Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Fachbereich Informatik,
Kommunikation und Wirtschaft
Studiengang FAR
HTW Berlin

Stock Price Prediction using Long Short-Term Memory (LSTM) and Large Language Models (LLM)

Paper for the Seminar

Deep Learning

by

Anastasiia Chekirda
Student-ID: 593923

submitted to

Dr. Alla Petukhina

February 20, 2025

Abstract

Stock market forecasting is a crucial task in financial analysis, requiring robust predictive models to manage price fluctuations. This study compares the effectiveness of two deep learning models — LSTM and Time-LLM — against the statistical ARIMA model, evaluating their performance using MAE, RMSE, and RMAE metrics. The analysis covered short-term (10–15 days), mid-term (30 days), and long-term (60–90 days) predictions. The findings indicate that while deep learning models generally outperform traditional statistical approaches, their accuracy heavily depends on stock volatility and the forecast duration.

Contents

List of Tables

List of Figures

1 Introduction and Motivation

Stock price prediction has long been an important area of financial research due to its potential for economic gains. Traditional statistical models, such as ARIMA (Box and Jenkins, 1970), have been widely used for time series forecasting. However, with recent advancements in machine learning, models like Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Large Language Models (LLMs) have emerged, introducing novel methodologies for time series forecasting.

LSTM networks, a specialized form of recurrent neural networks (RNNs), are designed to capture temporal dependencies by maintaining long-term memory of past observations (Hochreiter and Schmidhuber, 1997). They have demonstrated effectiveness of modeling sequential patterns, making them particularly suitable for stock price prediction (Fischer and Krauss, 2018).

Meanwhile, LLMs are deep learning algorithms capable of understanding and generating human language by processing large amounts of text data. These models utilize transformer architectures introduced by Vaswani et al. (2017)[1]. While LSTMs are specifically designed for structured numerical time series data, LLMs excel at understanding and generating human-like text. Therefore, LLMs can leverage textual information, such as financial news and sentiment analysis, which helps to improve predictive accuracy in financial forecasting (Onozo et al., 2024; Shen and Zhang, 2024).

This study aims to compare these two methodologies, evaluating the extent to which language-based models can handle time-series forecasting. Among the numerous approaches proposed in recent research on time series analysis with LLMs (Jiang et al., 2024), the Time-LLM model (Jin et al., 2024) has been selected for the experiment for its competitive performance.

To achieve the objective of the study, historical stock price data of the last 10 years

[1]The self-attention mechanism allows transformers to weigh the importance of different words in a sentence, regardless of their distance (Vaswani et al., 2017). Since their introduction, transformers have enabled the rapid scaling of language models, with architectures like GPT, LLaMA, and PaLM, containing hundreds of billions of parameters (Minaee et al., 2024).

is collected for five stocks representing different market conditions. The study evaluates the effectiveness of two deep learning models — LSTM and Time-LLM — in predicting future stock prices over various time horizons: short-term (10-15 days), mid-term (30 days), and long-term (60-90 days). Both models are benchmarked against statistical ARIMA model, and their performance is assessed using standard evaluation metrics, including Mean Squared Error (MSE) and Root Mean Squared Error (RMSE).

2 Literature Review

This chapter examines existing literature on the use of LSTMs and large language models for time-series forecasting, particularly applied for stock market predictions.

2.1 LSTM Model

LSTMs have been successfully applied to stock market forecasting due to their ability to capture long-term dependencies. Fischer and Krauss (2018) demonstrated that LSTM networks outperform traditional methods in predicting stock price movements, such as Random Forest (RF), deep neural network (DNN), and logistic regression classifier (LRC).

Jabed (2024) study examines the application of machine learning models, specifically LSTM, Facebook Prophet, and RF-Regressor, in stock price forecasting. The findings of the comparative analysis indicate that the LSTM model outperforms the other approaches in terms of predictive accuracy.

In other study, Ko (2021), incorporates market sentiment analysis into stock price forecasting by combining Bidirectional Encoder Representations from Transformers (BERT) and LSTM networks. BERT is utilized to assess sentiments from news articles and forum discussions, demonstrating a 12.05% improvement in RMSE over models that did not incorporate sentiment analysis. This underscores the effectiveness of integrating textual sentiment data from LLMs with numerical stock data processed by LSTMs to enhance prediction accuracy.

Despite their advantages, LSTM models can be computationally intensive and susceptible to overfitting when trained on limited data (Pang, 2024).

2.2 Large Language Models

Although originally designed for natural language processing (NLP) tasks, LLMs have been increasingly utilized in time series forecasting (Jiang et al., 2024). Time-LLM model, proposed by Jin et al. (2024), utilizes LLaMA and GPT-2 to align time-series data through

a reprogramming technique PaP (Prompt-as-Prefix). By transforming structured numerical input into a format compatible with LLM architectures, Time-LLM aims to enhance prediction accuracy without requiring significant modifications to the underlying transformer model. This methodology has been selected in this study for its competitive performance among other specialized forecasting models.

Gruver et al. (2024) introduced LLMTIME, which employs GPT-3 and LLaMA-2 to encode time-series data for zero-shot forecasting, meaning that the model performs without requiring extensive fine-tuning. LLMTIME integrates pre-trained language models with an encoding mechanism, allowing the model to infer patterns and trends directly from input sequences. This method offers advantages in terms of computational efficiency and adaptability to different datasets.

Chang et al. (2024) developed LLM4TS, a model that leverages both direct input alignment and task-specific fine-tuning. Using GPT-2, LLM4TS adapts LLM weights to better capture temporal dependencies, making it a more specialized solution for time-series forecasting tasks. The limitation is the increased computational cost associated with fine-tuning.

Finally, the Chronos model, proposed by Ansari et al. (2024), focuses on quantization-based techniques by tokenizing time-series sequences through a scaling mechanism. This approach is useful for reducing input complexity while retaining essential temporal features, thereby optimizing the LLM's ability to generalize across different time-series datasets. Reviewed literature and models have been summarized in Table 1.

Model	**Author**	**LLM**	**Key Characteristics**
Time-LLM	Jin et al., 2024	LLaMA, GPT-2	Aligning by reprogramming TS-data, PaP technique
LLMTIME	Gruver et al., 2023	GPT-3, LLaMA-2	Encoding and leveraging LLMs for zero-shot forecasting
LLM4TS	Chang et al., 2023	GPT-2	Combination of alignment and fine-tuning
Chronos	Ansari et al., 2024	GPT-2, T5	Quantization (tokenization of TS by scaling the data)

Table 1: Reviewed LLMs for Time-Series Forecasting (adapted from Jiang et al. (2024))

3 Data Description

This study employs financial time-series data from five stocks over the period from February 20, 2015, to February 19, 2025 (10 years). The data was sourced from Yahoo! Finance using the `yfinance` package in Python.

3.1 Dataset

The selected stocks listed in Table 2 represent a diverse range of sectors and market conditions, ensuring a well-balanced dataset. Apple Inc. (AAPL) and Microsoft Corporation (MSFT) are two of the largest technology companies, with significant market capitalization and influence on global stock indices. To incorporate different economic sectors and market dynamics, three additional stocks were selected: JPMorgan Chase & Co. (JPM), representing the financial sector; Tesla Inc. (TSLA), from the automotive industry; and Nvidia Corporation (NVDA), a leader in the semiconductor and artificial intelligence (AI) industry.

Stock Name	**Yahoo Finance Ticker**
Apple Inc.	`AAPL`
Microsoft Corporation	`MSFT`
JPMorgan Chase & Co.	`JPM`
Tesla Inc.	`TSLA`
Nvidia Corporation	`NVDA`

Table 2: Selected Stocks and their Yahoo Finance Ticker Symbols

3.2 Technical Indicators

Key technical indicators, adapted from the study of Basak et al. (2019), were computed to enrich the dataset with relevant market signals. These indicators include the Moving Average Convergence Divergence (MACD), Relative Strength Index (RSI), and On Balance Volume (OBV).

3.2.1 Moving Average Convergence Divergence

The MACD indicator (Appel, 2005) identifies momentum trends by computing the difference between two exponential moving averages (EMAs) of the stock price. It is defined as

$$\text{MACD}(t) = \text{EMA}_{\text{fast}}(t) - \text{EMA}_{\text{slow}}(t), \tag{1}$$

where $\text{EMA}_{\text{fast}}(t)$ and $\text{EMA}_{\text{slow}}(t)$ are EMAs over different time spans n, set to 12 and 26 days, respectively. The EMA is recursively computed as

$$\text{EMA}_n(t) = \alpha P_t + (1 - \alpha)\text{EMA}_n(t - 1), \tag{2}$$

where P_t is the closing price at time t and α is the smoothing factor, defined as

$$\alpha = \frac{2}{n + 1}. \tag{3}$$

Signal Line The Signal Line, which smooths fluctuations in the MACD, is derived by applying an EMA over a predefined window, set to nine days:

$$\text{Signal}(t) = \text{EMA}_9(\text{MACD}(t)). \tag{4}$$

This component helps to identify potential trend reversals.

MACD Histogram The MACD Histogram measures the difference between the MACD Line and the Signal Line, capturing trend strength:

$$\text{Histogram}(t) = \text{MACD}(t) - \text{Signal}(t). \tag{5}$$

A positive histogram value indicates bullish momentum, while a negative value suggests bearish movement.

3.2.2 Relative Strength Index

The RSI (Wilder, 1978) evaluates the magnitude of recent price changes to identify overbought or oversold conditions. It is computed as

$$\text{RSI}(t) = 100 - \frac{100}{1 + RS(t)}, \tag{6}$$

where $RS(t)$ represents the relative strength, given by

$$RS(t) = \frac{\text{Average Gain over } n \text{ periods}}{\text{Average Loss over } n \text{ periods}}. \tag{7}$$

In this study, n is set to 14 days. RSI ranges from 0 to 100 and indicates that the stock is overbought, when it is above 70, and oversold, when it is below 30.

3.2.3 On Balance Volume

The OBV indicator (Granville, 1963) is a cumulative momentum-based metric that measures the flow of volume in relation to price movements. It helps assess whether buying or selling pressure dominates the market. The OBV is computed as

$$\text{OBV}(t) = \text{OBV}(t-1) + \begin{cases} V_t, & \text{if } P_t > P_{t-1} \\ -V_t, & \text{if } P_t < P_{t-1} \\ 0, & \text{if } P_t = P_{t-1} \end{cases} \tag{8}$$

where V_t is the traded volume at time t, P_t is the closing price at time t, and $\text{OBV}(t-1)$ is the OBV value from the previous period.

The OBV increases when the closing price rises, indicating strong buying pressure, and decreases when the price falls, signaling selling pressure. When the OBV trend aligns with price movements, it confirms the strength of the trend.

3.3 Experimental Setup

The study employs multiple trading windows, ranging from 10 to 90 days. A trading window refers to a fixed time period of 10, 15, 30, 60, and 90 days over which data is analyzed to forecast future stock prices. The approach of experimenting with different time windows is adopted from the study of Basak et al. (2019). Each model is trained separately for each stock and across all time windows to evaluate the performance of the chosen algorithms — LSTM and Time-LLM — in mitigating posible overfitting caused by smaller windows and dealing with lag effects by larger time frames.

For training and testing, a manual time-based split is applied, ensuring that training and test sets follow a realistic chronological order. The first 70 percent of data are allocated for training, while the final 30 percent is used for testing.

3.4 Input Features & Prediction Target

In this study, the objective is to predict stock prices over the next 100 days using both historical closing prices and technical indicators, which are also computed based on historical price and volume data. The input features summarized in Table 3 include the stock's closing prices $P(t)$ at time t over a fixed trading window of length T, along with technical indicators calculated over the same trading window. A consistent set of features has been utilized for all the models models in the study to ensure a fair comparison.

The target variable is an output vector defined as

$$Y_t = \{P(t+1), P(t+2), P(t+3), \ldots, P(t+100)\}, \tag{9}$$

where $P(t+h)$ represents the stock price at time $t+h$, with h ranging from 1 to 100.

Feature	**Description**
$P(t)$	Closing price at time t
$P(t-1)$	Closing price at time $t-1$
$P(t-2)$	Closing price at time $t-2$
$\vdots$	Continuation of past closing prices
$P(t-T+1)$	Closing price at the start of the trading window $(t-T+1)$
$\text{EMA}_{12}(t)$	12-day Exponential Moving Average (EMA)
$\text{EMA}_{26}(t)$	26-day Exponential Moving Average (EMA)
$\text{MACD}_T(t)$	MACD Line: $\text{EMA}_{12}(t) - \text{EMA}_{26}(t)$
$\text{Signal}_T(t)$	Signal Line: 9-day EMA of $\text{MACD}_T(t)$
$\text{Histogram}_T(t)$	MACD Histogram: $\text{MACD}_T(t) - \text{Signal}_T(t)$
$\text{RSI}_T(t)$	Relative Strength Index (RSI)
$\text{OBV}_T(t)$	On-Balance Volume (OBV)

Table 3: Overview of Input Features

4 Deep Learning Algorithms

4.1 LSTM

LSTM networks (Hochreiter and Schmidhuber, 1997) are a specialized type of recurrent neural network (RNN) designed to address the limitations of traditional RNNs, particularly the vanishing gradient problem. Unlike standard RNNs, which struggle to maintain long-term dependencies, LSTMs incorporate memory cells that regulate information flow through a gating mechanism. These mechanisms allow the network to selectively retain or forget information over long sequences, making LSTMs effective for time-series forecasting and sequential data modeling.

4.1.1 LSTM Memory Cell

An LSTM unit consists of three gates: the forget gate, the input gate, and the output gate. These gates regulate the flow of information into, within, and out of the memory cell. Given an input sequence $X = (X_1, X_2, ..., X_T)$, where each $X_t \in \mathbb{R}^d$ represents a feature vector at time step t, the LSTM operations are formally defined as follows.

The *forget gate* determines the extent to which the previous cell state should be retained:

$$f_t = \sigma(W_f X_t + U_f h_{t-1} + b_f). \tag{10}$$

The *input gate* regulates the incorporation of new information into the memory cell:

$$i_t = \sigma(W_i X_t + U_i h_{t-1} + b_i). \tag{11}$$

A candidate memory update is computed using a transformation of the input and previous hidden state:

$$\tilde{C}_t = \tanh(W_c X_t + U_c h_{t-1} + b_c). \tag{12}$$

The memory cell state is then updated by combining the previous state and the new

candidate values:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t. \tag{13}$$

The *output gate* determines how much of the cell state contributes to the hidden state at the current time step:

$$o_t = \sigma(W_o X_t + U_o h_{t-1} + b_o). \tag{14}$$

Finally, the hidden state is computed as

$$h_t = o_t \odot \tanh(C_t). \tag{15}$$

In the previous equations, W_f, W_i, W_c, W_o and U_f, U_i, U_c, U_o represent weight matrices, while b_f, b_i, b_c, b_o are bias terms. The function $\sigma(\cdot)$ denotes the sigmoid activation function, $\tanh(\cdot)$ represents the hyperbolic tangent function, and $\odot$ signifies element-wise multiplication (Arras et al., 2021; Waqas and Humphries, 2024).

4.1.2 Prediction and Training

Given an input sequence $X = (X_1, X_2, ..., X_T)$, the model computes the final hidden state representation h_T. The predicted output is calculated as follows:

$$\hat{y} = \sigma(W_y h_T + b_y), \tag{16}$$

where W_y and b_y are the weight matrix and bias term of the output layer, respectively. The function $\sigma(\cdot)$ represents the sigmoid activation function, which is defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{17}$$

For regression task, the model is trained using the Mean Squared Error (MSE) loss function, which quantifies the difference between the predicted stock price $\hat{y}$ and the actual stock price y. The loss function is defined as

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{18}$$

where n denotes the total number of samples. The model parameters are optimized using gradient-based techniques such as the Adam optimizer.

4.1.3 LSTM Implementation

The LSTM model is implemented using the `NeuralForecast` library from Nixtla[2], where a multi-layer perceptron (MLP) decoder is used to transform the hidden representations, obtained from the LSTM encoder, into final predictions. By using activation functions such as the rectified linear unit (ReLU), the model introduces non-linearity, which enhances its ability to model dynamic stock price movements. Fully connected dense layers allow the model to integrate both temporal dependencies captured by the LSTM and external features, such as technical indicators. To ensure numerical stability and improve convergence during training, the input features are standardized using a `'standard'` scaler. The model architecture, illustrated in Figure 1, consists of an LSTM encoder with 128 hidden units. Following the encoding stage, the model includes two fully connected layers in the decoder, each with 128 hidden units.

4.2 Time-LLM

Tme-LLM, introduced by Jin et al. (2024), is a reprogramming framework designed to adapt LLMs for time series forecasting tasks without modifying the backbone model. The key aspect of this approach is that time-series data can be reformulated into a textual format, allowing pre-trained LLMs to use their knowledge and self-attention mechanisms for predictive modeling.

Time-LLM model operates through three key stages: (1) input transformation, where raw time-series data is normalized, segmented into overlapping patches, and embedded into a high-dimensional space; (2) pre-trained and frozen LLM processing, where the restructured

[2] `https://nixtlaverse.nixtla.io/neuralforecast/models.lstm.html`

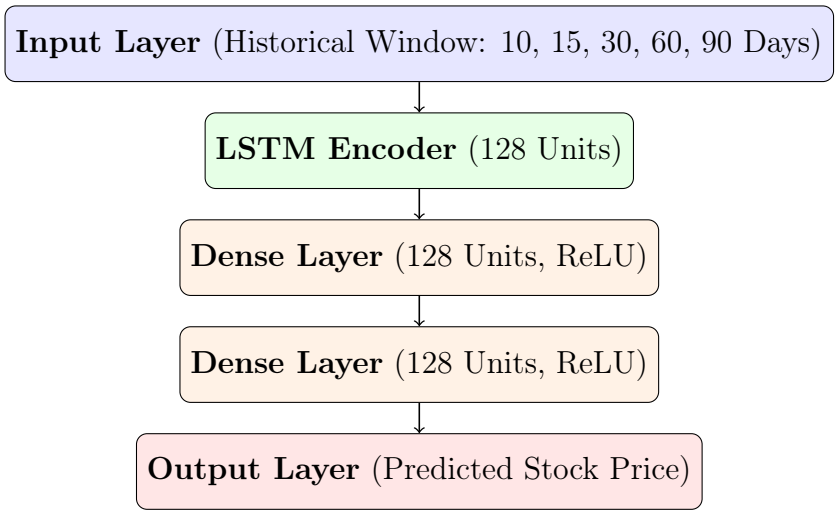

Figure 1: LSTM Model Architecture

input is fed into a fixed LLM alongside a structured prompt; and (3) output projection, where the LLM's embeddings are mapped back into the time-series domain through a learnable projection layer to generate future predictions.

4.2.1 Input Transformation

Given a multivariate time series $X \in \mathbb{R}^{N \times T}$, where N denotes the number of variables and T represents the time steps, the goal is to predict future values $\hat{Y} \in \mathbb{R}^{N \times H}$ over a horizon H.

Normalization Each univariate series $X^{(i)}$ is individually normalized to have zero mean and unit variance using Reversible Instance Normalization (RevIN) (Kim et al., 2022):

$$X^{(i)}_{\text{norm}} = \frac{X^{(i)} - \mu^{(i)}}{\sigma^{(i)}}, \tag{19}$$

where $\mu^{(i)}$ and $\sigma^{(i)}$ are the mean and standard deviation of $X^{(i)}$ respectively.

Patching The normalized series is divided into P overlapping patches of length L_p with stride S:

$$P = \left\lfloor \frac{T - L_p}{S} \right\rfloor + 2. \tag{20}$$

Given the patch representation $X_P^{(i)} \in \mathbb{R}^{P \times L_p}$, the patches are embedded into a higher-dimensional space as $\hat{X}_P^{(i)} \in \mathbb{R}^{P \times d_m}$, using a simple linear layer (Nie et al., 2023) as the patch embedder to project them into dimensions d_m.

4.2.2 Time-Series Patch Reprogramming

Given the embedded time series patches $\hat{X}_P^{(i)} \in \mathbb{R}^{P \times d_m}$, reprogramming is performed using the embeddings $E \in \mathbb{R}^{V \times D}$, where V represents the vocabulary size and D is the hidden dimension of the backbone model.

A multi-head cross-attention mechanism aligns the time series patches with the LLM's feature space. For each attention head $k = \{1, \ldots, K\}$, the query, key, and value matrices are defined as:

$$Q_k^{(i)} = \hat{X}_P^{(i)} W_Q^k, \quad K_k^{(i)} = E' W_K^k, \quad V_k^{(i)} = E' W_V^k, \tag{21}$$

where $W_Q^k \in \mathbb{R}^{d_m \times d}$ and $W_K^k, W_V^k \in \mathbb{R}^{D \times d}$, with $d = \lfloor d_m / K \rfloor$ representing the dimension of each attention head.

The reprogramming operation for time-series patches follows the multi-head attention mechanism:

$$Z_k^{(i)} = \text{Attention}(Q_k^{(i)}, K_k^{(i)}, V_k^{(i)}) = \text{Softmax}\left(\frac{Q_k^{(i)} {K_k^{(i)}}^\top}{\sqrt{d_k}}\right) V_k^{(i)} \tag{22}$$

with $Z^{(i)} \in \mathbb{R}^{P \times d_m}$.

4.2.3 Output Projection

Finally, a linear projection aligns the hidden dimensions with the backbone model, resulting in

$$O^{(i)} \in \mathbb{R}^{P \times D}, \tag{23}$$

where $O^{(i)}$ denotes the LLM's output embeddings. The Time-LLM model architecture is illustrated in Figure 2.

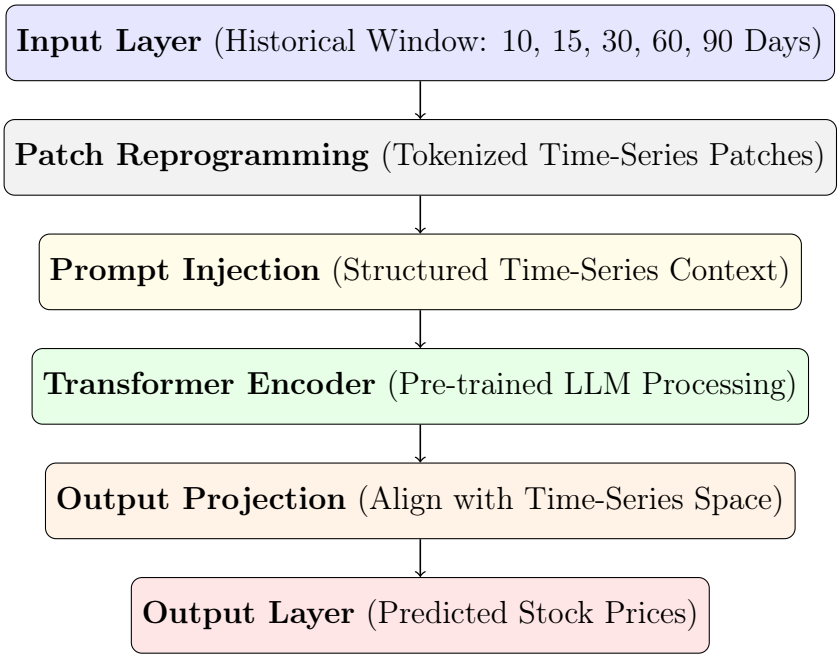

Figure 2: Time-LLM Model Architecture based on Jin et al. (2024)

4.2.4 Time-LLM Implementation

The Time-LLM model is implemented using the `NeuralForecast` library by Nixtla[3]. The model is configured with a set of hyperparameters to optimize its forecasting performance. A key component is the `prompt_prefix`, which provides structured contextual information that the dataset contains stock prices and technical indicators such as MACD, OBV, and RSI. This prompt aids the model in capturing domain-specific patterns within financial data. The training process is structured around a batch-based optimization approach, where the `batch_size` and `valid_batch_size` are both set to 16, ensuring consistency between training and validation phases. To control the learning dynamics, the model is trained over a maximum of 150 iterations, specified by the `max_steps` parameter. Additionally, the `windows_batch_size` is set to 16, determining the number of sliding windows processed per

[3] `https://nixtlaverse.nixtla.io/neuralforecast/models.timellm.html`

batch, which allows the model to efficiently extract sequential dependencies from historical stock prices.

5 Results

5.1 Performance Metrics

The Mean Absolute Error (MAE) measures the average absolute difference between actual and predicted stock prices. It is formally defined as

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{24}$$

where n denotes the total number of time steps in the sequence, y_i represents the actual stock price at time step i, and $\hat{y}_i$ corresponds to the predicted stock price at the same time step. A lower MAE indicates a higher prediction accuracy.

The Root Mean Squared Error (RMSE) evaluates the discrepancy between actual and predicted values. RMSE is computed as

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}. \tag{25}$$

Similar to MAE, RMSE considers the difference between actual and predicted values. However, by squaring the errors before averaging, RMSE penalizes larger deviations more heavily than MAE. A lower RMSE value indicates a more precise forecasting model.

While MAE and RMSE provide absolute error measures, they do not account for the scale differences across different stocks. Stocks with higher price may have larger absolute errors compared to lower-priced stocks, making direct comparisons between assets challenging. To address this, the Relative Mean Absolute Error (RMAE) is additionally computed, which normalizes absolute errors by the average stock price and expresses the error as a percentage of the stock's average price. RMAE is defined as

$$RMAE = \frac{MAE}{\frac{1}{n}\sum_{i=1}^{n} y_i} \times 100. \tag{26}$$

5.2 LSTM Performance Evaluation

Table 4 provides MAE and RMSE values for LSTM model over multiple prediction intervals (10, 15, 30, 60, and 90 days) across various stock tickers. For Apple Inc. (AAPL), the model demonstrates a reduction in MAE as the prediction horizon extends up to 30 days, followed by an increase at 60 days and a subsequent decline at 90 days. A similar trend is observed for Microsoft Corporation (MSFT), where relatively low MAE is recorded at the 30-day horizon. This pattern suggests that the model achieves optimal accuracy in the medium term.

The model exhibits relatively consistent performance across different time horizons for JPMorgan Chase & Co. (JPM), with the lowest errors recorded at 60 days. This outcome suggests that the LSTM model aligns effectively with historical trends for this stock. For Tesla Inc. (TSLA), the significantly high MAE and RMSE values indicate substantial difficulty in predicting TSLA's stock prices. The error metrics peak at 30 days, underscoring the challenges associated with forecasting highly volatile stocks. In contrast, the model achieves relatively stable errors for NVIDIA Corporation (NVDA) across different time horizons.

Table 5 and Figure 3 present the RMAE values, enabling a comparative evaluation of LSTM's performance. The lowest RMAE for Apple Inc. (AAPL) is observed at 90 days (4.80), suggesting enhanced relative performance for long-term forecasts. For Microsoft Corporation (MSFT), a decline in RMAE from 10 to 30 days is evident, followed by a significant increase at 60 and 90 days, indicating the model's difficulty in medium-to-long-term predictions. JPMorgan Chase & Co. (JPM) exhibits the lowest RMAE (3.46) among all stocks at 60 days. In contrast, TSLA's and NVDA's RMAE remain relatively high across all time horizons, signifying the model's struggle in forecasting their volatile price dynamics.

Overall, the results indicate that short- and medium-term forecasts, particularly 10 and 30 days, generally yield the lowest errors for most stocks, whereas longer-term predictions (60 and 90 days) introduce increased uncertainty.

Ticker	10 Days		15 Days		30 Days		60 Days		90 Days	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
AAPL	12.03	13.47	10.67	12.23	9.59	10.71	19.40	20.88	7.72	9.01
MSFT	21.45	25.61	20.02	23.88	19.11	21.92	32.77	37.51	17.40	20.93
JPM	7.47	8.69	8.16	9.38	9.87	10.93	4.46	5.25	9.33	10.59
TSLA	45.06	49.63	47.15	51.76	53.46	56.86	28.08	33.50	45.03	50.43
NVDA	6.48	7.06	6.22	6.84	5.72	6.48	7.57	8.17	5.47	6.14

Table 4: LSTM Absolute Errors MAE and RMSE

Ticker	10 Days	15 Days	30 Days	60 Days	90 Days
AAPL	7.49	6.64	5.97	12.08	4.80
MSFT	7.63	7.12	6.80	11.65	6.18
JPM	5.80	6.33	7.65	3.46	7.24
TSLA	24.77	25.91	29.39	15.43	24.75
NVDA	24.04	23.09	21.24	28.07	20.31

Table 5: LSTM Relative Errors RMAE as % of Stock's Average Price

5.3 Time-LLM Performance Evaluation

Table 6 provides MAE and RMSE values for Time-LLM model over different forecast horizons (10, 15, 30, 60, and 90 days) across various stock tickers. For Apple Inc. (AAPL), the model demonstrates an increase in MAE as the forecast horizon extends, peaking at 60 days. The error values slightly decline for longer horizon, indicating that the model captures short-term price movements but performs worse over extended forecasts. A similar trend is observed for Microsoft Corporation (MSFT), where errors rise with the increase of trading window. This pattern suggests that while the model can predict short-term price movements with reasonable accuracy, its performance declines as the prediction horizon extends.

For JPMorgan Chase & Co. (JPM), the MAE and RMSE values are consistently lower compared to other stocks, indicating relatively stable and predictable price movements. The lowest errors are observed at 60 days, suggesting that the Time-LLM model works well with historical price trends for this stock, which aligns with the results of the LSTM model.

In contrast, Tesla Inc. (TSLA) exhibits significant fluctuations in error values, par-

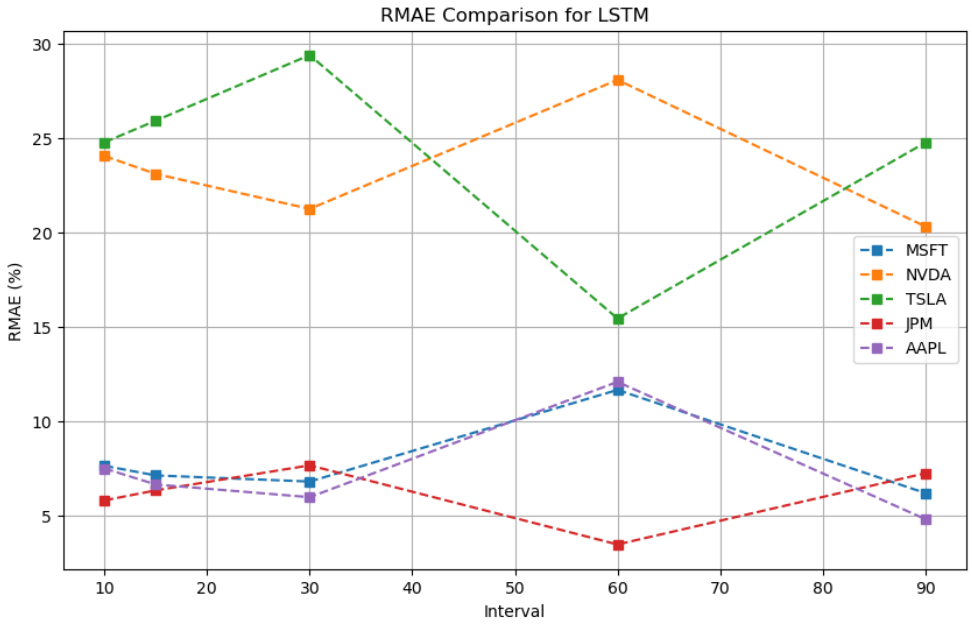

Figure 3: RMAE of the LSTM Model Across Different Time Intervals and Stocks

ticularly at shorter horizons, reflecting the volatility of the stock. NVIDIA Corporation (NVDA) demonstrates relatively stable error rates across all time horizons.

Table 7 and Figure 4 present the RMAE values, enabling a comparative analysis of Time-LLM's performance across different time horizons and stocks. For Apple Inc. (AAPL), the lowest RMAE is observed at 10 days, suggesting stronger short-term accuracy, while errors increase in longer-term forecasts, breaking down again at 90 days. NVIDIA Corporation (NVDA), reaching the highest RMAE among all stocks, and Microsoft Corporation (MSFT) follow a similar pattern, with RMAE increasing from the lowest value at 10 days to its highest value at 90 days. It reflects the growing difficulty of the Time-LLM model in capturing long-term price movements. JPMorgan Chase & Co. (JPM), exhibits the lowest RMAE values among all stocks, particularly at 60 days and 90 days (3.44 and 4.13), underscoring the observation that both deep learning models align well with historical price trends for this stock. In contrast, TSLA's RMAE remains relatively high across all time horizons.

Overall, the results suggest that while medium-term forecasts (30 and 60 days) generally

introduce the highest errors for most stocks, long-term predictions (90 days) yield more stable relative performance. These findings highlight the importance of considering stock-specific characteristics when utilizing the Time-LLM model for financial forecasting.

Ticker	**10 Days**		**15 Days**		**30 Days**		**60 Days**		**90 Days**	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
AAPL	10.97	12.84	12.04	14.26	17.11	19.63	18.63	21.43	16.29	18.92
MSFT	25.79	30.37	27.62	33.30	34.91	41.35	35.32	42.16	37.39	44.44
JPM	6.52	8.20	6.02	7.72	5.35	7.02	4.44	5.47	5.32	6.24
TSLA	21.14	25.12	15.14	19.22	20.48	25.02	18.16	22.72	17.85	22.34
NVDA	5.03	5.84	5.49	6.28	7.02	7.77	8.51	9.31	9.23	9.89

Table 6: Time-LLM Absolute Errors MAE and RMSE

Ticker	**10 Days**	**15 Days**	**30 Days**	**60 Days**	**90 Days**
AAPL	6.83	7.49	10.65	11.60	10.14
MSFT	9.17	9.82	12.41	12.56	13.29
JPM	5.06	4.67	4.15	3.44	4.13
TSLA	11.62	8.32	11.25	9.98	9.81
NVDA	18.65	20.37	26.06	31.59	34.26

Table 7: Time-LLM Relative Errors RMAE as % of Stock's Average Price

5.4 ARIMA Performance Evaluation

This section provides an assessment of the ARIMA model's[4] predictive performance as a benchmark for comparison with deep learning models – LSTM and Time-LLM. The ARIMA model serves as a baseline to evaluate improvements achieved by more advanced predictive algorithms.

Table 8 presents the MAE and RMSE values across different prediction horizons (10, 15, 30, 60, and 90 days) for various stock tickers. For Apple Inc. (AAPL), the model shows decreasing MAE and RMSE values as the prediction horizon increases, indicating relatively stable long-term performance. A similar pattern is observed for Microsoft Corporation (MSFT), where short-term predictions exhibit high error rates (with extremely high rate at

[4]Mathematical formulation of ARIMA model is presented in Appendix A.

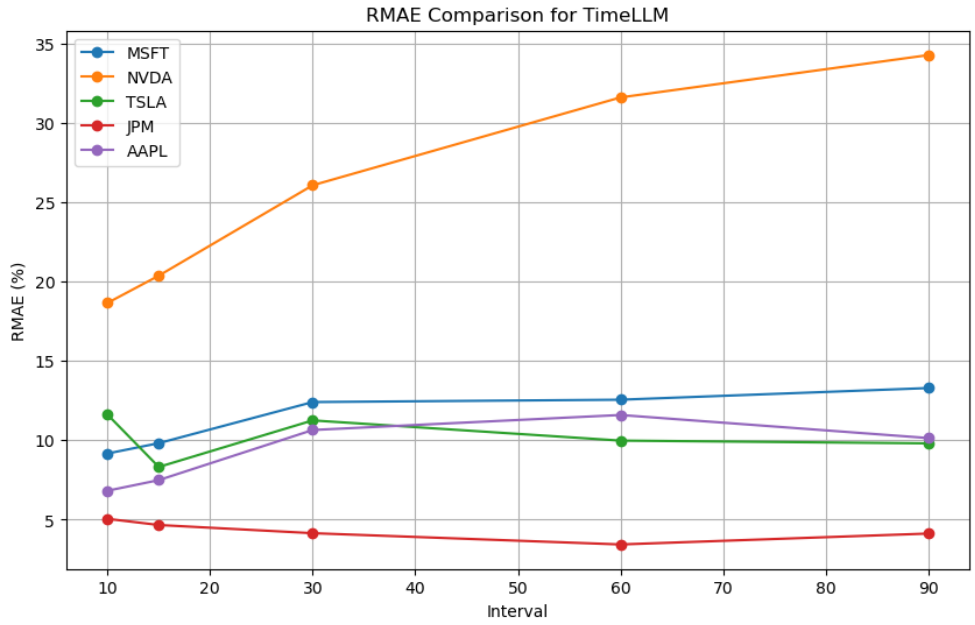

Figure 4: RMAE of the Time-LLM Model Across Different Time Intervals and Stocks

10 days), while long-term predictions demonstrate gradual improvement. JPMorgan Chase & Co. (JPM) exhibits relatively lower errors across all time horizons, signifying stable price movements. However, Tesla Inc. (TSLA) remains highly volatile, with extremely high MAE and RMSE values in the short term. NVIDIA Corporation (NVDA) shows decreasing values with growing trading windows, not aligning with the patterns of both deep learning models, having relative stable metrics values for this stock across different time horizons.

Table 9 and Figure 5 provide the RMAE values, enabling a comparative evaluation of ARIMA's effectiveness as a benchmark model. The results highlight the model's limitations when compared to more sophisticated forecasting techniques, especially at the short-time horizons (10 and 15 days). For Apple Inc. (AAPL), the lowest RMAE value is observed at 90 days, indicating improved relative accuracy in long-term forecasts, though shorter horizons still exhibit significant errors. Microsoft Corporation (MSFT) follows a different trend, with RMAE peaking at 10 days (205.83) and gradually decreasing to 23.23 at 90 days, suggesting that ARIMA struggles with short-term volatility but aligns better over extended periods. JPMorgan Chase & Co. (JPM) exhibits the lowest RMAE values among all stocks,

particularly at 90 days (3.15), reinforcing the observation that all the models effectively capture stable stock trends. In contrast, TSLA's RMAE values remain high across all time horizons, with extreme values at 10 days (206.32) and still elevated at 30 days (42.87), highlighting the challenge of forecasting highly volatile stocks for both statistical and deep learning models. NVIDIA Corporation (NVDA) presents mixed results, with significantly high RMAE at the short term (126.62 at 10 days) and extremely low at 90 days (3.20), comparing to other stocks and time intervals.

Overall, while ARIMA struggles with short-term fluctuations, it does not perform significantly worse than deep learning methods for longer-term trends.

Ticker	**10 Days**		**15 Days**		**30 Days**		**60 Days**		**90 Days**	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
AAPL	59.87	67.65	53.38	60.29	28.96	33.07	13.26	15.18	11.24	12.89
MSFT	774.80	852.51	95.62	104.56	51.22	57.29	132.09	146.34	87.45	97.67
JPM	85.82	114.90	33.57	42.34	18.01	25.27	13.16	18.29	5.63	7.47
TSLA	492.72	733.37	372.04	547.79	102.38	132.64	16.39	19.49	19.87	23.09
NVDA	100.99	125.70	41.59	50.41	19.78	25.74	19.58	25.43	2.55	3.51

Table 8: ARIMA Absolute Errors MAE and RMSE

Ticker	**10 Days**	**15 Days**	**30 Days**	**60 Days**	**90 Days**
AAPL	30.70	27.37	14.85	6.80	5.76
MSFT	205.83	25.40	13.61	35.09	23.23
JPM	47.96	18.76	10.06	7.35	3.15
TSLA	206.32	155.78	42.87	6.86	8.32
NVDA	126.62	52.14	24.80	24.55	3.20

Table 9: ARIMA Relative Errors RMAE as % of Stock's Average Price

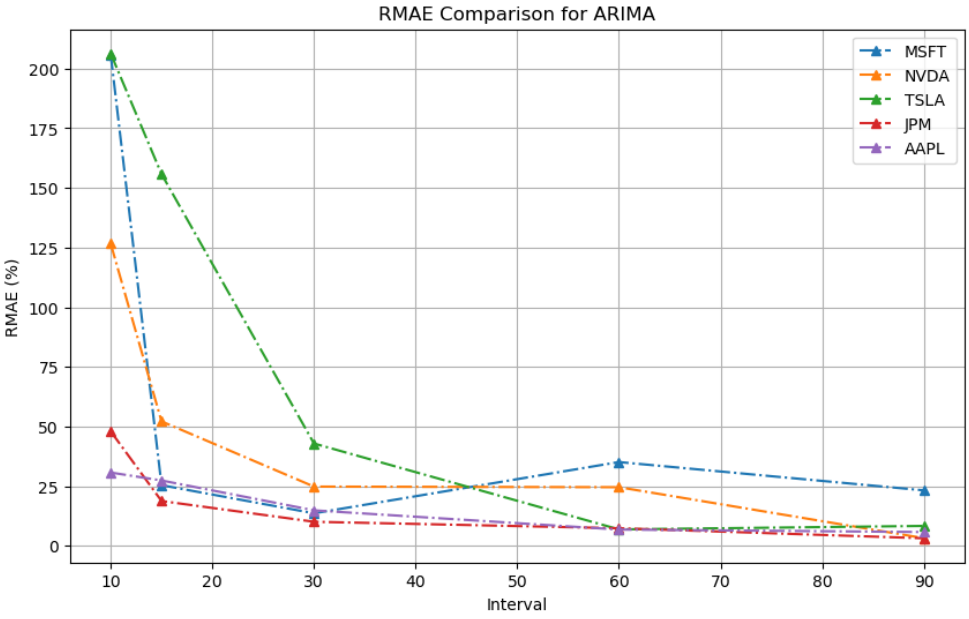

Figure 5: RMAE of the ARIMA Model Across Different Time Intervals and Stocks

6 Conclusion

This study evaluated the predictive performance of LSTM, Time-LLM, and ARIMA models for stock price forecasting across different time horizons. The analysis covered short-term (10–15 days), mid-term (30 days), and long-term (60–90 days) predictions. While LSTM and Time-LLM generally outperformed the benchmark ARIMA model, their effectiveness was notably higher for stocks with stable price trends, whereas highly volatile stocks like Tesla Inc. (TSLA) and NVIDIA Corporation (NVDA) posed greater challenges for accurate forecasting.

The results indicate that medium-term forecasts (30 days) generally yield lower errors, while longer-term predictions (60–90 days) introduce increased uncertainty. The findings suggest that if the training data comes from a relatively stable period and the test data includes more volatile market conditions, the model may perform worse over longer prediction windows because it has not learned to handle such volatility well. ARIMA model struggles significantly with short-term forecasts (10–15 days) but performs reliably for long-term trends.

JPMorgan Chase & Co. (JPM), consistently exhibited the lowest errors across deep learning models, indicating that its price movements are more predictable. In contrast, Tesla Inc. (TSLA) presented the highest error values, highlighting the challenge of forecasting volatile stocks. NVIDIA Corporation (NVDA) demonstrated stable high error rates across different terms.

Overall, deep learning models outperform ARIMA in handling stock market fluctuations, but exhibit limitations when forecasting highly volatile stocks. The Time-LLM model generally proves to be a strong and compatible approach for leveraging large language models in time-series forecasting. Incorporating additional features such as other technical indicators, or combining deep learning approaches, including news sentiment analysis, may enhance overall predictive accuracy.

References

Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S. S., Arango, S. P., Kapoor, S., Zschiegner, J., Maddix, D. C., Wang, H., Mahoney, M. W., Torkkola, K., Wilson, A. G., Bohlke-Schneider, M., and Wang, Y. (2024). Chronos: Learning the language of time series.

Appel, G. (2005). *Technical Analysis: Power Tools for Active Investors*. Financial Times Prentice Hall.

Arras, L., Arjona-Medina, J., Widrich, M., Montavon, G., Gillhofer, M., Müller, K.-R., Hochreiter, S., and Samek, W. (2021). Explaining and interpreting LSTMs. *Fraunhofer Heinrich Hertz Institute, Johannes Kepler University Linz, Technische Universität Berlin, Korea University, Max Planck Institute for Informatics*.

Basak, S., Kar, S., Saha, S., Khaidem, L., and Dey, S. R. (2019). Predicting the direction of stock market prices using tree-based classifiers. *The North American Journal of Economics and Finance*, 47:552–567.

Box, G. E. P. and Jenkins, G. M. (1970). *Time Series Analysis: Forecasting and Control*. Holden-Day.

Chang, C., Wang, W.-Y., Peng, W.-C., and Chen, T.-F. (2024). LLM4TS: Aligning pre-trained llms as data-efficient time-series forecasters.

Fischer, T. and Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669.

Franke, J., Härdle, W. K., and Hafner, C. M. (2019). *Statistics of Financial Markets: An Introduction*. Springer, third edition.

Granville, J. E. (1963). *Granville's New Key to Stock Market Profits*. Prentice Hall, Englewood Cliffs, NJ.

Gruver, N., Finzi, M., Qiu, S., and Wilson, A. G. (2024). Large language models are zero-shot time series forecasters.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9:1735–1780.

Jabed, M. I. K. (2024). Stock market price prediction using machine learning techniques. *American International Journal of Sciences and Engineering Research*, 7:1–6.

Jiang, Y., Pan, Z., Zhang, X., Garg, S., Schneider, A., Nevmyvaka, Y., and Song, D. (2024). Empowering time series analysis with large language models: A survey.

Jin, M., Wang, S., Ma, L., Chu, Z., Zhang, J. Y., Shi, X., Chen, P.-Y., Liang, Y., Li, Y.-F., Pan, S., and Wen, Q. (2024). Time-LLM: Time series forecasting by reprogramming large language models.

Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., and Choo, J. (2022). Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*.

Ko, C.-R. (2021). LSTM-based sentiment analysis for stock price forecast. *PeerJ Computer Science*, 7:e408.

Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., and Gao, J. (2024). Large language models: A survey. *arXiv preprint*, arXiv:2402.06196.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*.

Onozo, L., Arthur, F., and Gyires-Tóth, B. (2024). Leveraging LLMs for financial news analysis and macroeconomic indicator nowcasting. *IEEE Access*, PP:1–1.

Pang, T. (2024). Appl stock price prediction based on LSTM and GRU. *Applied and Computational Engineering*, 47:200–206.

Shen, Y. and Zhang, P. K. (2024). Financial sentiment analysis on news and reports using large language models and finbert.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Waqas, M. and Humphries, U. (2024). A critical review of RNN and LSTM variants in hydrological time series predictions. *MethodsX*, 13.

Wilder, J. W. (1978). *New Concepts in Technical Trading Systems*. Trend Research.

A ARIMA

Autoregressive Integrated Moving Average (ARIMA) is a statistical model used for time series forecasting. It combines three components: autoregression (AR), differencing (integrated process) (I), and a moving average (MA). The linear *autoregressive* process of order p, AR(p), is defined as

$$X_t = \nu + \alpha_1 X_{t-1} + \cdots + \alpha_p X_{t-p} + \varepsilon_t, \tag{27}$$

which can also be written as

$$\alpha(L) X_t = \nu + \varepsilon_t, \tag{28}$$

where $\alpha(L)$ represents the characteristic polynomial in terms of the lag operator L defined as $L^k X_t = X_{t-k}$; X_t is the time series at time t; and ϵ_t is a white noise error term. The process X_t is *integrated* of order d, denoted as $I(d)$, if the transformed series $(1-L)^{d-1} X_t$ is non-stationary, while applying an additional difference results in a stationary process $(1-L)^d X_t$. The *moving average* process of order q, MA(q), is defined as

$$X_t = \beta_1 \varepsilon_{t-1} + \cdots + \beta_q \varepsilon_{t-q} + \varepsilon_t, \tag{29}$$

which can also be written as

$$X_t = \beta(L) \varepsilon_t, \tag{30}$$

where the polynomial representation is given by

$$\beta(L) = 1 + \beta_1 L + \cdots + \beta_q L^q. \tag{31}$$

Parameter selection is used to determine the optimal values of p, d, q, utilizing criteria such as the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC). Once the parameters are selected, Maximum Likelihood Estimation is employed to estimate the coefficients, ensuring the model best fits the given time series data. Finally, the trained model is used for forecasting (Franke et al., 2019).