Predictive Modeling of Stock Prices Using Transformer Model

Leila Mozaffari (https://orcid.org/0009-0001-9120-0002), Department of Computer Science, AI Lab, Oslo Metropolitan University, Norway, s372060@oslomet.no **(mailto:s372060@oslomet.no)**
Jianhua Zhang (https://orcid.org/0000-0001-8051-4746), Department of Computer Science, AI Lab, Oslo Metropolitan University, Norway, jianhuaz@oslomet.no **(mailto:jianhuaz@oslomet.no)**

DOI: https://doi.org/10.1145/3674029.3674037 **(https://doi.org/10.1145/3674029.3674037)**
ICMLT 2024: 2024 9th International Conference on Machine Learning Technologies (ICMLT) **(https://doi.org/10.1145/3674029), Oslo, Norway, May 2024**

Financial market prediction utilizing deep learning has attracted the attention of both investors and researchers. Deep learning methods, such as convolutional neural networks and recurrent neural networks work well at predicting stock indices based on the non-linear characteristics of stock markets. The goal of this work is to predict the stock index using the latest deep learning framework, Transformer. This paper presents a comprehensive analysis of stock closing price prediction using three distinct machine learning models: Long Short-Term Memory (LSTM), Prophet, and Transformer. Using the encoder-decoder architecture and the multi-head attention mechanism, Transformer is able to better characterize stock market dynamics. The present study uses data from Yahoo Finance. The Transformer model demonstrated superior performance in comparison with LSTM and Prophet. In this work, we handle the complexities of market dynamics to improve stock price predictions.

CCS Concepts: • **Mathematics of computing** → **Time series analysis**; • **Computing methodologies** → **Neural networks**;

Keywords: Stock Market, Time Series, Transformer, Prophet, LSTM, Stock Price Prediction

ACM Reference Format:
Leila Mozaffari and Jianhua Zhang. 2024. Predictive Modeling of Stock Prices Using Transformer Model. In *2024 9th International Conference on Machine Learning Technologies (ICMLT) (ICMLT 2024), May 24--26, 2024, Oslo, Norway.* ACM, New York, NY, USA 8 Pages. https://doi.org/10.1145/3674029.3674037 **(https://doi.org/10.1145/3674029.3674037)**

1 INTRODUCTION

It has long been a challenge for investors, researchers, and data scientists to predict stock prices in the financial markets. Stock price forecasts can be immensely valuable to investors, helping them optimize trading decisions, manage portfolios, and mitigate risk [24]. In an attempt to uncover the complex patterns associated with stock price movements, robust models are continually being developed as a result of the attraction of potential financial gain from accurate predictions. The stock market, however, is a highly complex and dynamic system influenced by a multitude of variables and external factors, making predictions difficult [19].

There is great significance to stock price prediction, since it can yield substantial returns while minimizing losses. Trading decisions can be informed by successful forecasting, guiding traders on when to buy, sell, or hold stocks. Investors can also use it to protect their portfolios and hedge their positions, which can be a key component of risk management.

Stock price prediction presents a number of challenges. There are many factors that influence stock prices, including macroeconomic events, geopolitical shifts, market sentiment, and unforeseeable shocks. As a result of these inherent complexities, researchers have developed advanced prediction models designed to take into account this complex interaction of variables [1]. In recent years, neural networks, specifically deep learning models, have proven to be one of the most effective methods for stock price prediction. Since neural networks are capable of learning and adapting to non-linear patterns in data, they are well-suited to modeling stock markets, which are dynamic and unpredictable.

By optimizing neural network architectures, this research aims to improve the predictive accuracy of stock price models. It has been shown that neural networks are capable of forecasting financial time series, but the selection of appropriate hyperparameters (e.g., learning rate, hidden layer sizes, and training epochs) greatly influences their performance [13].

There is no doubt that the world of finance, particularly the stock market, is one of the most dynamic and consequential domains in the global economy. Companies and individuals can use it as a source of capital, as a measure of the health of the economy, and as an investment vehicle. Hence, stock market analysis is of paramount importance in this context. The purpose of in-depth market analysis is to help investors and financial analysts make informed decisions, optimize investment strategies, and manage risks.

Traditionally, two primary methods are used to analyze the stock market: fundamental analysis and technical analysis [10]. Fundamental analysis involves analyzing a company's financial statements, earnings, and overall health to determine its actual value. In contrast, technical analysis looks at past price and volume data to predict future stock price movements. There are, however, limitations to these methods, especially when it comes to dealing with vast and ever-changing financial data sets. This is where machine learning has found a valuable role in financial forecasting.

Stock market analysis can be revolutionized by machine learning, a subset of artificial intelligence. This method allows us to handle huge amounts of data in a way traditional methods cannot, allowing us to unlock new financial forecasting opportunities. Adaptive machine learning models learn from data and identify patterns that are difficult for humans to detect. The rest of this paper is organized as follows: A brief literature review is presented in Section 2. Our proposed method is described in detail in Section 3. Section 4 presents the results of a series of experiments conducted to test our approach. The final section concludes this paper and discusses future directions of research.

2 LITERATURE REVIEW

A significant revolution has taken place in the field of stock price prediction with the advent of machine learning techniques. Literature review highlighting relevant machine learning models and their applications in financial forecasting provides a comprehensive overview of the current landscape of stock price predictions.

In 1988, An important study of neural network models for stock price prediction was done by White [23]. IBM's daily common stock was used in his predictive model, and his training predictions were very optimistic. Later, many studies were conducted to test the neural network's accuracy of stock market forecasting.

Predictive models are complicated to build based on time series analysis of daily stock data. MR. Islam et al. [14] Compared three different methods for predicting stock prices, namely Autoregressive Integrated Moving Average (ARIMA), artificial neural network (ANN), and stochastic process-geometric Brownian motion (GBM). These methods are used to build predictive models using historical stock data collected from Yahoo Finance. A comparison is made between the outputs of each model and the actual stock price. For next-day stock price prediction, Using the S & P 500 index for analysis, the conventional statistical model ARIMA and the stochastic model-geometric Brownian motion model perform better than the artificial neural network models.

Priyank Sonkiya et al. [18] proposed an ensemble of state-of-the-art stock price prediction methods. A version of BERT, a pre-trained transformer model by Google for Natural Language Processing (NLP), is used to perform sentiment analysis of news and headlines for Apple Inc., listed on the NASDAQ. The stock price for Apple Inc is then predicted using technical indicators, stock indexes of various countries, some commodities, and historical prices along with sentiment scores. A comparison is made with baseline models such as Long Short Term Memory (LSTM), Gated Recurrent Units (GRU), vanilla GAN, and Auto-Regressive Integrated Moving Average (ARIMA). The proposed S-GAN model outperformed the traditional time series forecasting models GAN, GRU, LSTM and ARIMA. RMSE for training was observed at 0.5606 when learning rate was 0.00016, batch size was 128, and epochs were 165.

A number of back-testing experiments were performed by Chaojie Wang et al. [22] on the main stock market indices around the world, including the CSI 300, S&P 500, Hang Seng Index, and Nikkei 225. Several experiments have shown that Transformer outperforms other classic methods and can generate excess earnings for investors.

Muhammad Rizki Nur Majiid et al. [12] are collected datasets from Yahoo Finance and investing.com, as well as other aspects of the stock market. An extended dataset derived from Bank Central Asia's stock price is used to train these models. It includes open price, close price, low price, high price, and volume. Ensemble Transformer LSTM (ET-LSTM) and Ensemble Transformer GRU (ET-GRU) architectures forecast stock prices for the following day. In order to obtain more accurate findings, a variety of deep learning approaches are applied to the data set. A total of 9% of MAPE delivered by both of the suggested approaches using ensemble architecture. Stock prices are perfectly aligned between highs and lows. These models have been used to predict stock prices with remarkable success.

This domain, however, faces challenges. In order to predict stock prices, a variety of factors must be considered, including economic indicators, geopolitical events, and investor sentiment.

Therefore, machine learning models have been successfully used to predict stock prices. A combination of neural networks, hyperparameter optimization, and sentiment analysis offers a compelling framework for financial forecasting. Clearly, the field continues to evolve, and machine learning's application to stock price prediction holds great promise for investors, financial institutions, and researchers equally. In the following section, we will discuss the methodology and results of our study.

3 METHOD

3.1 Data Preprocessing

Financial data is rarely ready for machine learning models to be used immediately. To prepare the raw data for training and evaluation, a series of preprocessing steps were applied.

- **Missing Data Handling**
  In financial datasets, missing data is a common problem, which can be caused by a variety of factors. In order to prevent skewed or biased predictions, it is essential to address this issue [2].
- **Feature Selection**
  In financial datasets, many features are present, but not all of them are relevant to predicting stock prices [11]. Only those features that are most likely to contribute to the predictive power of the model were identified and retained.
- **Time Series Data Formatting**
  By nature, stock price data is a time series, and models must take this into account. Therefore, the data was organized into time series formats with features and target variables, allowing the model to learn from historical data and predict the future.
- **Outlier Detection**
  Model performance can be significantly impacted by outliers in the training data. Outliers were detected and handled appropriately using robust statistical methods and visualizations, such as box plots.
- **Z-Score Normalization** In data preprocessing, Z-Score normalization, also known as standardization, converts numerical data into a standard scale [15]. By rescaling the data, it has an average of zero and a standard deviation of one. It prevents one feature from dominating others during model training by creating a consistent scale across features or variables. The Z-Score normalization of a data point x is given by [4]:

  $$Z = \frac{X - \mu}{\sigma} \tag{1}$$

  where:
  z is the normalized value, x is the original value, μ is the mean of the data, σ is the standard deviation of the data.
  As a fundamental step in the predictive modeling process, data preprocessing is essential. By ensuring the data is of high quality and possesses the necessary attributes, machine learning models can capture patterns and relationships in stock prices accurately. Moreover, it facilitates more accurate predictions and insightful analysis in subsequent phases of research by mitigating common challenges like missing data and outliers.

3.2 Model 1: LSTM

A Long Short-Term Memory (LSTM) [8] is a type of Recurrent Neural Network that is designed to address challenges in learning long-term relationships among sequential data. In an effort to overcome the limitations of traditional RNNs, LSTMs incorporate memory cells and gate mechanisms to facilitate information retention and selective processing. LSTM networks have a number of key components. With its recurrent nature, the LSTM layer allows the model to capture dependencies over extended sequences from the input layer. A model's ability to retain information over time can be attributed to the cell state, along with the forget, input, and output gates.

LSTM's flexibility in handling sequential data, as well as its wide adoption in a variety of domains, including time series analysis and forecasting, is revealed in a comprehensive review of its formulation, training, and applications. Further research explores LSTM cell architecture variations and optimizations, highlighting its importance for tasks such as forecasting time series. To understand the LSTM architecture, it is necessary to recognize its evolution from basic neural network structures to the sophisticated and powerful models of today. With their ability to learn intricate long-term dependencies, these models have found applications in diverse fields, demonstrating their effectiveness in handling complex sequential data.

3.2.1 Model Architecture. [8]

- Input Layer:
  Sequential data is received by the input layer of LSTMs. In addition to these features, time-dependent information is crucial to the analysis.
- Cell State:
  LSTMs store and carry information across time steps using a cell state. RNNs with long-term dependencies have a hard time retaining them.
- Forget, Input, Output Gates:
  In LSTMs, information is controlled by three gates: Forget Gate, Input Gate, and Output Gate. In order to improve the model's ability to capture relevant patterns, these gates control information that is discarded, stored, or outputted.
- Memory Cells:
  In LSTMs, memory cells are crucial for storing and updating information. As a result, the model is able to remember past events, making it effective for predicting time series.
- Hybrid Architectures:
  Hybrid models can be created by combining LSTMs with other architectures, such as convolutional neural networks (CNNs). Combinations like this enhance the model's ability to understand spatial and temporal dependencies at the same time.

3.3 Model 2: Prophet Model

Prophet offers a robust methodology for forecasting that integrates seamlessly into research papers. Prophet [20], designed by Facebook, fits non-linear trends with yearly, weekly, and daily seasonality components using an additive model. In the architecture, holidays, trends, and seasonality are incorporated into a decomposable time series model. One of its key strengths is its ability to handle outliers and missing data effectively. The Prophet software contributes significantly to forecasting tasks by providing accurate predictions and flexibility in handling diverse datasets. Many studies combine Prophet with other techniques for enhanced forecasting performance, such as Long Short-Term Memory (LSTM) networks [17].

3.3.1 Model Architecture. [3]

- Additive Time Series Decomposition:
  Time series are broken down into three components using Prophet's additive decomposition model: trend, seasonality, and holidays. Decomposing the data in this way helps us to gain a deeper understanding of the fundamental patterns.
- Trend Modeling:
  Overarching trends are represented by the trend component. To accommodate diverse data patterns, Prophet uses a piecewise linear model to capture both abrupt and gradual changes in the trend.
- Seasonality Modeling:
  Time series data are heavily influenced by seasonality. A Fourier series expansion is incorporated into Prophet in order to identify and predict repeating patterns over time.
- Holiday Effects:
  It is common for time series data to be influenced by holidays. The Prophet model can incorporate holiday effects, allowing it to account for the impact of holidays on observed data.
- Parameter Tuning:
  There are several parameters that must be carefully tuned in the Prophet model, such as changepoints, seasonalities, and holidays. To enhance model performance, the methodology should detail how parameters were selected and any adjustments made.
- Forecasting Uncertainty:
  Prophet provides a unique feature by estimating the uncertainty intervals around forecasts. Decision-makers need this information, and the methodology should detail how Prophet quantifies and presents forecast uncertainty.

3.4 Model 3: Transformer Model

The Transformer model is a revolutionary architecture in deep learning, initially introduced in the paper "Attention Is All You Need" by Vaswani et al [21]. Transformers have achieved remarkable achievements across diverse domains. In the context of Natural Language Processing (NLP), they have proven their abilities in language translation, sentiment analysis, and text summarization. Expanding their applicability to Image Processing, transformers have been adeptly tailored for vision tasks, demonstrating success in image classification and object detection. Furthermore, their effectiveness extends to Time Series Analysis, where their unique ability to capture long-range dependencies renders them suitable for predicting sequential data, demonstrated in tasks such as forecasting stock prices or predicting weather patterns.

3.4.1 Model Architecture. [6]

- Self-Attention Mechanism:
  Unlike traditional recurrent neural networks (RNNs) and long short-term memory networks (LSTMs), Transformers rely on self-attention mechanisms. This enables the model to weigh the importance of different parts of the input sequence when making predictions.
- Parallelization:
  The Transformer's architecture allows for highly parallelized computation, making it more efficient than sequential models like RNNs. This results in faster training times.
- Encoder-Decoder Structure:
  The model is composed of an encoder and a decoder. The encoder processes the input sequence, capturing contextual information, while the decoder generates the output sequence.
- Multi-Head Attention:
  The self-attention mechanism is extended with multiple heads, allowing the model to focus on different aspects of the input sequence simultaneously. This enhances its ability to capture complex relationships.
- Positional Encoding:
  Transformers do not inherently understand the order of the input sequence. To address this, positional encodings are added to the input embeddings, providing information about the positions of tokens in the sequence.

3.5 Model Performance Metrics

In the domain of model evaluation, several metrics offer insights into the performance of machine learning models. Here are explanations and formulas for key evaluation metrics [7]:

- Mean Absolute Error (MAE):
  MAE measures the average absolute difference between predicted and actual values [7].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2}$$

- Mean Squared Error (MSE):
  MSE quantifies the average squared difference between predicted and actual values [7].

$$MSE = \frac{1}{n}\sum_{i-1}^{n}(y_i - \hat{y}_i)^2 \tag{3}$$

- Root Mean Squared Error (RMSE):
  RMSE is the square root of MSE, providing a measure in the original unit of the target variable [7].

$$RMSE = \sqrt{\frac{1}{n}\sum_{i-1}^{n}(y_i - \hat{y}_i)^2} \tag{4}$$

- Mean Absolute Percentage Error (MAPE):
  MAPE calculates the percentage difference between predicted and actual values on average [7].

$$MAPE = \frac{1}{n}\sum_{i-1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \tag{5}$$

These metrics aid in assessing the accuracy and reliability of machine learning models, providing valuable insights for model selection and improvement.

4 EXPERIMENTAL RESULTS AND DISCUSSION

4.1 Dataset

This study focuses on historical records between 2013 and 2023 and fetching the stock data from Yahoo Finance [5], which includes essential attributes such as Low, Open, Volume, High, Close, and Adjusted Close prices. AAL (American Airlines Group Inc.) and AAME (Atlantic American -Life Insurance) stock market indices were selected from Yahoo Finance in order to provide insight and analysis. The dataset contains historical stock prices and other important financial data that can be used to analyze the market dynamics of these companies. Market movements can be examined in a comprehensive way, enabling the examination of historical patterns, correlations, and predictive indicators. Financial data from Yahoo Finance can be accessed with Yfinance, a popular Python library. This API tool provides a comprehensive set of features for retrieving stock-related information, historical data, and fundamental company information. The dataset is split into three subsets for training, validation, and testing. Approximately 70% of the data is allocated for training. 15% for validation, and another 15% for testing.

4.2 Software and Hardware Requirements

The experiments were conducted utilizing the Jupyter Notebook. The system specifications included an Nvidia Tesla T4 GPU, 12GB of CPU memory, an Intel(R) Xeon(R) CPU running at 2.30GHz, and 12GB of RAM. Python served as the primary programming language for running the experiments, and the PyTorch library was employed for the computational tasks.

4.3 Data Preprocessing

Different preprocessing steps are performed on a dataset. It begins by counting the number of missing values in each column of the DataFrame, where there were no null values. Feature selection [11] aims to enhance model performance by focusing on the most relevant and informative features while discarding potentially redundant or less significant ones. In this case, ’Low,’ ’Open,’ and ’High,’ are chosen to serve as input variables for training a predictive model and predicting the target variable, ’Close,’ which represents the closing stock prices. Next, the "Date" column is converted to a DateTime format. Outlier analysis [16] is performed on the ’Open’ and ’Close’ columns. By determining the first quartile (Q1) and third quartile (Q3), the Interquartile Range (IQR) is calculated. Outliers are defined as data points that fall outside the lower and upper bounds of the IQR. Based on this, there were no outliers in the ’Open’ and ’Close’ columns. The data is normalized using z-score normalization, where each feature is standardized by subtracting its mean and dividing by its standard deviation. By normalizing, all features are brought to a similar scale, preventing one feature from dominating another.

4.4 Architectural Settings

4.4.1 LSTM: LSTM Model inherits from the nn.Module class and the constructor method initialize the LSTM model. It takes parameters such as input size, hidden size, and number of layers. The batch_first=True argument indicates that the input data has the batch size as the first dimension. A fully connected (linear) layer is added that transforms the output of the LSTM layer to the desired output size. The forward pass of the model takes an input x and computes the forward pass through the LSTM layer. The hyperparameters set hidden_size_lstm = 64 and num_layers_lstm = 2 for the LSTM model and the Mean Squared Error (MSE) loss function and the Adam optimizer defined to update the model parameters during training, with the learning rate 0.001.

4.4.2 Prophet: The historical stock prices are stored with the 'Date' column renamed to 'ds' and the 'Close' column to 'y'. Additionally, the timezone information is removed to ensure compatibility with Prophet. An instance of the Prophet class is created with the name m. The parameter daily_seasonality=True is set to include daily seasonality patterns in the model. The historical stock price data (hist) is used to train the Prophet model using the fit method. The make_future_dataframe method is employed to create a DataFrame (future) that extends beyond the historical data, projecting into the future for a specified period (365 days in this case). The prediction method is then applied to the future data frame, generating a forecast for the stock prices. The code includes a line (pd.options.display.max_columns = None) that ensures all columns are displayed when outputting the forecast.

4.4.3 Transformer: The model starts with an embedding layer (self.embedding) that transforms input data of size input_size to a higher-dimensional space represented by hidden_size. The core of the model is formed by the Transformer layers (self.transformer), following the architecture introduced in the paper "Attention is All You Need". It consists of multiple encoder and decoder layers, each containing multi-head self-attention mechanisms. Parameters such as num_layers and num_attention_heads allow fine-tuning of the model's capacity to capture different levels of contextual information. The final layer (self.fc_output) maps the output from the Transformer layers back to the original output dimensionality (output_size), facilitating the prediction of the target variable. The input sequence x is passed through the embedding layer to obtain a higher-level representation. The input sequence is permuted to match the input requirements of the Transformer. The Transformer processes the sequence, considering both encoder and decoder aspects, with self-attention mechanisms capturing contextual relationships. The output from the Transformer layers is permuted back to the original shape. The last layer's output is used for prediction through the fully connected output layer.

4.5 Hyperparameters

Table 1 depicts the hyperparameters for the models. Time series prediction is defined by the hyperparameters provided. A measure of the dimensionality of input features is the input_size, which is derived from the length of the feature_columns. A hidden layer of 64 neurons is incorporated into the network with a hidden_size of 64, which allows for intricate patterns to be captured. The output_size is set to 1 when predicting a single time series value. In hierarchical feature learning, 2 num_layers implies the use of two layers. The addition of four num_attention_heads enhances the model's ability to focus simultaneously on multiple input aspects. With a learning_rate of 0.001, the optimization step size influences convergence stability. A 100-epoch setting balances learning without the risk of overfitting the entire dataset. Adam optimizer, which adjusts learning rates dynamically, is chosen for efficient training. By measuring the squared difference between predicted and actual values, the MSELoss function quantifies prediction accuracy.

Table 1: The models Hyperparameters

Hyperparameters	Values
input_size	len(feature_columns)
hidden_size	64
output_size	1
num_layers	2
num_attention_heads	4
learning_rate	0.001
num_epochs	100
Optimizer	Adam
loss function	MSELoss

4.6 Data Analysis Experimental Results

The full code for the models can be found on GitHub:

https://github.com/layali64/Transformer-Model-Yahoo-Finance (https://github.com/layali64/Transformer-Model-Yahoo-Finance)

4.6.1 LSTM Model: The first model was trained on AAL stock Market indices of Yahoo Finance. Figure 1 illustrates the training, validation, and test losses over 1000 epochs for the LSTM model. The blue line represents the training loss with a value of 0.0002. The orange line represents the validation loss with 0.0651 and the green line corresponds to the test loss with a loss of 0.1972.

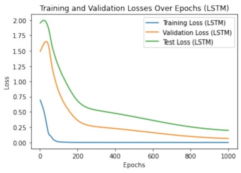

Figure 1: Values of the loss function for LSTM model over the AAL dataset.

Figure 2 depicts real and predicted values for assessing the performance of the LSTM model. The blue line in the plot represents the true or actual values of the target variable across different time steps. The orange dashed line illustrates the values predicted by the LSTM model for the same time steps. An alignment in the middle of the plot between the true and predicted values indicates a well-performing model.

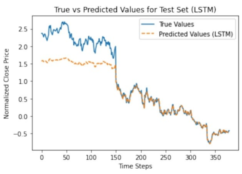

Figure 2: Comparison between LSTM model predictions and the ground truth over the AAL dataset.

4.6.2 Prophet Model: Figure 3 shows the result of the prophet model. Based on the specified date range from 2013 to 2023, the black dots represent data points used to train the model. A blue line shows the trend predicted by the Prophet model. The stock price trend is captured by considering historical patterns and seasonality. The light blue area around the forecasted trend represents the intervals and range of uncertainty. Based on the model's uncertainty, it shows the upper and lower bounds for actual prices. As can be seen from the plot, the prophet model predicted the close stock price from 2023 to 2024 for one year. It is possible to make informed decisions about future stock price movements for 2024 by comparing forecasted trends with actual stock prices and examining the uncertainty range.

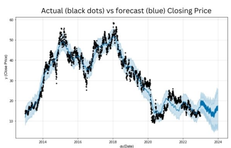

Figure 3: Comparison between Prophet model predictions and the ground truth over the AAL dataset.

4.6.3 Transformer Model: With 100 epochs, the final model was trained using AAL indexes from Yahoo Finance. Losses for training, validation, and testing of the Transformer model are shown in Figure 4. With a loss of 0.0100, the blue line represents the training loss. A loss of 0.0103 represents the validation loss, while a loss of 0.0085 corresponds to the test loss.

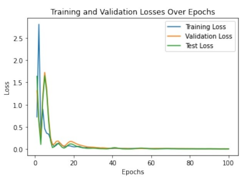

Figure 4: Values of the loss function for Transformer model over the AAL dataset.

Transformer model performance is shown in Figure 5. Based on the plot, the blue line represents the actual target variable value. Using an orange dashed line, the transformer model predicted values at the same time. A robust model has true and predicted values aligned in the plot.

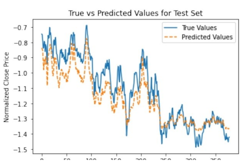

Figure 5: Comparison between Transformer model predictions and the ground truth over the AAL dataset.

In the third model, AAME indexes of Yahoo Finance were used for training with 100 epochs. The Transformer model's training, validation, and test losses are shown in Figure 6. As shown by the blue line, the training loss is 0.0185. Validation losses are 0.0501, while test losses are 0.0118.

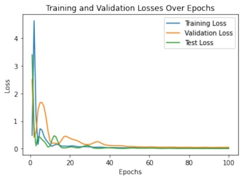

Figure 6: Values of the loss function for Transformer over the AAME dataset.

In Figure 7, real and predicted values are shown for assessing Transformer model performance. According to the plot, the blue line represents the true values of the target variable. The transformer model predicted values for the same time using an orange dashed line. The true and predicted values are aligned in the plot, which indicates a well-performing model.

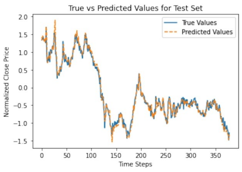

Figure 7: Comparison between Transformer model predictions and the ground truth over the AAME dataset.

4.7 Comparative Results

The evaluation metrics for the validation and test sets provide quantitative insights into the model's performance, including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). LSTM model achieves values in MAE (0.1855), MSE (0.0651), and RMSE (0.2551), indicating superior performance in accuracy and error metrics. Table 2 shows the comparative results of the models on Validation data. Prophet model has higher error metrics with MAE (6.8927), MSE (73.3724), and RMSE (8.5658), suggesting lower accuracy compared to LSTM with market indices AAL. Both Transformer models demonstrate competitive results, with AAL outperforming AAME in all metrics.

Table 2: Performance of the six models on the Validation set.

Model	Stock indices	MAE	MSE	RMSE	MAPE
LSTM	AAL	0.1855	0.0651	0.2551	16.9622
LSTM	AAME	0.7320	0.0194	0.1393	5.9126
Prophet	AAL	6.8927	73.3724	8.5658	-
Prophet	AAME	3.0534	10.7997	3.2863	-
Transformer	AAL	0.0827	0.0103	0.1016	8.2274
Transformer	AAME	0.1421	0.0501	0.2238	10.4670

Table 3 shows the comparative results of the models on test data. In the LSTM model, MAE, MSE, and RMSE are 0.2890, 0.1972, and 0.4440, respectively.

Although the prophet model improves compared to the validation set, MAE (3.5499), MSE (16.9321), and RMSE (4.1149) still exhibit higher error metrics.

The AAL Transformer model consistently outperforms the AAME Transformer model.

Table 3: Performance of the six models on the Test set.

Model	Stock indices	MAE	MSE	RMSE	MAPE
LSTM	AAL	0.2890	0.1972	0.4440	24.3381
LSTM	AAME	0.6890	0.0095	0.0976	14.7863
Prophet	AAL	3.5499	16.9321	4.1149	-
Prophet	AAME	4.1472	17.8601	4.2261	-
Transformer	AAL	0.0793	0.0085	0.0923	8.0455
Transformer	AAME	0.0826	0.0118	0.1085	16.8889

4.8 Discussion

Table 4 provides a comparative analysis of various models used for stock index prediction, with metrics such as RMSE, MAE, and Symmetric Mean Absolute Percentage Error (SMAPE). Different approaches and their performance on different market indices are highlighted in the results.

Table 4: Comparison of our modeling results with those in literature.

Author	Stock Index	Model	Metric	Value
MRN Majiid 2023 [12]	BBCA	ET-LSTM ET-GRU CNN-BiLSTM-AM	RMSE	490.3815 493.7659 2224.1882
Hu. Xiaokang 2022 [9]	Google Google S&P 500 S&P 500	Temporal Fusion Transformer (TFT)	MAE SMAPE MAE SMAPE	275.67 0.2642 52.77 0.0655
Chaojie Wang 2022 [22]	CSI 300 Hang Seng S&P 500	CNN Transformer RNN	MAE	0.0948 0.0881 0.1359
The Proposed Model	AAL AAME AAL AAME AAL AAME	LSTM LSTM Prophet Prophet Transformer Transformer	MAE	0.2890 0.6890 3.5499 4.1472 **0.0793** 0.0826

Majiid et al. [12] utilized ET-LSTM and ET-GRU models for BBCA stock, achieving RMSE values of 490.3815 and 493.7659, respectively. In contrast, Xiaokang Hu [9] employed Google Temporal Fusion Transformer (TFT) on Google stock, reporting a lower MAE of 275.67 and SMAPE of 0.2642. Chaojie Wang [22] implemented Transformer models on Hang Seng indices, demonstrating superior performance with MAE values of 0.0881. The proposed LSTM and Prophet model for AAL stock yielded an MAE of 0.2890 and 3.5499, respectively, while Transformer AAME and AAL achieved MAE values of 0.0826, and 0.0793, respectively.

Based on these results, it is important to select models that are appropriate to specific market conditions. In spite of the fact that some models exhibit remarkable accuracy, the choice depends on a variety of factors, including the dataset characteristics, the model architecture, and the targeted market index. For stock index prediction endeavors, researchers and practitioners can use the comparative values to determine which models are most effective.

5 CONCLUSION AND FUTURE WORK

The purpose of this study was to explore a comprehensive approach to predicting stock prices by using three different models: LSTM, Prophet, and Transformer-based models.

LSTMs are known to capture long-term dependencies in sequential data. Facebook's Prophet model demonstrated its ability to handle non-linear trends, seasonality, and holidays in time series data. Improved performance was achieved with the Transformer-based model.

In terms of accuracy metrics, the Transformer model consistently outperformed other two models on the AAL stock index with MAE, MSE, RMSE and MAPE value of 0.0793, 0.0085, 0.0923, and 8.0455, respectively.

A comparison with previous studies revealed competitive performance of Transformer model, indicating the importance of model selection based on market conditions and dataset characteristics.

The results obtained in this investigation demonstrate the feasibility of different models in predicting stock prices. It is imperative that future research in this domain takes into account the peculiarities of financial datasets and explores innovative approaches to improving the accuracy and reliability of stock price predictions.

In the future, along this line of research we need to develop hybrid model architectures. Using LSTM model and Transformer model in joint force might result in superior forecasting accuracy by combining temporal sequence learning and attention mechanism. Furthermore, comparing emerging algorithms and models in machine learning across different financial markets, datasets, and timeframes will contribute to a deeper understanding of their performance. Hopefully, these future efforts would lead to enhanced accuracy, robustness, and interpretability of stock price predictive model.

REFERENCES

[1] C Anand. 2021. Comparison of stock price prediction models using pre-trained neural networks. *Journal of Ubiquitous Computing and Communication Technologies (UCCT)* 3, 02 (2021), 122–134. Navigate to

[2] Svetlana Bryzgalova, Sven Lerner, Martin Lettau, and Markus Pelger. 2022. Missing financial data. *Available at SSRN 4106794* (2022). Navigate to

[3] Zheng Chen, Yin-Liang Zhao, Xiao-Yu Pan, Zhao-Yu Dong, Bing Gao, and Zhi-Wen Zhong. 2009. An overview of Prophet. In *Algorithms and Architectures for Parallel Processing: 9th International Conference, ICA3PP 2009, Taipei, Taiwan, June 8-11, 2009. Proceedings 9*. Springer, 396–407. Navigate to

[4] Wikipedia contributors. 2024. *Standard Score*. https://en.wikipedia.org/wiki/Standard_score **(https://en.wikipedia.org/wiki/Standard_score)** Accessed on February 4, 2024. Navigate to

[5] Yahoo Finance. 2024. *Yahoo Finance*. https://finance.yahoo.com/ **(https://finance.yahoo.com/)** Accessed on February 4, 2024. Navigate to

[6] Anthony Gillioz, Jacky Casas, Elena Mugellini, and Omar Abou Khaled. 2020. Overview of the Transformer-based Models for NLP Tasks. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 179–183. Navigate to

[7] JM González-Sopeña, V Pakrashi, and B Ghosh. 2021. An overview of performance evaluation metrics for short-term statistical wind power forecasting. *Renewable and Sustainable Energy Reviews* 138 (2021), 110515. Navigate to

[8] Alex Graves and Alex Graves. 2012. Long short-term memory. *Supervised sequence labelling with recurrent neural networks* (2012), 37–45. Navigate to

[9] Xiaokang Hu. 2021. Stock price prediction based on temporal fusion transformer. In *2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*. IEEE, 60–66. Navigate to

[10] Nti Isaac Kofi, Adebayo Felix Adekoya, and Benjamin Asubam Weyori. 2020. A systematic review of fundamental and technical analysis of stock market predictions. *The Artificial Intelligence Review* 53, 4 (2020), 3007–3057. Navigate to

[11] Salim Lahmiri. 2016. Features selection, data mining and finacial risk classification: a comparative study. *Intelligent Systems in Accounting, Finance and Management* 23, 4 (2016), 265–275. Navigate to

[12] Muhammad Rizki Nur Majiid, Renaldy Fredyan, and Gede Putra Kusuma. 2023. Application of Ensemble Transformer-RNNs on Stock Price Prediction of Bank Central Asia. *International Journal of Intelligent Systems and Applications in Engineering* 11, 2 (2023), 471–477. Navigate to

[13] Félix Morales, Carlos Sauer, Hans Mersch, Diego Stalder, and Miguel García Torres. 2022. Hyperparameter optimization of deep learning model for short-term electricity demand forecasting. In *Proceedings of the 3rd South American International Conference on Industrial Engineering and Operations Management*. Navigate to

[14] Nguyet Nguyen and Mohammad Islam. 2021. Comparison of Financial Models for Stock Price Prediction. In 2021 Joint Mathematics Meetings (JMM). AMS. Navigate to

[15] SGOPAL Patro and Kishore Kumar Sahu. 2015. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462* ***(https://arXiv.org/abs/1503.06462)*** (2015). Navigate to

[16] Karanjit Singh and Shuchita Upadhyaya. 2012. Outlier detection: applications and techniques. *International Journal of Computer Science Issues (IJCSI)* 9, 1 (2012), 307. Navigate to

[17] S Sivaramakrishnan, Terrance Frederick Fernandez, RG Babukarthik, and S Premalatha. 2022. Forecasting time series data using arima and facebook prophet models. In *Big data management in Sensing*. River Publishers, 47–59. Navigate to

[18] Priyank Sonkiya, Vikas Bajpai, and Anukriti Bansal. 2021. Stock price prediction using BERT and GAN. *arXiv preprint arXiv:2107.09055* ***(https://arXiv.org/abs/2107.09055)*** (2021). Navigate to

[19] Zihe Tang, Yanqi Cheng, Ziyao Wang, *et al*. 2021. Quantified Investment Strategies and Excess Returns: Stock Price Forecasting Based on Machine Learning. *Academic Journal of Computing & Information Science* 4, 6 (2021), 10–14. Navigate to

[20] Sean J Taylor and Benjamin Letham. 2018. Forecasting at scale. *The American Statistician* 72, 1 (2018), 37–45. Navigate to

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017). Navigate to

[22] Chaojie Wang, Yuanyuan Chen, Shuqi Zhang, and Qiuhui Zhang. 2022. Stock market index prediction using deep Transformer model. *Expert Systems with Applications* 208 (2022), 118128. Navigate to

[23] Halbert White. 1988. Economic prediction using neural networks: The case of IBM daily stock returns. In *ICNN*, Vol. 2. 451–458. Navigate to

[24] Yan Yu. 2022. A Study of Stock Market Predictability Based on Financial Time Series Models. *Mobile Information Systems* 2022 (2022). Navigate to

ICMLT 2024, May 24–26, 2024, Oslo, Norway

ACM ISBN 979-8-4007-1637-9/24/05.
DOI: https://doi.org/10.1145/3674029.3674037 **(https://doi.org/10.1145/3674029.3674037)**