformed, and enriched from the raw data before being fed into the training process of a prediction model. We have used technical indicators such as SMA, MACD, and RSI. These are the indicators of market trend and momentum, critical to stock price prediction. We standardized the data so that all features are on the same scale, by subtracting the mean and dividing by the standard deviation. This makes the model learn more effectively. The formula for normalization is as follows:

$$Y_{norm} = \frac{Y - \mu}{\sigma}$$

Where:

- Y is the original feature value,
- μ is the average of the feature,
- σ is the deviation of feature from mean

After calculating indicators and applying normalization, some rows in the dataset contained missing values (NaN). This happens because indicators like SMA and RSI need a window of past data to calculate their values. We dropped rows with NaN values to avoid errors during training. The technical indicators we calculated are based on the "Close" prices. To ensure the dataset remains consistent, we aligned all calculated indicators with the corresponding closing price data. This step ensures that the indicators correctly match the time frames they describe.

By the end of these steps, we had a clean dataset with normalized features that could be directly used to train our model. Each step in preprocessing was carefully designed to help the model focus on meaningful patterns in the data.

3.3. Model Design

This project's stock price prediction model is built on the transformer architecture, a deep learning model that was first created for natural language processing but was later modified to work with a range of sequential tasks. By using self-attention processes, the transformer's primary benefit is its capacity to identify long-range dependencies in the data. The essential parts and design decisions for constructing the model are described in this section.

3.3.1. Input Layer

The input of the model is time series data of technical indicators and stock prices. At every time step, the model is presented with some features that consist of the closing price of the stock, moving averages (SMA, EMA), the Relative Strength Index (RSI), and Moving Average Convergence Divergence (MACD). Since we are applying a sliding window that has 24 time steps, each input sequence contains 24 data points, which in turn contain multiple attributes. With 24 being the number of time steps and 4 the number of features the input data has dimensions as (24,4).

3.3.2. Positional Encoding

Because transformers work in parallel over sequences and do not have the innate sense of order, we embed information on where each time step is located within the sequence with positional encoding. Positioning of data points holds highly important connotations for predicting time series; therefore, to obtain the positioning of the elements in the encoded dimension, we will be using the following formulas for calculating the positional encoding: For even indices:

$$\boldsymbol{PE(p,2i) = sin(\frac{p}{1000^{\frac{2i}{d}}})}$$

For odd indices:

$$\boldsymbol{PE(p,2i+1) = cos(\frac{p}{1000^{\frac{2i}{d}}})}$$

Where:

- $\boldsymbol{p}$ is the placement in the series,
- $\boldsymbol{i}$ is the index of the feature,
- $\boldsymbol{d}$ is the dimensionality of the positional encoding.

This encoding is added to the input features, allowing the model to use both the feature values and their positions in the sequence to make predictions.

3.3.3. Transformer Block

A stack of transformer blocks serves as the model's central component. Every transformer block is made up of two primary parts:

- **Multi-Head Self-Attention:**

Attention mechanism allows the model to quantify how significant each step of time is compared to other steps of time. The method allows the model to select most significant steps of time throughout the series in hopes to predict the subsequent stock price. Self-attention process is expressed as:

$$\boldsymbol{Attention(Q,K,V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V}$$

Where:

- Q is the query,
- K is the key,
- V is the value,
- $\boldsymbol{d_k}$ is the dimension of the key vector

The model can pay attention to independent sequence parts at every time step with this method. The model can pay attention to a single sequence part at every time step with this method.

- **Feed-Forward Neural Network:**

Following the attention mechanism, the output is fed through a feed-forward network. This is most commonly a fully connected layer with an activation function, often ReLU:

$$\boldsymbol{FFN(x) = ReLU(W_x + b)}$$

Where:

- $\boldsymbol{x}$ is the input
- $\boldsymbol{W_x}$ is the weight matrix
- $\boldsymbol{b}$ is the bias factor

Dropout is used following the attention layer and FFN layer to prevent overfitting. Residual connections follow both the attention layer and the feed-forward network. This enables the input to circumvent the general computation and then be

added onto the output again, which promotes smoother gradient flowing during backpropagation.

3.3.4. Global Average Pooling

After the transformer layers, the model's output is a sequence of representations for each time step. To reduce this sequence to a single prediction, **Global Average Pooling (GAP)** is applied. This operation computes the average of all values across time steps for each feature. The output of GAP is a fixed-size vector regardless of the input sequence length. The formula for GAP is:

$$GAP(x) = \frac{1}{T}\sum_{i=1}^{T} x_i$$

Where:

- x_i is the feature value at the i-th time step,
- $\boldsymbol{T}$ is the number of time steps (24 in this case).

This pooling operation helps condense the temporal information while preserving the learned features.

3.3.5. Output Layer

The output layer, which forecasts the stock price, is a completely connected layer. This layer has a single neuron with a linear activation function since the objective is to predict a continuous value:

$$\boldsymbol{Output = W_x + d}$$

Where:

- W are the weights,
- d is the bias term.

The final output is a scalar value representing the predicted stock price.

3.4. Training

The training phase feeds pre-processed input data to the model, optimizing its parameters for minimizing the loss function. Adam optimizer iteratively adjusts the weights based on the gradients calculated from the Mean Absolute Error (MAE) loss. A personalized learning rate scheduler is ensured by gradually increasing the learning rate during warm-up and gradually reducing it with time. Then, it is used for monitoring performance after the model has been trained on 100 epochs. Early stopping stops the training if there's no improvement in order not to overfit.

3.5. Evaluation

The model's performance was checked by testing the model using the test dataset. Mean Squared Error was the major statistic that would determine the prediction accuracy. Here, MSE refers to the average of squared differences between expected and actual values. The smaller the MSE, the better the performance of the model. During training, the test dataset was hidden from the model for an unbiased test. The generation of expected and actual results also tested the model's ability to discover patterns. This investigation has brought out the limitations and reliability of this model.

3.6. Visualization

In performance evaluation, MSE on the test set is also calculated. This measures the accuracy of predictions. Training and validation losses plotted against epochs show how the model learns. It keeps track of the state of a model and shows how the two training losses are connected, so it's easier to notice overfitting or underfitting. These will be plotted against the test dataset's actual values to further evaluate the capability of the model in identifying trends and patterns. This combination of measurements and plots will give a decent assessment of the model's performance.

4. Result

As shown by plots and measurements, the model predicts the stock prices very well. The model is capturing the trend in the data very well with hardly any overfitting; two confirmations are training loss and validation loss plots, both smooth descents in both measurements as training proceeds. Two losses converging could be a sign of very good generalization to new data:

Furthermore, the actual vs. predicted stock price plot is an argument for the correctness of the algorithm. The fact that the two curves are near to one another places the model in a position to match patterns and trends of the stock prices with accuracy, which is a minimum requirement in the case of a stock price predictive model-it means that the model does not simply predict isolated points but also picks up on big picture trends in the data set.

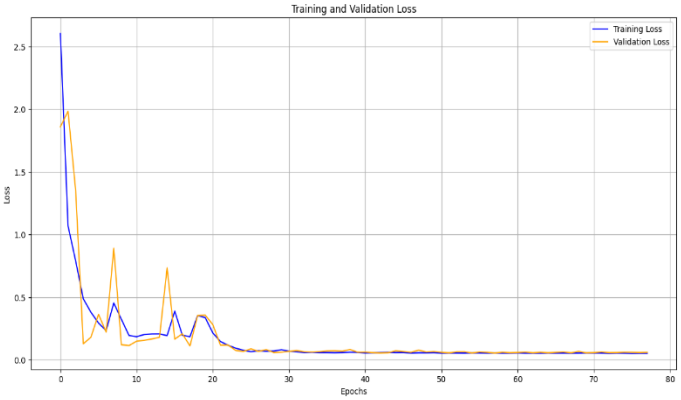
The model's test data MAE is 0.0787 numerically, i.e., the average difference between calculated and actual values of stocks is infinitesimally small. Also, because it dislikes proportionally more large-sized

errors than small-sized errors, the MSE of 0.0107 also attests that the errors are infinitesimally small-sized and uniformly sized. In addition to the graphical proof, these data attest how insensitive the model is.

This performance indicates that the model is capable of producing good predictions that approximate actual market action, and thus a good tool for stock price forecasting. Low MAE and MSE of the model

and noticeable patterns of the graphs indicate that the model can be effectively used to applications requiring good stock price predictions. near-real stock prices.

Fig 2 Training and Validation Loss



('Open', 'RELIANCE.NS')	('High', 'RELIANCE.NS')	('Low', 'RELIANCE.NS')	('Close', 'RELIANCE.NS')	('Volume', 'RELIANCE.N
1305.699951171875	1310.8499755859375	1304.300048828125	1310.199951171875	0.0
1310.0999755859375	1313.949951171875	1309.0999755859375	1313.5999755859375	428096.0
1313.6500244140625	1315.0	1311.0	1311.050048828125	294382.0
1311.050048828125	1313.25	1309.4000244140625	1310.3499755859375	280916.0
1310.1500244140625	1311.3499755859375	1309.300048828125	1311.1500244140625	198310.0
1311.1500244140625	1313.75	1310.449951171875	1312.0	226665.0
1312.0	1314.449951171875	1311.1500244140625	1311.4000244140625	228806.0
1311.4000244140625	1311.4000244140625	1309.4000244140625	1309.949951171875	236484.0
1310.0	1310.699951171875	1309.199951171875	1309.8499755859375	154954.0
1309.6500244140625	1310.0999755859375	1308.5	1309.5	180788.0
1309.6500244140625	1309.6500244140625	1307.949951171875	1308.050048828125	134936.0
1308.0	1308.050048828125	1307.0	1307.0	156579.0
1307.199951171875	1309.050048828125	1307.050048828125	1307.8499755859375	177803.0
1307.8499755859375	1310.199951171875	1307.5	1310.0999755859375	178021.0
1310.0999755859375	1310.199951171875	1308.449951171875	1309.300048828125	144499.0
1309.25	1309.9000244140625	1308.5	1308.5999755859375	156876.0
1308.5999755859375	1309.75	1308.5999755859375	1308.75	123503.0
1308.949951171875	1309.699951171875	1308.1500244140625	1308.199951171875	132502.0
1308.199951171875	1310.949951171875	1308.1500244140625	1309.9000244140625	196994.0
1309.8499755859375	1310.699951171875	1308.6500244140625	1309.0	138337.0
1309.0	1309.699951171875	1307.699951171875	1308.199951171875	235683.0

Fig 1(a) Reliance Stock Data

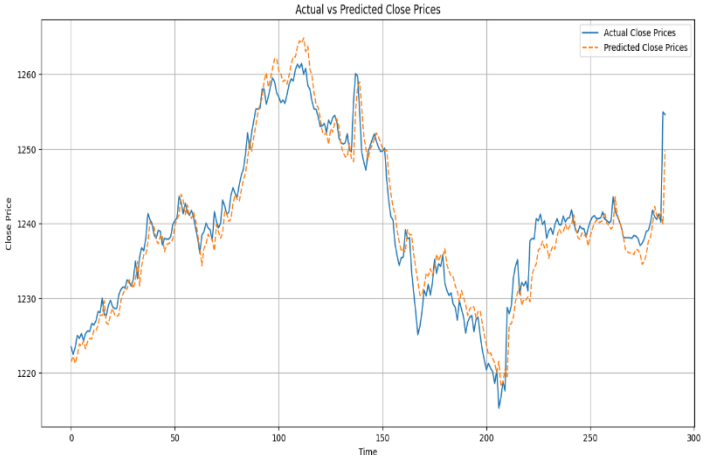


Fig 1(b) Predicted Stock Price

5. Conclusion and Future Scope

This body of research provides an answer to one of the biggest questions in finance prediction: it proves that deep models can actually perform out-of-the-box, and that performance is forecasting stock value to an incredibly high level of accuracy. Lower MSE and MAE values indicate that the model was sufficient in identifying and capturing the smaller waves of the market in an orderly manner. Repeated results demonstrate the model's prowess in making a quantum leap in addition and projection. Replication of the result from prediction to the actual stock price also confirms the model's ability to learn the patterns of the market, thus validating the model. Further model convergence confirmation was maintained by observing training and validation loss against epochs, thus stabilizing and preventing the possibility of overfitting. The findings allow for validity tests of this approach and thus enable stakeholders to oversee more risky financial markets and make investment decisions based on better information.

Since this project has been successful, there is massive scope for expansion and development in the near future. Incorporating other types of data, such as macroeconomic data, worldwide market trends, geopolitics, news, or social opinion about news media, will also make the model more predictable. These alternative sources of information will allow the model to incorporate externalities that are likely to impact the direction of stock prices and thus enhance the model's accuracy. Probing deeper structures, such as Transformers or attention-based

models, may also enhance accuracy, especially in highly volatile or non-linear market conditions. Ensemble methods, which cast multiple models at once, can reduce predictability uncertainty and enhance aggregate performance. To learn from a greater dataset, expanding the horizon in time for the history or other exchanges as data additions to the dataset would be one of the frontiers of research.

The stability of the system is achieved through this method under various states and conditions. Model decisions can remain simple and remain understandable for users if explanation mechanisms, such as SHAP (Shapley Additive Explanations), are utilized. Another potential application is to implement this model into a live trading environment. It would be of invaluable worth to investors and traders as a real-time forecasting tool and enhance their competitiveness at the decision-making level. This level of constant updating with new information about the market would make it dynamic in response to changes in trends, provided that the updates are ongoing and not erratic in the long run. This is a good bargain for financial future realization, and it positions machine learning as a valuable tool in finance.

6. References

1. [1] Box, G., Jenkins, G., & Reinsel, G. (2015). *Time Series Analysis: Forecasting and Control*. Wiley.
2. [2] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273-297
3. [3] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735-1780.
4. [4] Kim, T. Y., Kim, H. J., & Lee, K. J. (2019). Forecasting stock prices with convolutional neural networks. *Expert Systems with Applications*, 123, 125-133.
5. [5] Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal Fusion Transformers for interpretable multi-horizon time-series forecasting. *International Journal of Forecasting*, 37(4), 1740-1757.
6. [6] Murphy, J. J. (1999). *Technical Analysis of the Financial Markets*. New York Institute of Finance.
7. [7] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30, 5998-6008
8. "Proceedings of the 5th International Conference on Data Science, Machine Learning and Applications; Volume 1", Springer Science and Business Media LLC, 2025
9. Burak Yüksel. "Comparative Analysis of LSTM Model in Predicting ETF Stock Prices for Dierent Sectors" , Marmara Universitesi (Turkey), 2024