Comparing XGBoost and LSTM Models for Prediction of Microsoft Corp's Stock Price Direction

Osaigbokan Peter Uhunmwangho

Department of IT-Management, International University of Applied Science, North Rhine Westphalia, Germany

Author's Email: osaigbovopeter869@gmail.com

Abstract

This study investigates the effectiveness of XGBoost and LSTM models in predicting the directional changes in Microsoft stock price using incremental values to mitigate inaccuracies in Directional Accuracy Percentage (DAP) evaluations. A twenty-three-year Microsoft stock prices dataset from Yahoo Finance spanning January 1st, 2000, to December 31st, 2023 is utilized, the dataset is scaled, and technical indicators are computed. Both models are trained on the extracted features to predict directional changes, evaluated using the Directional Accuracy Percentage (DAP). The XGBoost model achieves an accuracy metric value of 71.02%, whereas LSTM ranges from 62.42% to 67.10%. A Mann-Whitney U test confirms a significant difference in predictive performance, favouring XGBoost. This finding suggests XGBoost as a more effective tool for short-term decision-making in stock trading, underlining its potential to improve the predictive accuracy in financial forecasting and advancing predictive modelling research.

Keywords: XGBoost, LSTM, Microsoft stock prices, Directional Accuracy Percentage (DAP).

1.0 Introduction

More often than ever, it is not a secret that investors desire the value of their investment to grow greatly. The values of these investments, which are denominated by stock prices, represent how valuable companies are. The more valuable a company is, the more profitable it is for its shareholders. Hence, before investing, investors will seek to determine in advance if the stock prices or value of a company will grow in the foreseeable future.

The ability to forecast stock price movements accurately is essential to investors, financial institutions, and policymakers alike for informed decision-making purposes. Accurately predicting the values of companies to rise or to fall is basically challenging and difficult. Thanks to the host number of fundamental variables and multi-market conditions that needed to be evaluated. There is a vast amount of historical data that needs to be analysed as well. The

unpredictability inherent in stock prices necessitates sophisticated predictive models that leverage advanced computational techniques. Machine learning, particularly algorithms such as xGboost and Long Short-Term Memory (LSTM), has emerged as an important tool in stock price prediction tasks, offering promise in enhancing predictive accuracy and robustness.

This paper focuses on evaluating and contrasting the predictive capabilities of xGboost and LSTM models in determining the directional change of Microsoft stock prices. Microsoft Corporation is a multinational computer technology company. Bill Gates and Paul Allen founded Microsoft on April 4, 1975, in Albuquerque, New Mexico. Its current best-selling products include the Microsoft Windows operating system; Microsoft Office, a suite of productivity software; Xbox, a line of entertainment of games, music, and video; Bing, a line of search engines; and Microsoft Azure, a cloud services platform (Wikipedia contributors, 2023). Currently, it is also making giant strides in AI technologies, and Microsoft is the most valuable company in the world with a market capitalization of $3.342 trillion (Forbes, 2024).

XGboost is a gradient boosting algorithm which is known for its ensemble learning approach and capability to handle structured data, unlike LSTM, which is a type of recurrent neural network that is designed to capture temporal dependencies in sequential data. The essence of the comparisons of these two machine learning models lies in their different methodologies and theoretical frameworks. XGboost works by iteratively improving the performance of weak learners, whereas LSTM excels in capturing long-term dependencies through its memory cells and gates. Understanding how these two models perform in the specific context of stock price prediction is crucial for informed decision-making by investors and refining financial forecasting methodologies and techniques (Olah, 2015; Xia et al., 2019; Luo et al., 2021; Barrera-Animas et al., 2022; XGBoost, 2024).

The comparison between XGBoost and LSTM is significant for several reasons:

i. Different Methodologies: Their different approaches highlight which model may be more effective for specific data types and forecasting tasks.

ii. Handling of Data Types: Financial data can be both structure and unstructured. XGBoost may perform better with structured data, whereas LSTM is excellent at analyzing time series data.

iii. Predictive Accuracy: Evaluating both models allows researchers to determine which better captures the patterns of stock price movements, leading to improved investment decisions.

iv. Computational Efficiency: XGBoost generally offers faster computation and training times compared to LSTM, which may require more resources due to its complexity. Understanding these trade-offs is crucial for practitioners selecting a model for real-time applications.

v. Application in Financial Contexts: Insights gained from this comparison can significantly impact investment strategies and risk management practices, which could lead to better decision-making and more robust financial models.

vi. Insights into Model Interpretability: XGBoost provides greater interpretability due to its reliance on decision trees, helping investors understand feature importance. In contrast, LSTM can often act as a "black box," and evaluating both models offers insights into how interpretability affects trust and usability in financial contexts.

Related Research Works

The prediction of stock price movements has been a core focus of financial research which is driven by the need for effective trading strategies and investment decisions. Amongst various predictive models, machine learning approaches, particularly XGBoost and LSTM networks, have gained prominence due to their capability to handle complex, non-linear data relationships and patterns. Machine learning in stock price prediction recent literature has pointed out the effectiveness of machine learning algorithms in financial forecasting. There is no doubt that traditional statistical method like ARIMA and GARCH, often struggle with the non-stationary nature of stock prices (Chindanur, 2014). In contrast, machine learning models can adaptively learn complex patterns from historical data which make them suitable for this task (Beg et al., 2019).

XGBoost is an ensemble machine learning method, and per researches, has shown exceptional performance in various predictive tasks which include but not limited to stock price forecasting. Chen and Guestrin (2016) introduced XGBoost, emphasizing its speed and accuracy, which stem from its gradient boosting framework that optimizes for computational efficiency and predictive power. Studies have demonstrated that XGBoost can effectively capture the non-linear relationships in stock data, outperforming traditional models in terms of prediction accuracy (Hongjoong, 2021). XGBoost has additionally been proved to be effective when integrated with other models. Oukhouya et al. (2024), observed that employing the LSTM-XGBoost hybrid model for forecasting daily prices of major stock indices which include MASI, CAC 40, DAX, FTSE 250, NASDAQ, and HKEX which yielded superior prediction accuracy compared to using LSTM and XGBoost separately. Similarly, Shi et al. (2022) concluded that

hybrid models are more effective with notably high prediction accuracy. They integrated a time series model, Convolutional Neural Networks with Attention mechanism, Long Short-Term Memory network, and XGBoost regressor to capture complex, nonlinear relationships in historical stock market data across various time periods.

LSTM networks are a type of recurrent neural network designed to learn from sequences of data which make them ideal for time-series forecasting. Hochreiter (1997), first proposed LSTMs to address the vanishing gradient problem associated with standard recurrent neural networks. Recent studies have shown that LSTMs can effectively capture the temporal dependencies inherent in financial time series, leading to improved prediction outcomes (Elminaam et al., 2024). LSTMs have been employed in various stock price prediction tasks, demonstrating their capability to handle high-dimensional and sequential data, as well as combined effectively well with other models (Hochreiter, 1997). In the study of Elminaam et al. (2024), two neural network models, which include LSTM and Deep Neural Network (DNN), were used to predict the weekly and daily stock prices of the Indian BSE Sensex index employing historical Tech Mahindra stock data ranging from 1997 to 2017. It was observed that the DNN model returned a smaller RMSE compared to LSTM, demonstrating better accuracy in predicting close prices. In contrast, LSTM exhibits a reduced prediction bias compared to DNN, which means that it tends to make predictions closer to actual values without consistently overestimating. Overall, both models showed strong predictive abilities for the stock's daily closing prices in the analysis. In addition, the authors extended the focus to weekly stock predictions, which cover a 7-day trading period. Researchers utilized Directional Accuracy (DA) as an index to assess both models. Based on the findings, the LSTM model performed better than the DNN in this aspect.

Furthermore, Korstanje (2021), confirmed that by adding more features, the accuracy of LSTM predictions could be improved. They performed research focusing on feature extraction, dataset analysis, and stock price prediction using a LSTM neural network model. Historical stock data from the CSI 300 Index was obtained from the stock data interface of the JoinQuant platform, which spanned the period from May 18, 2014, to January 29, 2017.

Test results demonstrated an accuracy of approximately 0.66 for the single-layer LSTM model, whereas the three-layer LSTM model returned an accuracy exceeding 0.78. This pointed out that increased layers correlated with stronger forecasting outcomes with high demand for computational resources. In conclusion, the study suggested that improving the prediction

accuracy could be achieved by integrating more extracted attributes for training the LSTM model.

Moreover, several studies have aimed to analyse and compare the predictive performance of XGBoost and LSTM in stock price prediction. For instance, studies by (Kristanti et al., 2024; Zhou, 2024) evaluated both models on multiple stocks, revealing that while LSTM generally excelled in capturing long-term dependencies, XGBoost often outperformed LSTM in shorter time frames and less complex datasets. The authors suggested that the choice of model may depend on the specific characteristics of the data and the prediction horizon. Similarly, another comparative analysis by Zhu (2003) highlighted the strengths and weaknesses of both models. The authors found that XGBoost demonstrated superior performance in terms of speed and interpretability, while LSTM provided better accuracy when predicting long-term price movements. Their findings suggest that hybrid models, which leverage the strengths of both approaches, could offer a promising avenue for future research. Corroborating this, Hossain & Kaur 2004) highlighted the complementary strengths of XGBoost in tabular data processing and LSTM in time-dependent capture.

Through evaluation and comparative analysis, this research aims to contribute insights into which model, XGBoost or LSTM, demonstrates superior accuracy in forecasting the direction of Microsoft's stock price movement. By examining performance metrics and discussing the practical implications of the findings of this research work, the study seeks to advance the discourse on the application of machine learning in financial forecasting, thereby bridging the gap between theoretical concepts and real-world applications in predictive analytics.

2.0 Data Collection and Pre-processing, Methodology and Experimental Setup

2.1 stock price data description

For the prediction tasks, historical datasets on Microsoft Corporation stock prices are collected from Yahoo finance. To download data from Yahoo finance, first, the website had to be accessed, and the Microsoft stock price dataset then had to be accessed on the website. There are many operations that the website allows one to perform on the data such as view analysis of the stock, which included metrics such as Microsoft earnings estimates, revenue estimates, and earning history among others. One can also view the statistics on the stock, these have metrics such as stock price history, stock profitability, and stock management effectiveness and these can give good summary statistics on the stock and help one to quickly analyze a stock at face value.

The data of concern was the stock history, this is provided by the website and entails the stock information recorded daily, excluding weekends and holidays when the stock market is closed (other combinations such as weekly and monthly are also possible). The data collection allows a user to choose a start and end date to be able to choose stock over a given period. Stock price datasets covering a comprehensive 23-year period from January 1, 2003, to December 31, 2023, are obtained. Once the Microsoft Corporation stock datasets are obtained, data exploratory analysis was performed on the raw data. Yahoo finance has a good user experience, and is easy to operate even for a novice, and provides good clean data that would require no pre-processing to be able to use.

The dataset consists of thousands of rows and seven columns, which include date, open, close, high, low, adjusted close, and volume. The data demonstrates fluctuating trends and recurring peaks and troughs, indicating that the Microsoft Corporation stock price is affected by a blend of short-term and long-term factors. These fluctuations may correlate with economic reports, geopolitical events, technological advancements, or shifts in market sentiment, which are particularly difficult and challenging to forecast. Additionally, the dataset reflects only the historical prices of Microsoft Corporation, ignoring any companies that may have been competitors or relevant to the context but no longer exist. Moreover, fluctuations in stock prices may heavily reflect market sentiment influenced by news or events which could overshadow underlying fundamentals. Again, the choice of daily data collection might introduce noise from short-term market fluctuations which could also obscure longer-term trends.

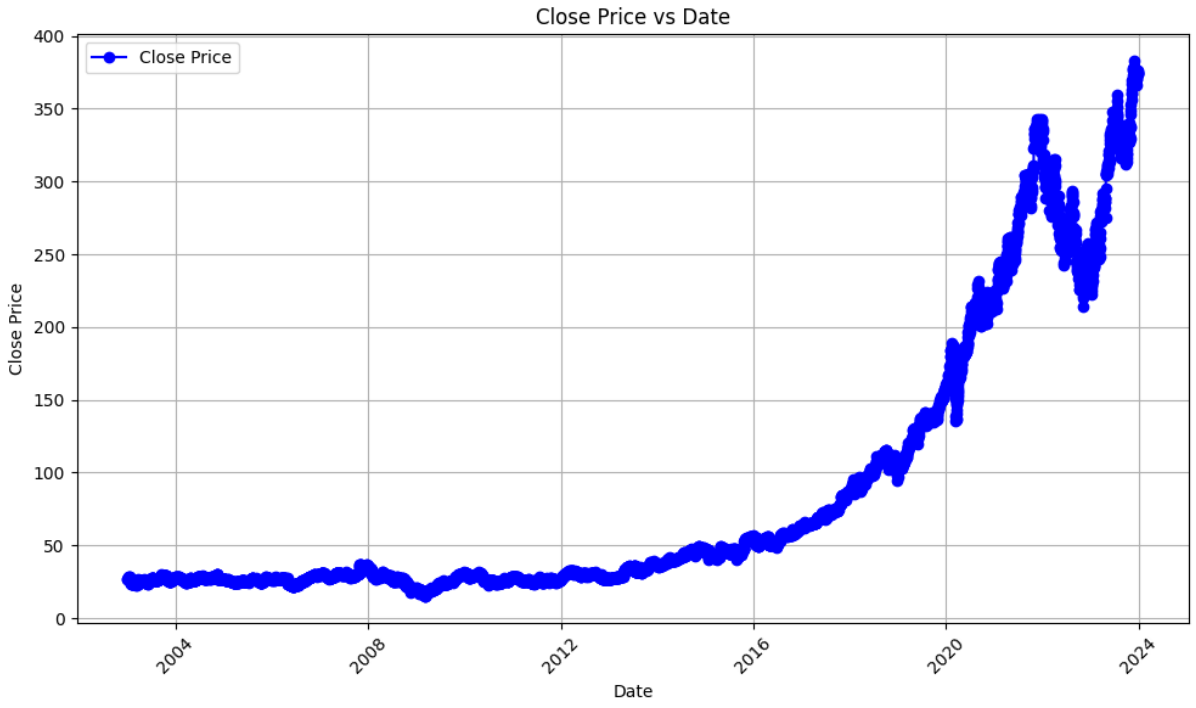

Figure 1: plot showing closing rates of Microsoft stock

2.2 Extreme Gradient Boosting Model Description

XGBoost is the short form for Extreme Gradient Boosting, an efficient machine learning algorithm and it is classified under ensemble learning methods. It implements gradient boosting, which focuses on optimizing both speed and model performance. It operates within the gradient boosting framework. XGBoost improves forecasting accuracy by summing output from numerous weak learners, which are referred to as the decision trees. Sequentially integrated into an ensemble, and the aim of each tree is to correct errors from the combined existing models. XGBoost incorporates regularization techniques such as L1 (LASSO) and L2 (Ridge) regularization to prevent over-fitting tendencies, which improve the model's ability to generalize effectively (XGBoost, 2024).

Gradient boosting simply works by generating subsequent models to forecast the residuals or errors of previous models by combining these predictions to make the final prediction. The term "gradient boosting" comes from its use of a gradient descent algorithm to cut loss while incorporating new models. This ensemble learning technique is highly touted for its effectiveness in managing structured tabular data, leading to improved performance of decision tree models. The benefits of the xGboost model include but not limited to robust handling of missing data, prevention of over-fitting, and efficiency improvements through parallel and distributed computations (Luo et al., 2021). In contrast, the performance of XGBoost can be affected by the size and distribution of the dataset, with potential declines in detection accuracy as datasets become more imbalanced. The basic outline of the gradient boosting method is as follows:

Input is given as:

Training set:

$$\{(x_i, y_i)\}^2 \quad (1)$$

$$x_{i=features} \quad (2)$$

$$y_i = target\ variable \quad (3)$$

Differentiable loss function:

$$L\big(y, F(x)\big) \quad (4)$$

The number of iterations is N (5)

Model initialization:

$$F_0(x) = argmin_\gamma \sum_{i=1}^{n} L(y_i, Y) \quad (6)$$

$F_0 =$ initial prediction model (7)

$Y =$ constant value (8)

For $N = 1\ to\ n,$ (9)

Compute initial prediction by calculating the mean value of the target variable

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \quad (10)$$

determine pseudo-residuals Calculate the difference between the actual target values and the initial predictions: for $i = 1 \ldots n$

$$Y_m = -\left[\frac{\partial L(y_i, F(X_i))}{\partial F(X_i)}\right], \text{where } F(X) = F_{m-1(x)} \quad (11)$$

The next step is to fit a base learner or weak learner, for example tree: $g_m(X)$ to pseudo-residuals, using training set

$\{(X_i, Y_m)\}_{i=1,}^{n}$ where n is the number of instances. Determine constant multiplier:

$$Y_m = argmin_\gamma \sum_{i=1}^{n} L\big(y_i, F_{m-1}(X_i) + Y g_m(X_i)\big) \quad (12)$$

Update the model:

$$F_m(X) = F_{m-1}(X) + \gamma Y_m g_m(X) \quad (13)$$

Here, γ is the learning rate

Output the final model

$$F_N(X) \quad (14)$$

2.3 Long Short-term Memory Model (LSTM) description

The LSTM model, a type of recurrent neural network (RNN) is developed to address the problem of gradient vanishing that often occurs in traditional RNN training. In contrast to

standard feedforward neural networks, LSTM includes feedback connections which allow it to effectively capture long-range interrelationships in sequential data. Its architecture is made up of three essential gates, which include the input gate, the forget gate, and the output gate (Olah, 2015).

The update gate transmits new information into the cell state. In contrast, the forget gate identifies and removes information considered unnecessary by the model. Meanwhile, the output gate controls the amount of information output as activations for the next layer. Compared to a standard RNN, which has a simple structure with a single tanh layer, LSTM is more complex. It has four interconnected layers within each repeating module arranged in a chain-like structure, as shown in the diagram below. Each line in the diagram denotes a complete vector, transmitting from the output of one node to the inputs of others. Pink circles indicate pointwise operations like vector addition, while yellow boxes represent learned neural network layers. Lines merging denote concatenation, and branching lines imply duplicating content directed to different destinations (Olah, 2015).

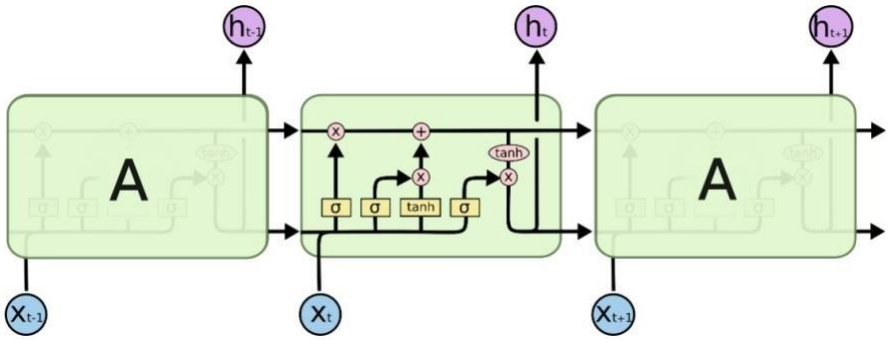

Figure 2: shows LSTM network layers.

Source: Olah, 2015.

LSTMs offer several benefits in time series forecasting. They are efficient and effective at capturing complex temporal patterns and trends, specifically when structured in multi-layer networks. LSTMs are effective in handling long-term dependencies and adapting to highly nonlinear and non-stationary data. They excel at learning trends and temporal relationships within time series data. In addition, LSTMs can be utilized in combination with other models such as CNNs to encompass both extended temporal dependencies and local trend features. LSTM also demonstrates strong abilities in sequential modelling and shows promise in clinical prediction (Xia et al., 2019).

However, LSTM models have their disadvantages as well, which include but not limited to challenges like over-fitting and discrepancies, especially over prolonged time periods between observed and predicted values. These issues pose significant difficulties in the complex task of time series prediction (Barrera-Animas et al., 2022). The performance of LSTM models can also be affected by the number of features used during training, and designing optimal LSTM architectures and fine-tuning parameters can be a demanding and complex task that requires human supervision.

The comprehensive steps below outline how LSTM models are initialized, update their cell states, and compute outputs in the context of forecasting. The use of two LSTM layers with dropout improves the model's ability to generalize and make accurate predictions. The basic LSTM modeling process includes the following steps:

- Initialization of Network Parameters:

$$X = (X_1, X_2 \ldots . X_4) \quad (15)$$

- Forget Gate Layer: Use a sigmoid layer called the "forget gate layer" to decide what information to forget from the cell state. The forget gate layer analyzes the previous hidden state and the current input. Output values range between 0 and 1, with 1 indicating "completely keep this" and 0 indicating "completely get rid of this." And it is calculated as:

$$F_t = \sigma\big(W_i(h_{t-1}, X_t) + b_f\big) \quad (16)$$

- Update the cell state (C_{t-1}) based on the decision made by the forget gate layer, resulting in the new cell state (C_t). To update the cell state, compute the input gate and the candidate values to be added to the cell state.
- Inpute gate is caculated as:

$$i_t = \sigma(w_i . [h_{t-1}, x_t] + b_i) \quad (17)$$

- Candidate values is computed as:

$$\hat{C}_t = tanh(w_c . [h_{t-1}, X_t] + b_c) \quad (18)$$

- Updating the cell state using the forget gate and input gate:

$$C_t = F_t . C_{t-1} + i_t . \hat{C}_t \quad (19)$$

- Calculate the output gate to determine which parts of cell state to output

$$O_t = \sigma(W_o . [h_{t-1}, X_t] + b_o) \quad (20)$$

- Update the hidden state h_t based on the cell state C_t:

$$h_t = O_t \times tanh(C_t) \quad (21)$$

2.4 Technical Indicators

Technical indicators such as SMA, EMA, RSI, and MACD are computed from closing prices, with lag variables capturing trends and patterns in the dataset. Data cleaning addresses missing values and outliers, and then, followed by normalization and scalling. The choice of machine learning models are xGboost and LSTM for subsequent model development in Google Colab environment.

2.4.1 Simple Moving Average (SMA)

A moving average is a statistical method used to smooth out fluctuations in time series data by averaging consecutive values. Smoothing time series filters out noise. Moving averages are utilized in finance to smooth stock price series and are vital in trend forecasting. It is basically implemented through weighted averages (Raudys & Pabarškaitė, 2018). This method computes averages for a subset of data points within a fixed interval of the entire dataset. Basically, it is valuable in determining the average price over a specified period of time, often focusing on the closing prices of stocks across a set number of days.

$$SMA = \frac{A_1 + A_2 \ldots A_n}{n} \quad (22)$$

Where: A_n represents the price of the asset in period n, n denotes the total number of periods. SMA helps to smooth out price data to identify the underlying trend. It averages prices over a specific period, making it easier to see overall direction. Again, SMA lines act as support or resistance whhich help traders make decisions about entry and exit points.

2.4.2 *Exponential Moving Average (EMA)*

The Exponential Moving Average (EMA) offers a good option to the Simple Moving Average (SMA) by emphasizing recent observations through weighted averaging. Its formula,

$$EMA_{Today} = Value_{Today} \times \left(\frac{Smoothing}{(Days+1)} + EMA_{Yesterday} \times \left(1 - \frac{Smoothing}{Days+1}\right)\right), \quad (23)$$

uses a smoothing factor to balance responsiveness and stability. EMA is computed over one less day than SMA, which reacts faster to market changes, leading to capturing short-term trends (Grebenkov & Serror, 2014). However, adjusting the smoothing factor requires caution to prevent bias. EMA helps in capturing short-term trends quickly.

2.4.3 *Relative Strength Index (RSI)*

The Relative Strength Index (RSI) is an important momentum indicator in technical analysis which gauges the strength or weakness of financial instruments. RSI readings above 70% imply overbought conditions, while readings below 30% indicate oversold conditions. Financial instruments with Relative Strength Index (RSI) scores above 50% are regarded as bullish, while those below 50% are deemed bearish. Computed over 14-day intervals, RSI evaluates potential shifts in trends by analyzing average gains and losses. This forecasting ability assists investors in selecting favourable moments to enter or exit positions by leveraging market sentiment to make informed trading choices (Gumparthi, 2017). The RSI is calculated through the following formulas:

1 First Step:

$$RSI = 100 - \left(\frac{100}{1+\left(\frac{Average_{Gains}}{Average_{Loss}}\right)}\right), where: \quad (24)$$

- Average_Gain is the average percentage gain of the financial instrument over the look-back period.
- Average_Loss is the average percentage loss of the financial instrument over the look-back period.

2 Second step:

$$RSI = 100 - \left(\frac{100}{1+\left(\frac{Previous_{Average_{Gains}} \times 12 + Current_{Gains}}{Previous_{Average_{Loss}} \times 12 - Current_{Loss}}\right)}\right) \quad (25)$$

RSI helps to identify potential reversal points and to determine whether the current movement is likely to continue or reverse.

2.4.4 *Moving Average Convergence Divergence (MACD)*

The moving average convergence divergence (MACD) is a momentum indicator which is used to spot the relationship between two exponential moving averages (EMAs) of a financial instrument's price. It is computed by subtracting the 26-day EMA from the 12-day EMA. The MACD line highlights the difference between these moving averages. Complemented by a signal line, typically a 9-day EMA of the MACD line, it provides insights into potential buying or selling actions (Aguirre et al., 2020). In contrast to the RSI, which focuses on overbought or oversold conditions, the MACD indicates upward trends when the MACD line crosses above the signal line and downward trends when it crosses below. Divergences between MACD and price can suggest potential trend reversals which can add an extra layer of analysis.

2.4.5 *MACD Signal*

The MACD signal line is derived from the Exponential Moving Average (EMA) of the MACD indicator over a typical 9-day period, and it serves as an important guide for traders in identifying potential buying or selling opportunities. When the MACD line crosses above the signal line, there is a bullish signal indicating an upward momentum shift and indicating a chance to purchase the instrument. In contrast, a bearish signal occurs when the MACD line falls below the signal line, which indicates a potential downward momentum and an opportunity to sell the instrument (Kang, 2021).

2.5 Implementation Processes

Once data is collected, technical indicators are computed from the closing prices to enhance the attribute set for the machine learning models. These indicators include but not limited to moving averages, RSI, MACD, and their respective lag features, were added to capture intricate patterns in the dataset. Furthermore, 5-day, 10-day, 15-day, and 30-day moving averages are computed to provide a better understanding of the stock datasets. Following the computation of these technical indicators, the data is meticulously prepared for the training of machine learning models through scaling and normalization. The overarching goal in this phase is to train two distinct machine learning models: LSTM and XGBoost. Consequently, there will be 2 distinct scenarios for model training and evaluation.

In the first scenario, XGBoost machine learning model is trained with the dataset. Meanwhile, in scenario 2, LSTM machine learning model is trained with the same dataset. The prediction results from both models are evaluated using a directional accuracy percentage (DAP) metric

to determine if there is significant difference in both models' ability to predict the direction of change in Microsoft Corp. stock prices.

The two models are constructed using a pipeline, integrating a range of hyperparameters like learning rate, maximum depth, n_estimators, and gamma. Leveraging a grid search algorithm on the training data helps to identify the optimal hyperparameter values. The model is then fitted and fine-tuned using the best parameters obtained from the grid search.

2.6 Increment Values

Evaluation metrics are computed on actual and predicted values using increment values to determine how the models fare in the prediction tasks. If absolute values for both actual and predicted exchange rates are x_i and $\hat{x}_i$ respectively, the corresponding increment values are $y_i := x_i - x_{i-1}$ and $\hat{y}_i := \hat{x}_i - x_{i-1}$. Hence, the decisions whether there is significant difference in both models‘ ability to predict the direction of change in Microsoft Corporation stock prices will be determined by metrics computed on increment values, which is: y_i and $\hat{y}_i$. While absolute values provide a measure of the magnitude of prediction errors, they are trivial and may not adequately capture the effectiveness of a model in terms of directional accuracy and practical decision-making. Increment values that focus on directionality are often preferred in such contexts for their ability to better reflect the model's performance in predicting the correct stock price movement.

2.7 Directional Accuracy Percentage (DAP)

DAP is a measure of the direction of accuracy of the predictions made by machine learning models. It is the ratio of the number of correct predictions to the total number of predictions multiplied by 100 (Giskard, 2024). DAP is computed on increment values, and higher DAP values are desirable.

$$DAP = \frac{number\ of\ correct\ predictions\ of\ direction\ of\ change}{total\ number\ of\ predictions} \times 100\% \qquad (26)$$

2.8 Mann-Whitney U Test

The Mann-Whitney U test, also referred to as the Wilcoxon rank sum test, is a test done to determine the differences between two groups on a single, ordinal variable with no particular distribution (Mann & Whitney, 1947; Wilcoxon, 1945). Mann-Whitney U test is a non-parametric statistical test used to determine whether two independent groups differ significantly in their distributions of a continuous variable. It requires a single variable to be

evaluated at the ordinal level, and to be normally distributed. It is a nonparametric test that assumes no specific distribution. Hence, the Mann-Whitney U, seeks to find out if two sampled groups are from a single population. When data failed the parametric assumptions requirement of the t-test, the Mann-Whitney U is deemed to be more appropriate (McKnight & Najab, 2010).

Here's a step-by-step outline of how the Mann-Whitney U statistic U is calculated:

1. Ranking the Data:
    - Combine the data from both groups into a single ranked dataset.
    - Rank all the observations from smallest to largest, assigning ranks without regard to which group they belong to.
2. Assigning Ranks:
    - Assign ranks starting from 1 for the smallest observation, 2 for the next, and so on, up to the total number of observations N.
3. Summing Ranks:
    - Calculate the sum of ranks R_1 for the first group.
    - Calculate the sum of ranks R_2 for the second group.
4. Calculating U:
    - Calculating U: Calculate the Mann-Whitney U statistic using the smaller of the two sums of ranks $U = min(U_1, U_2)$.
    - $U_1 = R_1 - \frac{n_1(n_1+1)}{2}$ (27)
    - $U_2 = R_2 - \frac{n_2(n_2+1)}{2}$ (28)

The Mann-Whitney U statistic U is then used to determine the significance of the difference between the two groups by comparing it to critical values from the Mann-Whitney U distribution. This test is particularly useful when analyzing ordinal or continuous data that does not meet the assumptions of normality or homogeneity of variances required by parametric tests like the t-test. Its implications extend beyond mere statistical significance, emphasizing the importance of understanding data characteristics, independence, and the context in which findings are applied. One of the primary advantages of the Mann-Whitney U test is its robustness to violations of normality assumptions.

2.9 Features Important Scores Derivation

The first step involves obtaining the LSTM layer from the model. This is done by accessing the first layer of the model using `model.layers[0]` by assuming the LSTM layer is positioned first in the model's architecture. Once the LSTM layer is retrieved, the next step is to obtain the weights associated with the connections between the input variables and the LSTM units. These weights indicate the significance of each feature in the model's predictions. Once the weights are retrieved, the code computes the feature importance scores. This is implemented by calculating the absolute sum of the weights across each feature dimension using `np.sum(np.abs(weights), axis=0)`. This operation adds together the importance of each feature across all LSTM units. To standardise the feature importance scores between 0 and 1 and ensure they sum to 1, the computed scores are divided by the total sum of all feature importance scores.

Whereas, in xGBoost, a DataFrame named `feature_importance_df` is created using the Pandas library. This DataFrame has two columns: 'Feature', which maintains the feature names, and 'Importance' which has the importance scores of each feature. `feature_names` is a list containing the names of these features, while `feature_importances` is a corresponding list containing their importance scores. To identify the top contributing attributes to the model's predictions, the `feature_importance_df` DataFrame is sorted based on the 'Importance' column in descending order. This sorts the features from most to least important. The `ascending=False` argument in the `sort_values()` function ensures this descending order is maintained.

3.0 Results

3.1 Scenario 1: XGBoost Models Empirical Result

The results from training the xGboost machine learning model using Microsoft Corporation stock price datasets are presented below. Twenty-three-year datasets are utilized to train the model. The xGBoost model underwent training on the training dataset using both default and adjusted parameter configurations and subsequently generated predictions for the test dataset. Technical indicators: SMA, EMA, RSI, MACD, and MACD signal, which were computed based on the closing rates, were integrated into the datasets. Simple moving averages (SMA) for 5, 10, 15, and 30 days, along with a 9-day period for the exponential moving average (EMA) used for the next day, were computed. Lag variables of N+1 and N+2 were calculated for each of these features, resulting in a total of 24 features utilized in the machine learning model's training. Directional accuracy percentage (DAP) performance evaluation metric was utilized

to measure the effectiveness of the model in accurately predicting the direction of change of Microsoft Corporation stock prices.

The initial configuration of model parameters after grid search is outlined in the Table 1

Table1: xGboost Parameters and Values.

Parameters	**Values**
n_estimators: Sets the number of boosting rounds, or trees, to build in the model.	1700
h max_depth: Sets the maximum depth of each tree.	3
Learning_rate: Sets to control the contribution of each tree to the final model.	0.15
Subsample: Specifies the fraction of samples to be used for each tree.	1
Gamma: Specifies the minimum loss reduction required to make a further partition on a leaf node.	0.0

The training dataset was scaled using StandardScaler() to scale the features used in training the model to ensure that after scaling, the features will have zero mean and a standard deviation of 1, resulting in a consistent dataset.

The process of splitting datasets into training and testing datasets followed a strategy based on monthly and yearly divisions. This approach was implemented to prevent the model from overfitting the testing dataset as this ensures the out-of-sample dataset is utilized in testing the model's predictive performance. Specifically, in the 23-year period datasets, the training dataset is composed of the first 249 months and the testing dataset is made up of the next 15 months. After running the code, the xGboost prediction model produces a directional accuracy percentage (DAP) metric value of 71.02% based on increment values. Corresponding feature score plot is also presented below.

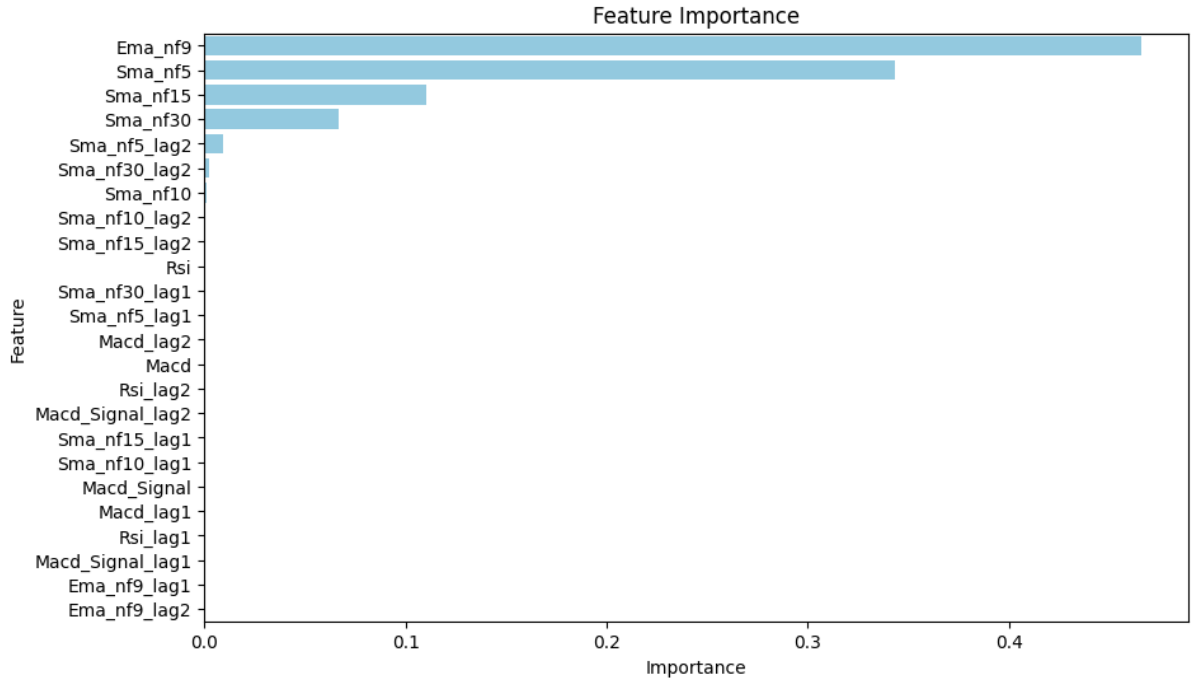

Figure 4: shows xGboost feature importance score

With respect to the importance of features utilized in training the xGboost model with the Microsoft Corporation dataset, the figure above shows that the exponential moving average of a 9-day period (Ema_nf9) emerges as the most significant feature of the xGboost machine learning prediction task as it ranked first among all features with an important score of 0.47. Following closely are the next top four features: Sma_nf5, Sma_nf15, Sma_nf15 and Sma_nf15_lag2. Simple moving averages mainly dominated the feature importance curve, implying a high correlation between closing rates and simple moving averages.

4.2 SCENARIO 2: LSTM MODELS EMPIRICAL RESULT

Similar to the first scenario, in the second scenario, the LSTM model underwent training on the training dataset with initial parameter configurations and generated predictions for the test dataset. Technical indicators: SMA, EMA, RSI, MACD, and MACD signal, were computed on the closing rates, and were added to the dataset. Simple moving averages (SMA) for 5, 10, 15, and 30 days, along with a 9-day period for the exponential moving average (EMA) used for the next day, were also calculated. Lag variables of N+1 and N+2 were computed for each of these variables, which resulted in a total of 24 features utilized in the LSTM machine learning. LSTM parameters were manually configured and are shown in Table 2

Table 2: LSTM parameters and values

Parameters	Values
Dropout probability: Controls the dropout rate during training, which is a regularization technique used to prevent overfitting.	0.1
LSTM units: Sets the number of LSTM units in the LSTM layer.	120
Optimization: Adapts the learning rate for each parameter, which can lead to faster convergence and better performance.	Adamax optimizer
Huber: This loss function is used for regression tasks and is less sensitive to outliers than mean squared error.	Loss
batch size: Sets the number of samples processed before the model's weights are updated, which can help with memory efficiency and speed.	6
Epochs: Sets the total number of complete passes through the training dataset to regulate training time.	100,000
save_best_only: Sets to retain the best-performing model while avoiding unnecessary storage of intermediate models.	True
Monitor: Sets to indicate that the validation loss should be monitored during training.	Val_loss
Mode: Set to indicate that the goal is to minimize the monitored quantity to ensure training process will look for the lowest validation loss.	Min
Patience: Sets how many epochs to wait for an improvement in validation loss before stopping training early.	8
restore_best_weights: Sets to ensure that the model restores the weights of the best epoch after training is complete, and this helps to retain the best performance.	True
Verbose: Sets to control the amount of output during training.	1

The training dataset was scaled using MinMaxScaler() to scale the features used in training the model to a specific range of 0–1, which ensures that all input features have a consistent scale. This can help the LSTM model converge faster during training.

The process of splitting datasets into training and testing datasets followed a strategy based on monthly and yearly divisions. This approach was implemented to prevent the model from overfitting the testing dataset as this ensures an out-of-sample dataset is utilized in testing the model's predictive performance. Specifically, in the 23-year period, the training dataset is

composed of the first 249 months, the testing dataset is made up of the next 15 months, and the 1-month dataset was used as a validation dataset to enable LSTM to compute val-loss. Once the code is executed, the LSTM machine learning model produces a directional accuracy percentage (DAP) metric value of 67.83% based on increment values. A corresponding feature score table for the top 5 features is also presented below.

Table 3: LSTM feature importance scores.

S/N	Features	Contribution scores
1	Sma_nf10_lag2	0.0060
2	Sma_nf30	0.0051
3	Sma_nf5_lag1:	0.0050
4	Ema_nf9	0.0046
5	Sma_nf5	0.0045

In the analysis, simple moving averages consistently showed the highest feature contribution scores when the LSTM machine learning model was trained on the dataset. This dominance is due to their strong correlation with the stock closing prices, which makes them valuable predictors in time series forecasting. Notably, during this training session, the 10-day simple moving average with a 2-day lag (referred to as Sma_nf10_lag2) emerged as the most significant feature influencing the model's predictions. This indicates that incorporating a slight lag allows the model to effectively capture trends and fluctuations in stock prices, enhancing its predictive accuracy. In contrast, the 9-day exponential moving average (Ema_nf9) saw a decline in its contribution, dropping to the fourth position in terms of feature importance. This shift suggests that while exponential moving averages are useful, their influence was overshadowed by the performance of the simple moving averages, particularly the Sma_nf10_lag2. The 2-day lag in the simple moving average might have provided the model with a more relevant temporal context, thus leading to its higher performance. These findings highlight the crucial role of moving averages in stock price prediction tasks, and they reveal the importance of feature selection in optimizing LSTM model performance. By understanding which features contribute most significantly, we can better refine our models for more accurate predictions in financial markets.

3.3 Predicted and Actual Values of xGBoost model plot

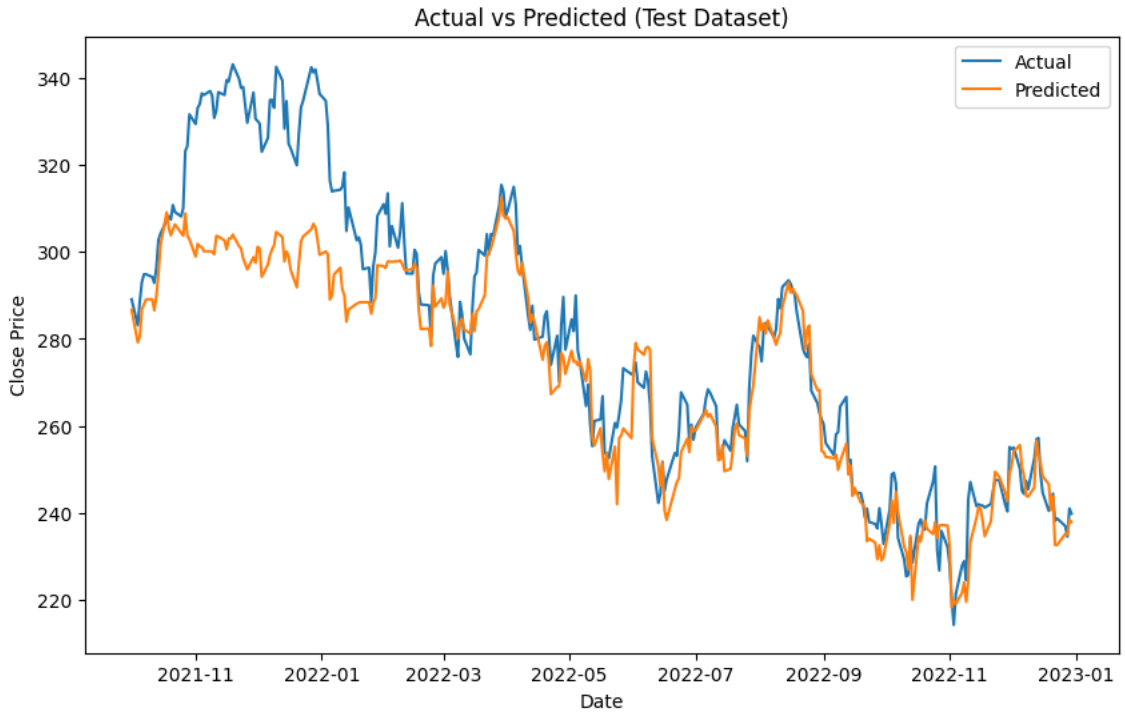

Figure 5: shows xGboost actual and predicted values.

The plot above (Figure 5) shows a consistent alignment between the forecasted closing stock prices and the actual closing stock prices, indicating that the predicted values effectively mirror the changes in trends of the Microsoft stock prices, despite a notable gap between actual and predicted values in the last two months of 2021. This alignment signifies a moderate fit of the model to the underlying dataset. XGboost was able to predict the direction of change of Microsoft stock prices more accurately when the closing rates were less volatile.

Nonetheless, there exists a slight lag between the actual stock prices and the corresponding predicted stock prices apart from the last two months of 2021. This lag points to the difficulties and challenges of capturing the underlying changes and trends in the ever fluctuating stock prices. However, the xGboost forecasting model was still able to predict the majority of the Microsoft stock price's direction of change correctly, and it also fared better than the LSTM machine learning model with a DAP of 71.02% compared to a DAP of 62.42% - 67.10% for the latter. The Mann-Whitney U test on both xGboost and LSTM predicted values returns a U Statistic value of 38653.0, a significance level (alpha) of 0.05 and P-value value of 1.6046912397672742e-06, which affirms and accepts alternate hypothesis which state that there is a significant difference in the accuracy of predicting Microsoft Corp stock price direction of change between the XGBoost and LSTM models. This is comparable to the

findings from the research done by Dey et al., in 2016, when they sought to accurately predict the trend of stock market using XGBoost which they similarly claimed to be an efficient algorithm with over 87% of accuracy. As the focus is on predicting increments rather than absolute values, the LSTM likely struggle with the variability and noise inherent in stock price changes, whereas XGBoost was able to manage these fluctuations more effectively with its ensemble approach.

4.0 Conclusion

This paper seeks to determine if there exists a significant difference between the xGboost model and the LSTM model when predicting the direction of change with respect to Microsoft stock's price based on increment values rather than absolute values. In the course of the research, a twenty-three-year period of Microsoft stock price dataset was collected from yahoo finance covering January 1st, 2000, to December 31st 2023.

Technical indicators were computed based on the dataset, and scaling was subsequently applied. Both xGboost and LSTM models were trained using scaled datasets, and predictions were made by the models. DAP, which is the ratio of the number of correct predictions to the total number of predictions, was computed based on increment values instead of absolute values to avoid the false impression that both models completely and correctly predict the direction of change of Microsoft stock prices. The XGboost model returns a DAP metric value of 71.02% in terms of accuracy. In contrast, the LSTM model returns DAP metric values within the range of 62.42% to 67.10% in terms of accuracy. The Mann-Whitney U test was computed on the predicted values of both models. It shows that there is a significant difference between xGboost and LSTM models' predicted values, and that xGboost performs better than LSTM when accurately predicting the direction of change of Microsoft stock prices.

In a nutshell, for investors, the implication is a potential tool enhancement for short-term decision-making based on xGboost's predictive superiority in directional changes of Microsoft stock prices. For both investors and scholars, the conclusion that xGboost performs better than LSTM in predicting the direction of change in Microsoft stock prices based on incremental values suggests a practical application of a sterner machine learning technique in financial forecasting. It underscores the potential for improving predictive accuracy in stock price movements, thereby influencing decision-making processes and contributing to improvement in predictive modelling research.

This paper focuses exclusively on the analysis of technical indicators for predicting changes in Microsoft stock prices. However, this may not sufficiently account for periods of extreme market volatility that could significantly influence stock prices, thereby limiting the robustness of the findings. Incorporating sentiment analysis could enhance the models by capturing the impact of public opinion. Additionally, the emphasis on DAP may not fully reflect model performance. Adding other metrics such as precision, recall, F1-score, and mean squared error would provide a more comprehensive evaluation of the models' effectiveness.

Future research efforts could be directed at developing hybrid models of both xGboost and LSTM techniques to improve accuracy in predicting the direction of change of stock price movement, with evaluation based on increment values. This can be done by using xGboost to extract features to train LSTM model prediction tasks. Combinatorial optimization could be applied to the hybrid model to improve prediction accuracy. This research paper simply focus on performance metric that only evaluate the directional accuracy of the prediction model. In future work, more performance evaluation metrics could be utilized to evaluate the general robustness of the prediction model.

References

Aguirre, A. A. A., Medina, R. A. R. and Méndez, N. D. D. (2020). Machine learning applied in the stock market through the Moving Average Convergence Divergence (MACD) indicator. *Investment Management & Financial Innovations,* 17(4),

Barrera-Animas, A. Y., Oyedele, L. O., Bilal, M., Akinosho, T. D., Delgado, J. M. D. and Akanbi, L. A. (2022). Rainfall prediction: A comparative analysis of modern machine learning algorithms for time-series forecasting. *Machine Learning with Applications*, 7, 100204.

Beg, M. O., Awan, M. N. and Ali, S. S. (2019). Algorithmic machine learning for prediction of stock prices. *In FinTech as a Disruptive Technology for Financial Institutions* (pp. 142-169). IGI Global.

Brownlee, J. (2018). Deep Learning for Time Series Forecasting. *Machine Learning Mastery.*

Chen, T. and Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. *In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).

Chindanur, N. B. and B, E. (2014). Prediction of selected Indian stock using a partitioning-interpolation based ARIMA-GARCH model. *Applied Computing and Informatics, 1*, Article 10.1016/j.aci.2014.09.002. https://doi.org/10.1016/j.aci.2014.09.002

Dey, S., Kumar, Y., Saha, S. and Basak, S. (2016). Forecasting to Classification: Predicting the direction of stock market price using Xtreme Gradient Boosting. *PESIT South Campus*, 1-10.

Elminaam, A., Salama, D., El-Tanany, A. M. M., El Fattah, M. A. and Salam, M. A. (2024). StockBiLSTM: Utilizing an Efficient Deep Learning Approach for Forecasting Stock

Market Time Series Data. International Journal of Advanced Computer Science & Applications, 15(4).

Forbes, (2024) *Top 10 largest companies by market cap*. Forbes India. Retrieved July 21, 2024, from https://www.forbesindia.com/article/explainers/top-10-largest-companies-world-market-cap/86341/1

Giskard AI. (2024). *Machine learning model accuracy*. Retrieved July 24, 2024, from https://www.giskard.ai/glossary/machine-learning-model-accuracy

Grebenkov, D. S. and Serror, J. (2014). Following a trend with an exponential moving average: Analytical results for a Gaussian model. *Physica A: Statistical Mechanics and its Applications,* 394, 288-303.

Gumparthi, S. (2017). Relative strength index for developing effective trading strategies in constructing optimal portfolio. *International Journal of Applied Engineering Research,* 12(19), 8926-8936.

Hochreiter, S. (1997). Long Short-term Memory. Neural Computation MIT-Press.

Hongjoong, K. I. M. (2021). MEAN-VARIANCE PORTFOLIO OPTIMIZATION WITH STOCK RETURN PREDICTION USING XGBOOST. *Economic Computation & Economic Cybernetics Studies & Research*, 55(4).

Hossain, S. and Kaur, G. (2024, May). Stock Market Prediction: XGBoost and LSTM Comparative Analysis. In 2024 3rd International Conference on Artificial Intelligence For Internet of Things (AIIoT) (pp. 1-6). IEEE.

Korstanje, J. (2021). ADVANCED FORECASTING WITH PYTHON : with state-of -the-art-models including lstms, facebook's... prophet, and amazon's deepar. Apress.

Kang, B. K. (2021). Improving MACD technical analysis by optimizing parameters and modifying trading rules: evidence from the Japanese Nikkei 225 futures market. *Journal of Risk and Financial Management,* 14(1), 37.

Kristanti, F. T., Febrianta, M. Y., Salim, D. F., Riyadh, H. A., Sagama, Y. and Beshr, B. A. H. (2024). Advancing financial analytics: Integrating XGBoost, LSTM, and Random Forest Algorithms for precision forecasting of corporate financial distress. Journal of Infrastructure, Policy and Development, 8(8), 4972.

Luo, J., Zhang, Z., Fu, Y. and Rao, F. (2021). Time series prediction of COVID-19 transmission in Amer-ica using LSTM and XGBoost algorithms. Results in Physics, 27, 104462. IEEE.

McKnight, P. E. and Najab, J. (2010). Mann-Whitney U Test. *The Corsini encyclopedia of psychology*, 1-1.

Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18, 50–60.

Olah, C. (2015). Understanding LSTM Networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Oukhouya, H., Kadiri, H., El Himdi, K. and Guerbaz, R. (2024). Forecasting International Stock Market Trends: XGBoost, LSTM, LSTM-XGBoost, and Backtesting XGBoost Models. *Statistics, Optimization & Information Computing*, 12(1), 200-209.

Raudys, A. and Pabarškaitė, Ž. (2018). Optimising the smoothness and accuracy of moving average for stock price data. *Technological and Economic Development of Economy*, 24(3), 984-1003.

Shi, Z., Hu, Y., Mo, G. and Wu, J. (2022). Attention-based CNN-LSTM and XGBoost hybrid model for stock prediction. arXiv preprint arXiv:2204.02623.

Wikipedia contributors. (2023, September 20). *History of Microsoft*. Wikipedia. https://en.wikipedia.org/wiki/History_of_Microsoft

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1, 80–83.

Xia, J., Pan, S., Zhu, M., Cai, G., Yan, M., Su, Q., Yan, J. and Ning, G. (2019, November 3). A Long Short-Term Memory Ensemble Approach for Improving the Outcome Prediction in Intensive Care Unit. *Computational and Mathematical Methods in Medicine*, 2019, 8152713. https://doi.org/10.1155/2019/8152713

XGBoost. (2024). XGBoost documentation. Retrieved from https://xgboost.readthedocs.io/en/latest/tutorials/model.html

Zhou, J. (2024). Predicting Stock Price by Using Attention-Based Hybrid LSTM Model. Asian Journal of Basic Science & Research, 6(2), 145-158.

Zhu, Y. (2023). Stock Price Prediction based on LSTM and XGBoost Combination Model. Transactions on Computer Science and Intelligent Systems Research, 1, 94-109.