Stock Price Prediction Using Time Series Analysis

Aymane Bokhamy, Khue Doan

Friday 22nd November, 2024

Abstract

Since the market is highly volatile and nonlinear, stock price prediction is still regarded as one of the most important activities in financial analysis. In view of this, this research work will develop a model for stock price prediction in the SP 500 using time series analysis based on past observations. The aim of this research will be to compare the performance of 2 forecasting techniques, ARIMA and LSTM for the most accurate model in SP 500 stock prediction. Our approach includes detailed data preprocessing, exploratory data analysis, and then the implementation followed by the evaluation of LSTM and ARIMA models for forecasting a chosen stock price. Preliminary results obtained from our analysis of the Apple Inc. stock, AAPL, show us the trend and correlation that will come in handy in model selection. These models will later be compared using measures such as MAE, RMSE, and MAPE; hence, this will provide an insight into the strengths and weaknesses of a certain technique. The aim of this research is to provide researchers and scientists with a detailed perspective of how the 2 models perform in stock prediction and which model of the 2 is more suitable for this type of task.

1 Introduction

In recent years, predicting stock market trends, particularly for influential indices like the SP 500, has garnered significant research interest due to its potential impact on investment strategies and economic forecasting. The SP 500, as a major index that tracks 500 large-cap U.S. corporations, serves as a crucial indicator of market health and economic performance. The advances in machine learning, more so in deep learning, have enabled the researchers to consider some sophisticated approaches to time-series forecasting, such as LSTM neural networks and the ARIMA model. Both models bear unique advantages and thus always are put to test for their efficiency in capturing market dynamics.

LSTM models have gained much popularity in stock forecasting due to the capability of capturing long-term dependencies in sequential data, which is a key factor in financial time series analysis. These models remember information over extended periods by using memory cells and gating mechanisms, making them well-suited for detecting complex, nonlinear patterns in stock price movements. For example, Wu, 2023, concluded that LSTM models outperformed traditional approaches like ARIMA in forecasting the SP 500 ESG Index, especially over longer forecast horizons, underlining thereby the model's suitability for capturing complex financial patterns. The work of Kamalov et al. [19] further demonstrated that recurrent neural networks, such as LSTM optimized with RMSprop, had quite low error rates in predicting the stock price, thereby showing that the networks had high predictive capabilities in modeling volatile behaviors of financial indices.

Despite the rising popularity of LSTM models, ARIMA remains a helpful tool in timeseries forecasting. It is particularly fit for short-term and linear trend prediction. Simple and interpretable, ARIMA is often preferred when stationary data is dealt with. Wu [18] demonstrated that ARIMA was very efficient in predicting stock prices; nonetheless, in longer-term forecasting, while performing well, its results proved to be less accurate than the results from an LSTM model, especially for the SP 500 ESG Index data set. Recent studies, on the other hand, by D'Amato, D' Ecclesia, and Levantesi within 2022 show that the integration of traditional econometric methods, such as ARIMA, with machine learning methods could provide better performance by embedding the strengths of both interpretability and predictive power. The dataset that has been proposed includes daily stock prices of companies listed in the SP 500 index and has been sourced from Kaggle [3]. In fact, this includes daily attributes of date, open price, high price, low price, close price, adjusted close price, and trading volume for each stock. We begin our process by preprocessing, cleaning missing values, and normalizing data. Further, we shall do EDA based on key trends, correlations, and patterns necessary for Apple's stock estimation. The investigation at hand is aimed at finding out which model gave the best prediction of stock prices. The performance

metrics that will be observed include Mean Squared Error (MSE), Root Mean Square (RMSE), and Directional Accuracy (DA). Such a comparison will help us get close to the best-performing model and provide insights about the strengths and weaknesses of each.

Various comparative studies underscore the versatility and strengths of deep learning models in stock prediction. For instance, C.K. Lee et al. [15] demonstrated that ARIMA outperformed backpropagation neural networks in forecasting the Korean stock price index, indicating the utility of traditional statistical models in certain conditions. However, recent studies highlight the competitive edge of deep learning models, such as Selvin et al.'s work [16], which found that CNN architectures effectively capture trend changes and outperformed other models in forecasting NSEI-listed stocks. Additionally, Yan and Ouyang's integration of wavelet transforms with LSTM models further boosted performance, outperforming traditional Support Vector Machines and K-nearest Neighbors [17].

In the context of stock time series forecasting, deep learning models, particularly the Long Short-Term Memory (LSTM) networks, have shown promising results due to their ability to capture temporal dependencies and long-term patterns.

Significant research supports that LSTM achieves superior results in time-series prediction. For instance, Shi et al. [9] enhanced LSTM with convolution for precipitation forecasting, demonstrating that this approach outperformed other existing precipitation models. Ma [10] applied LSTM to model nonlinear traffic dynamics in short-term traffic prediction, achieving top performance in both accuracy and stability. Additionally, Liu [11] employed LSTM for wind speed prediction, and Ding et al. [24] constructed a deep convolutional neural network to examine how event occurrences influence stock prices across different time horizons.

The study by Mehtab and Sen [13], for instance, demonstrates the power of using both CNN and LSTM architectures to forecast stock prices, specifically focusing on the NIFTY 50 index in the Indian stock market. Their findings suggest that a univariate encoder-decoder convolutional LSTM model using two weeks of prior data as input achieved the highest accuracy, due to its ability to capture temporal dependencies in time series while avoiding overfitting, which can be an issue in more complex LSTM models. This paper also reveals that while CNN models are faster and efficient, they may fall short in capturing sequential dependencies in financial time series compared to LSTM-based models.

Traditional LSTM networks have a simpler architecture compared to deeper stacked-LSTM models, which layer multiple LSTM units to theoretically enhance model complexity and capture more nuanced patterns. However, Zou and Qu (2020) [14] argues that the stacked-LSTM does not necessarily outperform a single-layer LSTM in this context, and in some cases, the simpler LSTM architecture achieves better performance. This result suggests that the additional complexity of stacked-LSTMs can lead to overfitting, making them less effective on unseen data compared to single-layer LSTMs, especially for continuous time-series prediction tasks like stock prices. It has also been suggested that stacked-LSTM models may perform better in classification tasks rather than continuous prediction tasks due to their structural complexity.

In addition to LSTM, our study involves ARIMA as well, which is also a statistical tool widely applicable in time series forecasting. ARIMA captures the linear temporal dependencies in data; hence, it would be especially suitable for short-term predictions of stock prices since the patterns are more stable there and less influenced by external factors. According to the study of Jiahui Wu on the SP 500 ESG Index, ARIMA showed its ability in treating non-stationary data with differencing techniques and hence is practical and interpretable for stock price prediction [18]. In a similar vein, Firuz Kamalov et al. stated that the ARIMA model, while being simpler than advanced neural networks, performs well when temporal patterns are predominantly linear. It demonstrated ARIMA giving robust performance for stock price prediction, emphasizing its simplicity and computational efficiency [19].

The key elements in the ARIMA methodology include autoregression, moving averages, and differencing. Optimal selection of parameters (p, d, q) by techniques such as autocorrelation and partial autocorrelation analysis ensures that ARIMA minimizes the prediction error. In this study, first-order differencing was performed on Apple stocks, which proved to stabilize non-stationarities verified by the Augmented Dickey-Fuller test. Then, application of ARIMA at order (1,1,1) represented reasonable accuracy, capturing most of the lagged relationships present in the data.

Application of ARIMA in stock prediction is supported by several studies. Mondal et al. reported that in the case of several industry stocks, ARIMA achieved accuracy above 85%, emphasizing its effectiveness while forecasting stable trends [18] Xiao et al. demonstrated the adaptive characteristics of ARIMA by comparing it with neural networks and finding it more suitable when there is a need for interpretability over complexity [18].

ARIMA thus plays the role of benchmark as it has a very straightforward nature with concentration

on modeling linear dependencies only. While it may not be as flexible in capturing complex nonlinear patterns compared to LSTM, ARIMA is computationally efficient and interpretable; it is highly reliable at short-term predictions and hence stands out in analysis related to financial time series. To complement that, our study has employed LSTM in stock market prediction for exploring its potential in the capturing of long-term temporal dependencies and further evaluating its performance across both simpler and more complex architectures. More specifically, we compare the performance of LSTM against traditional methods such as Exponential Moving Averages-a long-standing naive approach-used to assess the relative efficacy of deep learning techniques against simpler forecasting methods. This in-depth comparison will hopefully reveal several exciting insights into the strengths and weaknesses of each model, and it can form the bedrock of informed financial forecasting. This paper tries to extend those findings through the predictive performance of both LSTM and ARIMA models on one of the SP 500 Index stocks. We present in detail the whole data preprocessing and model building.

2 Methods

2.1 Dataset and Features

Our dataset consists of daily stock data for companies in the SP 500 index, one of the most widely recognized financial benchmarks globally. This index tracks 500 leading U.S. companies across various industries and provides a comprehensive view of the market. Our data includes several essential daily metrics: Date, Symbol, Adj Close, Close, High, Low, Open, and Volume. However, our target variable to forecast will be Adj Close, which represents the final closing price for the stock on that day.

First few rows of the dataset:

	Date	Symbol	Adj Close	Close	High	Low	Open	Volume
0	2010-01-04	MMM	44.016735	69.414719	69.774246	69.122070	69.473244	3640265.0
1	2010-01-05	MMM	43.741035	68.979935	69.590302	68.311035	69.230766	3405012.0
2	2010-01-06	MMM	44.361351	69.958191	70.735786	69.824417	70.133781	6301126.0
3	2010-01-07	MMM	44.393162	70.008362	70.033447	68.662209	69.665550	5346240.0
4	2010-01-08	MMM	44.705978	70.501671	70.501671	69.648827	69.974915	4073337.0

Figure 1: First Few Rows of the Dataset

First, we pre-process the dataset to prepare it for modeling. To do this, we filter the data on the stock symbol "AAPL," since our stock analysis is based on one chosen stock of the SP 500. Then we proceed to fill in the gaps in this dataset, an essential step in most time series analytics, as missing values tend to strongly disturb the performance of forecasting models. To deal with this, we used linear interpolation to approximate the missing value. In this technique, two points are utilized from the neighboring data in order to come up with a reasonable estimate for the lost value. Since changes in stock prices have a nature of gradualness day after day, it is an effective choice, which will preserve the integrity of the data. This approach ensures the data is kept continuous and avoids introducing artificial jumps or drops, which may otherwise distort the models' ability to detect patterns.

Date	Adj Close
2023-01-01	150.0
2023-01-02	152.0
2023-01-03	NaN
2023-01-04	155.0
2023-01-05	157.0

Date	Adj Close
2023-01-01	150.0
2023-01-02	152.0
2023-01-03	153.5
2023-01-04	155.0
2023-01-05	157.0

Figure 2: Dataset after Interpolation

One very useful figure to learn more about the features of our dataset would be the correlation matrix of our variables. In Figure 3, there is a strong positive correlation among the columns of stock prices such as Adj Close, Close, High, Low, and Open, which is expected since these features are interdependent

and for the most part will be tracked accordingly. However, there was a moderate negative correlation in trading Volume with respect to stock prices, which may indicate higher trading activity when prices fall. Because of the high level of correlations between price variables, we consider only the Adj Close price for our analysis, since it reflects the final price of the stock on that day with all market factors included. In addition to that, the dataset we use has a daily time unit, and the forecasting goal is to get the closing price for the next 20 days.

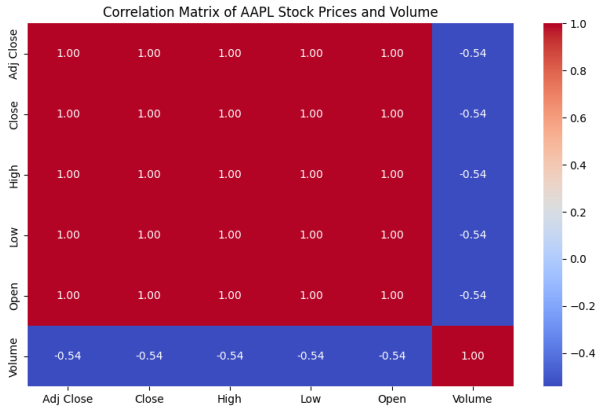


Figure 3: Correlation Matrix for AAPL Stock Prices

2.2 Exploratory Data Analysis (EDA)

To analyze the underlying trends and volatility within Apple's stock prices, we calculated and visualized the rolling mean and standard deviation over a 30-day window. The rolling mean will help to identify the long-term tendency of the stock, while the rolling standard deviation will provide insight with regard to the stock's volatility over time. In Figure 4, the red rolling mean closely follows the actual adjusted close prices shown in blue as the value of the stock of Apple rises gradually over the years (surges after 2020). At the same time, the green rolling standard deviation remains quite low (below $10), which reflects consistent price stability with periodic spikes that correspond to significant market events. This analysis is essential in uncovering probable structural changes in the stock prices and checking for stationarity, which is a prerequisite for ARIMA modeling.

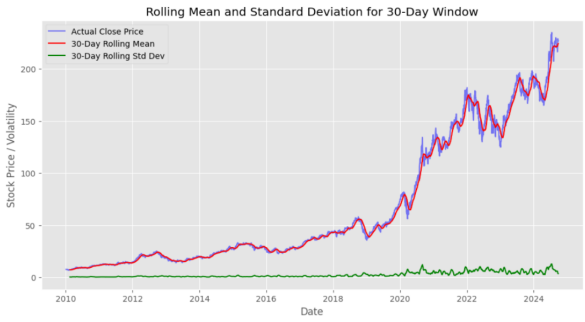


Figure 4: Rolling Mean and Standard Deviation for AAPL

Figure 5 demonstrates the sliding window approach used to prepare data for the LSTM model. The input sequence (blue line) represents 10 previous adjusted closing prices, while the prediction target (red point) corresponds to the next day's closing price. By using overlapping input sequences, the model captures sequential patterns in the stock price movements. This approach ensures that the model leverages temporal dependencies effectively to make predictions. The figure reflects recent Apple stock prices which showcases a realistic scenario where price movements over 10 days provide context for predicting the next day's price. This sequential framework highlights LSTM's strength in handling time-series data with inherent temporal dependencies.

After that, we plot a histogram of the first-order differences of Apple's adjusted closing prices to check the distribution of price changes and stationarity. Figure 6 shows that most of the price changes

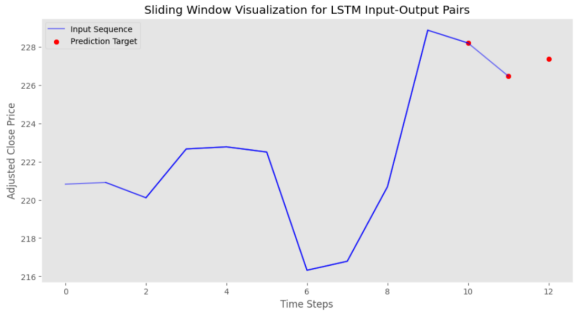


Figure 5: Sliding Window Visualization with Size 10

are concentrated around zero, in a roughly normal distribution with some skew. Such a pattern indeed suggests that the differencing process effectively removes the trend and make data more stationary. The long tails in both directions indicate occasional days with significant price jumps or drops, which could be due to market events, earnings reports, or other external factors, but for the most part the difference now is normal and centered around zero. Stationarity is important in any ARIMA model because it guarantees that the statistical properties-means and variances-are held constant over time to help the model provide forecasts reliably.

Figure 6: Distribution of Differenced Adj Closing Prices for AAPL

Next up, with our differenced data, we generate a lag correlation plot. It helps identify which past price changes influence current changes which guides the selection of input sequence length for LSTM modeling.

The plot reveals significant correlations at specific lags, such as lag 8, lag 9 and lag 18, which may indicate short-term dependencies in the stock price movements. Most lags, however, exhibit near-zero correlations, suggesting limited long-term influence.

Based on these findings, the input sequence for the LSTM model could focus on lags with meaningful correlations to capture relevant dependencies while minimizing noise from uncorrelated lags. Therefore, the recommended input sequence length ranges from lag 8 to lag 18.

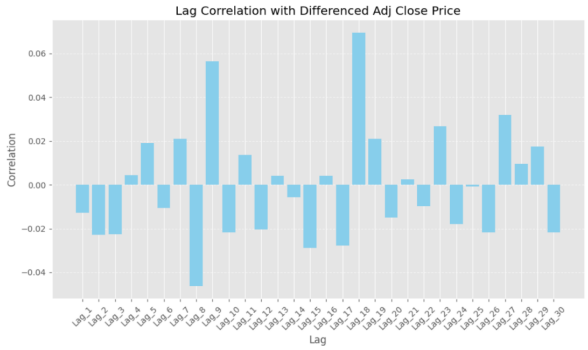


Figure 7: Lag Correlation for Differenced AAPL Prices

2.3 ARIMA and LSTM

In order to implement these two models, we need to undergo a series of tests and checks beforehand.

2.3.1 ARIMA Model

1. Checking for Stationarity

ARIMA models assume that the data is stationary, meaning that its statistical properties (like mean and variance) are consistent over time. In the context of financial time series, stationarity means that there are no trends or seasonal components affecting the data. Stationarity is crucial for ARIMA because non-stationary data can result in inaccurate or biased forecasts (Hayes, 2024).

To check for stationarity, we used the Augmented Dickey-Fuller (ADF) test. The ADF test helps determine whether a time series is stationary by testing for the presence of a unit root. The test produces a p-value:

- If the p-value is below a significance level (typically 0.05), we reject the null hypothesis that the series is non-stationary, indicating that the data is stationary.
- If the p-value is above the significance level, the data is considered non-stationary, and further steps are needed to make it stationary.

This test was done before the first-order differencing to our data.

```
ADF Statistic: -1.4679007509294606
p-value: 0.5493264343888273
Critical Value (1%): -3.4320580884979983
Critical Value (5%): -2.8622947397344367
Critical Value (10%): -2.567171757564323
```

Figure 8: ADF Statistics Before Differencing Data

Since the ADF test suggests non-stationarity, we proceed with differencing.

2. Differencing the Data:

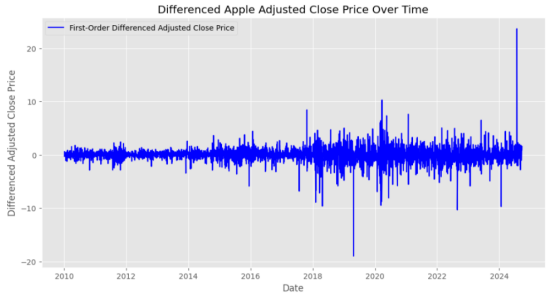


Figure 9: Differencing Data

Differencing is a technique used to transform non-stationary data into stationary data. By calculating the difference between consecutive observations, we can often remove trends or seasonality from the dataset. This process involves creating a new series where each value represents the difference between the current and previous values. In our analysis, we applied first-order differencing (subtracting the previous day's value from the current day's value) to remove any linear trends. If the data remains non-stationary after first-order differencing, higher-order differencing can be applied, although this is usually not required in financial data. In the figure above, the differenced series appears to fluctuate around a mean of zero, with relatively consistent variance for most of the time period. After differencing, we re-run the ADF test to confirm that the data has become stationary. This step ensures that the mean and variance of the differenced series remain constant over time, which is crucial for the ARIMA model's accuracy.

```
ADF Statistic: -19.329720786969297
p-value: 0.0
Critical Value (1%): -3.4320580884979983
Critical Value (5%): -2.8622947397344367
Critical Value (10%): -2.567171757564323
```

Figure 10: ADF Statistics After Differencing Data

Now, the ADF Statistic complies with stationarity requirements.

3. Identifying ARIMA Parameters (p, d, q):
The general form of an **ARIMA(p,d,q)** model can be written as:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t$$

Where:

- y_t: The actual value at time t.
- c: A constant or intercept term.
- $\phi_1, \phi_2, \ldots, \phi_p$: The **autoregressive coefficients**, which represent the relationship between the current value y_t and its previous values $y_{t-1}, y_{t-2}, \ldots$.
- $\theta_1, \theta_2, \ldots, \theta_q$: The **moving average coefficients**, which represent the relationship between the current value y_t and the previous error terms $\epsilon_{t-1}, \epsilon_{t-2}, \ldots$.
- ϵ_t: The error term (or residual) at time t.
- p: The number of autoregressive (AR) terms.
- d: The number of differencing steps required to make the series stationary.
- q: The number of moving average (MA) terms.

Once the data is stationary, we need to identify the best values for the ARIMA model parameters: p (autoregressive terms), d (degree of differencing), and q (moving average terms). The parameter selection is typically based on:

- Autocorrelation Function (ACF): This plot shows the correlation of the time series with its lagged values. The ACF plot helps in identifying the moving average (MA) term, q.
- Partial Autocorrelation Function (PACF): This plot shows the correlation of the time series with its lagged values, after removing the effects of shorter lags. The PACF plot helps in identifying the autoregressive (AR) term, p.

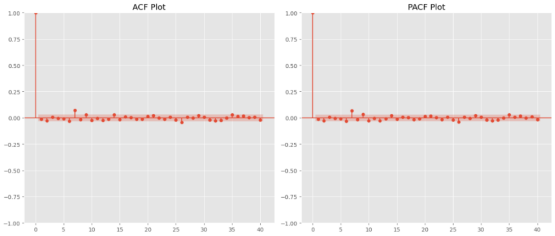


Figure 11: Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) Plots

Using the ACF and PACF plots, we can estimate p and q, while d is predefined as 1 due to the first-order differencing applied to achieve stationarity. The ACF plot shows the correlation between the series and its lagged values, while the PACF plot removes the influence of intermediate lags. Significant lags are identified by spikes exceeding the confidence interval (dashed horizontal lines). In the figure above, both plots show a sharp cutoff after lag 1, indicating p=1 (from the PACF plot) and q=1 (from the ACF plot) while d=1 because we applied a First-Order Difference to our data. This suggests that the ARIMA process requires minimal complexity, as no significant correlations exist beyond lag 1. In the context of stock prediction, this indicates that the model primarily relies on the most recent lag to predict future values, making it suitable for capturing short-term dependencies in stock price movements. However, ARIMA's reliance on immediate lags also means it may struggle to capture long-term trends or sudden, non-linear market shifts. This limitation highlights the importance of combining ARIMA with other methods, such as LSTM, to enhance the predictive power and account for more complex patterns inherent in stock price dynamics.
Therefore, according to these findings, ARIMA(1,1,1) is the most optimal.

5. Model Fitting and Evaluation:
Once the parameters p, d, and q are determined, we fit the ARIMA model to our training data. After fitting, we generate predictions for the test set and evaluate the model's accuracy using metrics such as Mean Squared Error (MSE) or Root Mean Squared Error (RMSE), and Directional Accuracy (DA). These metrics give us a sense of how well the model has captured the underlying patterns in the stock price data.

2.3.2 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks, a specialized type of recurrent neural network (RNN), are designed to handle long-term dependencies in sequential data by overcoming the vanishing gradient problem common in traditional RNNs [17]. This makes LSTMs particularly effective for time series forecasting tasks, such as stock price prediction, where historical data significantly influences future behavior.

LSTM Architecture An LSTM unit contains multiple gates but three are key:

- **Forget Gate**: Decides which information to discard from the previous cell state.
- **Input Gate**: Determines which new information to store in the current cell state.
- **Output Gate**: Regulates the output based on the updated cell state.

The gates operate as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad \text{(Forget Gate)} \tag{1}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad \text{(Input Gate)} \tag{2}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad \text{(Cell Candidate)} \tag{3}$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad \text{(Cell State)} \tag{4}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad \text{(Output Gate)} \tag{5}$$
$$h_t = o_t * \tanh(C_t) \quad \text{(Hidden State/Output)} \tag{6}$$

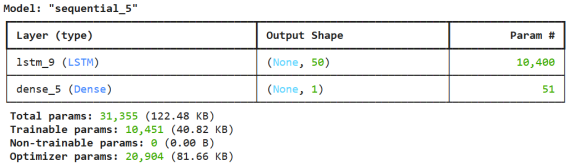
Where σ is the sigmoid activation function, and tanh is the hyperbolic tangent function. The weights (W) and biases (b) are learned during training.

This architecture allows LSTMs to retain important information over long sequences while selectively discarding irrelevant data, making them ideal for stock price prediction.

Single-Layer vs. Stacked-LSTM Models:

In our study, we build two LSTM configurations for our experiments

1. **Single-Layer LSTM**: A standard LSTM model with one LSTM layer followed by a Dense layer.



Model: "sequential_5"

Layer (type)	Output Shape	Param #
lstm_9 (LSTM)	(None, 50)	10,400
dense_5 (Dense)	(None, 1)	51

Total params: 31,355 (122.48 KB)
Trainable params: 10,451 (40.82 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 20,904 (81.66 KB)

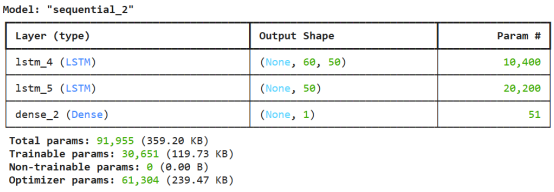
Figure 12: Single Layer LSTM Architecture Summary

The single-layer LSTM architecture is another simple yet efficient model for stock predictions. This architecture consists of a single layer of the LSTM, which processes sequences of input in the shape of '(None, 60, 1)'. In this, 'None' signifies batch size, 60 represents timesteps in sequence, and 1 is for one feature or stock closing price. This LSTM layer outputs a tensor of shape '(None, 50)', which compresses the input sequence into 50 high-level features that encapsulate the temporal dependencies and patterns in the stock price data.

Further down the line, after this LSTM layer, one dense, or fully connected, layer will take these 50 features as input and transform them into one output scalar: the predicted stock price. This more compact architecture eliminates some of the model's complexity, which could be prone to overfitting, particularly in cases of limited or noisy data. Finally, the computational cost is lower compared to a stacked LSTM model. Specifically, it has 31,355 total parameters and 10,451 trainable ones, against the total 91,955 parameters of the stacked LSTM.

Single-layer LSTMs are suitable for tasks that have a clear temporal pattern in the data and do not require deep hierarchical abstractions. This simplicity lets it generalize better and be more stable across different datasets, especially when there are less intricate relationships in data. Though it cannot capture the full range of nonlinear dynamics realized in more complex models, the single-layer LSTM represents a robust and efficient means of time-series forecasting, which balances interpretability, computational efficiency, and predictive performance.

2. **Stacked-LSTM**: A deeper model with two stacked LSTM layers, then followed by a Dense layer.



Model: "sequential_2"

Layer (type)	Output Shape	Param #
lstm_4 (LSTM)	(None, 60, 50)	10,400
lstm_5 (LSTM)	(None, 50)	20,200
dense_2 (Dense)	(None, 1)	51

Total params: 91,955 (359.20 KB)
Trainable params: 30,651 (119.73 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 61,304 (239.47 KB)

Figure 13: Stacked LSTM Architecture Summary

The architecture of the stacked LSTM in our analysis contains two LSTM layers, each adding one level of abstraction to the process of learning sequences. First, the LSTM layer takes input sequences of shape '(None, 60, 1)', where 'None' corresponds to the batch size, 60 represents the number of timesteps in the input sequence, and 1 indicates a single feature of the stock closing price. It outputs a tensor of the shape '(None, 60, 50)', which means that it processes every timestep in each sequence and generates 50 feature representations per timestep.

The output from the first LSTM layer is fed into a second LSTM layer, which further processes these representations, reducing the output to a shape of '(None, 50)'. This reduction summarizes the temporal information into a higher-level representation; in essence, it allows the network to learn complex patterns and long-term dependencies of the data. Finally, a dense (fully connected) layer transforms those 50 features into a single scalar output-the predicted stock price.

Compared with a single-layer LSTM, the stacked architecture provides more depth for the model to capture more sophisticated temporal dynamics and nonlinearities that are usually inherent in stock price data. However, this comes at a price, as added complexity also increases an already significant danger of overfitting on limited data, while the increased computational cost also comes into play due to extra parameters. For example, the total parameters in the stacked LSTM model are 91,955, out of which 30,651 are trainable, against the single-layered LSTM, which contains considerably fewer parameters.

On stock prediction, such a stacked architecture could capture the more complicated relationships driven by exogenous market factors and investor behaviors. However, for simpler scenarios where data is sparse or less-complex patterns occur, the single-layer LSTMs may perform better to avoid overfitting. The decision of whether to use a stacked or single-layer architecture depends on the particular dataset and objective of the predictions, bringing us to the trade-off between model complexity and generalization capability.

P.S: Both LSTMs are using a 60-day look-back period in order to make a prediction. This would ensure almost 2 months of historical data and information before making a prediction.

2.4 Evaluation Metrics (MSE, RMSE, DA):

To evaluate the predictive performance of ARIMA and LSTM models, we use three key metrics: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Directional Accuracy (DA).

2.4.1 Mean Squared Error (MSE)

MSE measures the average squared difference between actual values (y_i) and predicted values ($\hat{y}_i$), penalizing larger errors more heavily:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{7}$$

This metric is ideal for stock prediction as it amplifies larger deviations due to its squaring effect, which ensures models are penalized more for making significant errors in high-volatility financial data.

2.4.2 Root Mean Squared Error (RMSE)

RMSE, the square root of MSE, provides error magnitude in the same unit as stock prices:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{8}$$

It is particularly suited for stock forecasting because it emphasizes large prediction errors, which are critical in markets where minor mispredictions are tolerable but major inaccuracies can lead to significant financial losses.

2.4.3 Directional Accuracy (DA)

DA measures how often the model correctly predicts the direction of price changes:

$$DA = \frac{\text{Correct directional predictions}}{\text{Total predictions}} \times 100 \tag{9}$$

This metric is essential in stock prediction as it captures the model's ability to correctly anticipate upward or downward trends, which is mathematically significant in guiding investment decisions regardless of the exact price values.

These metrics collectively evaluate error magnitude (MSE, RMSE) and directional reliability (DA). Analyzing all three of them when discussing the results provides a comprehensive assessment of each model's performance.

3 Results

After splitting our dataset into train set (80%) and test set (20%), we obtain the following numbers for each set.

```
Train set: (3072,)
Test set: (769,)
```

Figure 14: Sets Sizes

We train our ARIMA(1,1,1) on our training set and then forecast the next 30 days.

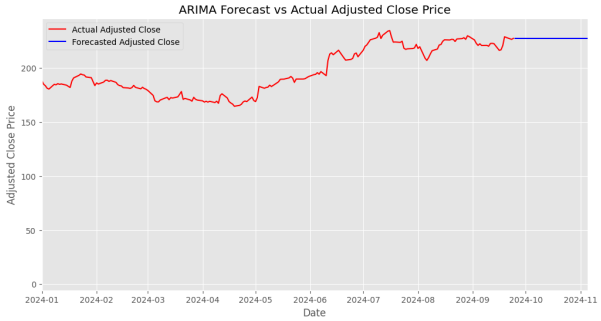


Figure 15: ARIMA(1,1,1) Forecasts on AAPL

After fitting the model on the training data, we used it to generate forecasts for the test set. The ARIMA model achieved a Mean Squared Error (MSE) of **975.0073**, Root Mean Square Error (RMSE) of **37.0811** and a Directional Accuracy of about **51.32%**. From the figure, and all of the evaluation metrics, we can see that our ARIMA model failed to capture any trend or seasonality in our dataset. The red blue representing the ARIMA predictions remains flat which indicates that the model was not able to adapt to the significant fluctuations and overall upward trend in the actual stock prices. This might be due to the parameters p,d,q that we set earlier. One thing to learn from this attempt is that more trials of p,d,q parameters are needed here. More details about this behavior in the discussion section.

Similar to ARIMA, we keep our splitting the same way and train our Single Layer LSTM on our training set. However, this time we will forecast a longer range than 30 days (2021 - 2024) and graph a comparison between the actual stock price and the LSTM predictions. We perform 10 epochs.

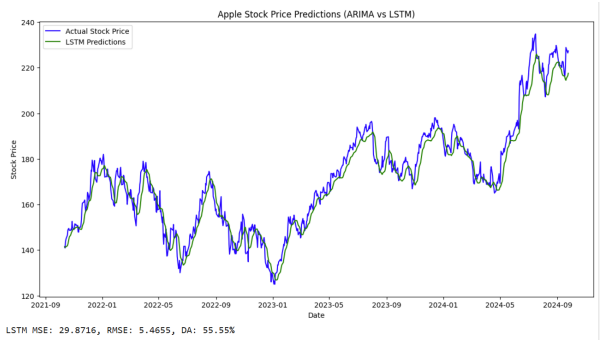


Figure 16: Single Layer LSTM Forecasts (60 days look-back) on AAPL with Evaluation Metrics

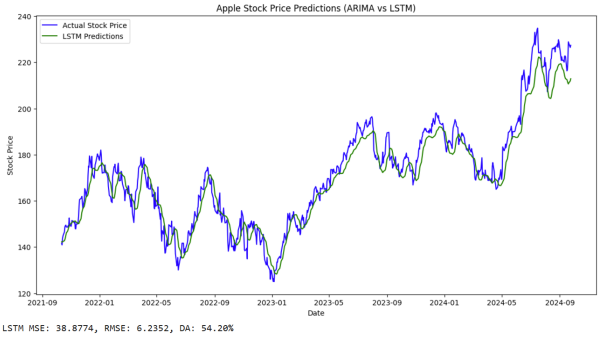


Figure 17: Stacked LSTM Forecasts (60 days look-back) on AAPL with Evaluation Metrics

The single layer LSTM demonstrated a very good predictive accuracy, as shown in Figure 16, with an MSE of **29.8716**, RMSE of **5.4655**, and a DA of **55.55%**. Being less complex, this architecture has proven its capability in capturing the temporal dependencies of Apple stock prices effectively, at least for a single-step-ahead prediction. Its simplicity will ensure that the risk of overfitting is reduced, something particularly important in noisy financial time series. However, the relatively modest DA suggests that though the model has been correct with respect to the magnitudes of the prices, its performance in correctly predicting the direction of movements can be further improved.

This is opposed to the stacked LSTM architecture, which consists of two LSTM layers and which therefore produces a slightly higher MSE and RMSE of **38.8774** and **6.2352**, respectively-compared to the single layer LSTM. Besides, it has a slightly lower DA of **54.20%**. The added layer contributes greater complexity to capturing more detailed patterns in stock price movements. However, this may have also made the model somewhat overfitting, considering the size of the dataset and high volatility in financial data. Though the stacked model might allow better performance in more complex data or when performing multi-step ahead predictions, the simpler single-layer LSTM was found here more reliable for one-step-ahead predictions in terms of finding a good trade-off between model performance and computational cost.

The Single-Layer LSTM performed better than the Stacked-LSTM model. While Stacked-LSTM was expected to capture more stochasticity in the stock market due to its complex structure, it struggled with overfitting and convergence issues, leading to worse performance. The simplicity of the Single-Layer LSTM allowed it to generalize better and adapt to the high stochasticity of financial time series.

Comparing the forecasts of the ARIMA(1,1,1) with Single Layer LSTM predictions, there are remarkable differences in the performance metrics and graphical trends. Looking at Figure 15, it is evident that the ARIMA model fails to mimic the inherent oscillations and volatile part of price changes, but rather kept on a flat overall trend in the Apple stock price over time. The flat-line nature of the ARIMA predictions, despite achieving an RMSE of **37.0811**, reflects an inability in capturing the underlying pattern of the data. Whilst the ARIMA provides simplicity and interpretability, its inability to handle nonlinearities and non-stationarities in the dataset is very evident and hence its relatively low DA was recorded at **51.32%**.

In contrast, Figure 17 shows that a Single Layer LSTM performed much better in tracking the temporal dependencies of Apple stock prices. The MSE reduced to as low as **29.8716** and RMSE of

5.4655. Unlike ARIMA, the LSTM model captures both the short-run and long-run movements in the data. Still, its DA of **55.55%** indicates that it has many areas of improvement for predicting the direction of price movements. Overall, the Single Layer LSTM demonstrates a distinct advantage over ARIMA in dealing with the series complexity, which is inherently nonlinear, in nature, thus making it a more robust choice for forecasting stock prices in this study. Further tuning of the parameters of the ARIMA model, or adding more features in the dataset that correlate with the price may reduce the gap in performance between the two models.

4 Discussions

4.1 ARIMA:

Various techniques were applied in this study in an effort to ensure an optimum performance of ARIMA with its weaknesses. The main problem with our ARIMA is that it is not able to capture the nonlinear and stochastic nature of financial markets very well. To address this problem, we dive into the parameters p,d,q selection of earlier in the process. This time, we use the auto_arima library that automatically chooses optimum parameters for the ARIMA model by minimizing information criteria such as AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion). We also change the forecasting range to cover the same range as the LSTM model (2021 - 2024) to be able to compare to the actual values.

The auto_arima function designated the following parameters ARIMA(5,1,0).

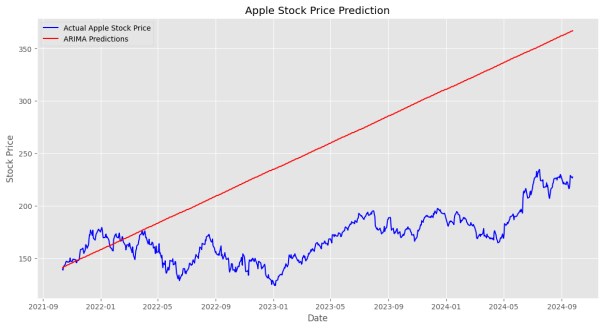


Figure 18: ARIMA(5,1,0) Forecasts on AAPL

The best model, ARIMA(5,1,0), chosen by the 'auto_arima' algorithm, is still too limited to catch the dynamics of the Apple stock price. This is clearly demonstrated in the graph, as the model yields a linear upward trend- red line. The observed behavior hints that the model, while identifying a general upward drift in the stock prices, misses the nonlinear and stochastic patterns in financial time series data.

So, even the best-fit model ARIMA(5,1,0) is not able to follow the fluctuations. Since, the best (p,d,q) combination parameters did not work as well, we must review the dataset's features. More features that correlate with the stock price should be added in the dataset in order to provide more room for the model to learn linear relationships between other predictor variables. Therefore, we can conclude from this that the ARIMA model is not efficient on a dataset with features about only the stock's different prices throughout the day. More information needs to be added to the dataset in order for the model to uncover relationships and trends between features. Such new columns in our dataset might include "Company Revenue", "Company Profit", or "Company Market Value" etc...

4.2 Single Layer LSTM:

The Single Layer LSTM model demonstrates a strong capability to predict Apple's stock prices over the test period, as evidenced by its close alignment with the actual stock prices in the corresponding graph. Unlike traditional models such as ARIMA, the Single Layer LSTM effectively captures both short-term fluctuations and broader market trends, highlighting its strength in learning temporal dependencies from sequential data. The model's evaluation metrics further reinforce its robustness, with an MSE of 29.87 and an RMSE of 5.46, indicating a relatively low level of error in predicting stock prices. The Directional

Accuracy (DA) of 55.55% suggests that while the model performs well in capturing the magnitude of price movements, there is room for improvement in accurately predicting the direction of these movements. I

In order to attempt to improve this model, we divide the look-back period by 2 to see if this improves the accuracy. It is critical to choose an optimal look-back period. Consequently, we get the following graph:

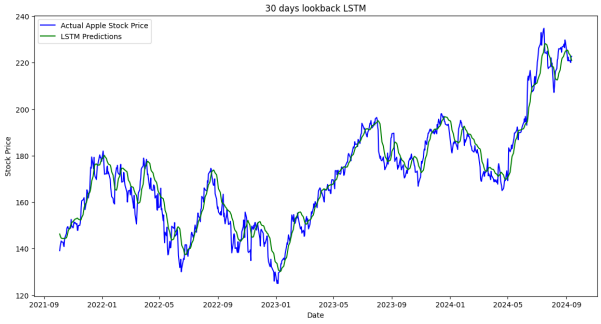


Figure 19: 30-Day Look-Back Single Layer LSTM Forecasts on AAPL

The lookback period, representing the number of past days used to predict future stock prices, was varied from 60 to 30 days. Our experiments revealed significant differences between the 2 look-back periods:

- **30-Day Lookback**: Delivered the best performance, with an MSE of **29.4925** and an RMSE of **5.4307**, and a DA of **56.04%**. This configuration effectively balanced capturing short-term fluctuations and broader market trends, producing predictions that are closely aligned with actual stock prices.
- **60-Day Lookback**: Produced an MSE of **29.8716** and an RMSE of **5.4655** and a DA of **55.55%**. While it tracked overall market trends adequately, it was less responsive to recent changes, leading to higher errors and reduced accuracy compared to the 30-day look-back model.

The superior performance of the 30-day lookback highlights the importance of selecting an optimal window size for financial forecasting, as shorter or longer periods can either miss critical trends or overfit historical noise. However, a 56.04% accuracy might sound better chances than flipping a coin; consequently, we need to compare the accuracy of our model to a naive approach. The chosen approach is the Exponential Moving Average Approach (EMA).

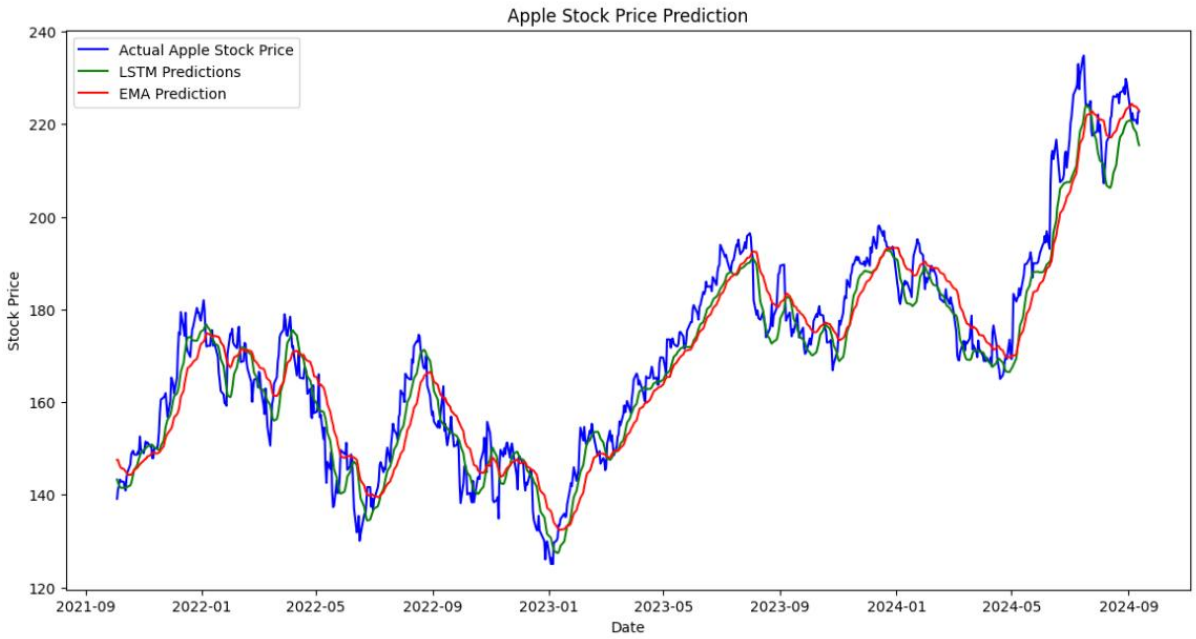


Figure 20: Comparison of EMA and 20-Day LSTM predictions against actual stock prices.

Exponential Moving Average (EMA) is a widely-used method in financial forecasting due to its simplicity and ability to capture trends by weighting recent data points more heavily. As discussed in the related works, EMA excels at smoothing noisy data and identifying general price trends but falls short in adapting to the stochastic and volatile nature of stock markets. While its simplicity makes it a staple in financial analysis, it struggles with responsiveness to sudden price changes and complex time-series patterns.

In our experiments, EMA achieved an MSE of **52.098**, an RMSE of **7.2179**, and a DA of **54.16%**, significantly higher than the 30-day LSTM's MSE of 29.4925 and RMSE of 5.4307. From Figure 20, it is evident that the EMA model (red line) lags behind the actual stock prices (blue line), particularly during periods of rapid market movement. This delay highlights the inherent limitations of EMA in capturing short-term fluctuations.

In contrast, the 30-day LSTM (green line) closely tracks the actual stock prices, effectively balancing responsiveness to short-term changes with the ability to model broader market trends. The LSTM's advantage stems from its ability to learn temporal dependencies in the data while leveraging its internal memory cells to retain and utilize information over a defined look-back period.

While EMA provides a general understanding of market trends, it lacks the adaptability and precision required for accurate stock price prediction in volatile markets. The 30-day LSTM, with its ability to learn from historical patterns, proves to be a more robust and reliable approach. This comparison demonstrates the value of incorporating advanced deep learning models like LSTM into financial forecasting tasks, where both precision and responsiveness are critical for success.

4.3 Conclusion

This study explored the application of ARIMA and LSTM models for stock price prediction, focusing on Apple Inc. stock as a case study. Our results demonstrate the superior performance of the 30-day Single-Layer LSTM model, which achieved the lowest error metrics (MSE = 29.4925, RMSE = 5.4307, DA = 56.04%), outperforming the Exponential Moving Average (EMA), ARIMA model, and Stacked LSTM. The LSTM model's ability to capture long-term dependencies and nonlinear patterns in sequential data makes it particularly effective in modeling the complex dynamics of stock prices. In contrast, the EMA model (naive approach), while useful for trend identification, struggled with responsiveness to rapid market changes and exhibited higher error rates (MSE = 52.098, RMSE = 7.2179). Similarly, while ARIMA provided interpretable results for short-term trends, it fell short in capturing fluctuations and stochastic behaviors inherent in financial markets.

The impact of these findings lies in their practical application. Accurate stock price predictions can support portfolio optimization, risk management, and strategic investment planning. The simplicity and

computational efficiency of the 30-Day Single Layer LSTM also make it accessible for integration into real-world financial systems, such as trading algorithms and market analysis platforms. However, the limitations were observed particularly in the accuracy (56.04%). Even though the model achieved a better prediction level than the naive approach EMA, it is still far from the trusted benchmark. This limitation may affect its utility for traders who rely heavily on directional signals.

For ARIMA, our results show that despite its simplicity and interpretability, it is ill-suited for capturing the nonlinear and stochastic behaviors inherent in financial markets. Even the best model ARIMA(5,1,0) failed to account for fluctuations in stock prices, as evidenced by its flat predictions and high error metrics (MSE = 975.00, RMSE = 37.08). This is why we concluded that more linear features need to be added to the dataset in order to for the model to understand relationships. This study relies solely on historical stock price data for predictions, omitting potentially influential factors such as macroeconomic indicators, market sentiment, or industry-specific variables. For instance, a metric about the company's yearly performance would be very informative. Incorporating these features could improve model performance and provide a more holistic view of stock price movements using ARIMA or LSTM. Second, the dataset used in this study focused on a single stock (Apple Inc.), which may limit the generalizability of our findings to other stocks or markets. Testing the model on a broader range of stocks would help evaluate its robustness and scalability.

Future research may remedy these drawbacks by investigating hybrid solutions integrating the respective benefits from LSTMs with traditional methods like ARIMA. Expanding the feature set beyond technical indicators to include news sentiment, trading volume, or global market indices may lead to increased prediction accuracy. Another direction worth exploring is the further development of explainable AI techniques to increase interpretability of the LSTM models-a weak point in financial applications.

In conclusion, this study supports the increasing popularity of LSTM models for stock price forecasting; it creates an opportunity for continuous innovation in financial modeling. Building upon the findings and controversies presented, future research can potentially bridge the gap between theoretical advancements in machine learning and their practical applications in the financial domain.

References

[1] Soni, P., et al. (2022). Machine learning approaches in stock price prediction: A systematic review. *Journal of Physics: Conference Series*, 2161(1), 012065. `https://doi.org/10.1088/1742-6596/2161/1/012065`

[2] Wu, J. M.-T., Li, Z., Herencsar, N., Vo, B., & Lin, J. C.-W. (2023). A graph-based CNN-LSTM stock price prediction algorithm with leading indicators. *Multimedia Systems*, 29(Special Issue), 1751–1770.

[3] Kaggle S&P 500 Stocks Dataset. `https://www.kaggle.com/datasets/andrewmvd/sp-500-stocks?select=sp500stocks.csv`

[4] Fischer, T., & Krauss, C. (2018). Deep Learning with Long Short-Term Memory Networks for Financial Market Predictions. *European Journal of Operational Research*, 270(2), 654-669.

[5] Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. *The American Statistician*, 72(1), 37-45.

[6] Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4), 679-688.

[7] Chatterjee, S., & Simon, D. (2015). Robust and computationally efficient parameter estimation techniques for robust control using RMSE. *IEEE Transactions on Instrumentation and Measurement*, 64(10), 2786-2794.

[8] Armstrong, J. S., & Collopy, F. (1985). Error measures for generalizing about forecasting methods: Empirical comparisons. *International Journal of Forecasting*, 8(1), 69-80.

[9] Xingjian, S. H., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., and Woo, W. C. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, pp. 802–810.

[10] Ma, X., Tao, Z., Wang, Y., et al. (2015). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, 54, 187–197.

[11] Liu, H., Mi, X., and Li, Y. (2018). Smart multi-step deep learning model for wind speed forecasting based on variational mode decomposition, singular spectrum analysis, LSTM network and ELM. *Energy Conversion and Management*, 159, 54–64.

[12] Ding, X., Zhang, Y., Liu, T., and Duan, J. (eds) (2015). Deep learning for event-driven stock prediction. In *International Conference on Artificial Intelligence*.

[13] Stock Price Prediction Using Convolutional Neural Networks on a Multivariate Time Series. (2020). In *Proceedings of the 3rd National Conference on Machine Learning and Artificial Intelligence*, February 2020, New Delhi, India, 7 pages. Published online: 15 Sep 2020.

[14] Zou, Z., and Qu, Z. Using LSTM in Stock Prediction and Quantitative Trading.

[15] Lee, C. K., Sehwan, Y., and Jongdae, J. (2007). Neural network model versus SARIMA model in forecasting Korean stock price index (KOSPI). *Issues in Information Systems*, 8(2), 372–378.

[16] Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., and Soman, K. P. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1643–1647. IEEE.

[17] Yan, H., and Ouyang, H. (2018). Financial time series prediction based on deep learning. *Wireless Personal Communications*, 102(2), 683–700.

[18] Wu, Jiahui. "S&P 500 ESG Index Prediction with LSTM and ARIMA Model." *Proceedings of the 2023 International Conference on Education, Management, Economics and Social Science (ICEMESS 2023)*, CSP, 2023, pp. 672–679.

[19] Kamalov, Firuz, Smail, Linda, and Gurrib, Ikhlaas. "Stock Price Forecast with Deep Learning." *arXiv preprint*, arXiv:2103.14081, 29 Mar. 2021. DOI: 10.1109/DASA51403.2020.9317260.

[20] D' Amato, V., D' Ecclesia, R., and Levantesi, S. "ESG Score Prediction through Random Forest Algorithm." *Computational Management Science*, vol. 19, no. 2, 2022, pp. 347–373.