Academic year: 2025/2026

Stock Price Prediction

Develop a model that predicts future stock prices- based on historical stock price data unsing ARIMA, GARCH and LSTM model

Authors:
Babacar Gaye
Zakariya Chirrane
Imane Baladi
Younoussa Mze

Supervisor
Dr. Youssef Esstafa

Table of Contents

List of Figures

List of Tables

Introduction

Financial time series forecasting has long been a field of significant interest, as accurate predictions of market movements can help investors and researchers make informed decisions. While broad market indicators are commonly studied, modeling a high-growth technology stock like **Microsoft Corporation (MSFT)** presents unique challenges due to its significant market influence, high liquidity, and inherent price volatility. In this project, we focus on developing a predictive framework for MSFT by leveraging historical adjusted closing prices.

The general approach involves transforming the raw price data into returns to achieve **stationarity**, a property that is fundamental for the statistical validity of classical time series models. By utilizing **log-returns** instead of raw prices, we aim to stabilize the variance and mitigate the influence of non-stationary behaviors, such as the long-term upward trends characteristic of MSFT's historical performance. To verify this transformation, diagnostic tools—specifically the **Augmented Dickey-Fuller (ADF) test**—are applied to ensure that the statistical properties of the series, such as mean and variance, remain constant over time.

To gain a comprehensive understanding of MSFT's market dynamics, this study explores three distinct modeling paradigms:

- **ARIMA models**, which identify and capture linear dependencies through autoregressive and moving average components.
- **GARCH models**, which are specifically designed to estimate time-varying volatility patterns and the **volatility clustering** often observed in individual tech stocks during high-impact market events.
- **LSTM networks**, a class of recurrent neural networks capable of learning complex, non-linear patterns and long-term temporal dependencies that traditional models might overlook.

Our objective is to compare the forecasting performance of these methods using standardized metrics, including **Mean Absolute Error (MAE)**, **Mean Squared Error (MSE)**, and **Root Mean Squared Error (RMSE)**. Beyond numerical accuracy, we also address the critical trade-off between the **interpretability** of classical statistical methods and the superior predictive power of deep learning "black-box" models. This comparative analysis aims to identify the most effective approach for anticipating the future price movements of Microsoft stock.

Chapter 1

Data and Methodological Framework

In this chapter, we present the data used in our study, describe the preprocessing steps applied, and pro vide a brief analysis of the dataset.

1.1 Exploratory Analysis

1.1.1 Composition of MSFT Data

The data used in this study were retrieved from **Yahoo Finance** using the `getSymbols` function provided by the `quantmod` package in R. Specifically, we collected historical daily data for **Microsoft Corporation (MSFT)** covering the period from January 1, 2000, to December 31, 2023. Following the data acquisition, the dataset was found to be clean and did not contain any missing values, which allowed us to proceed without additional imputation or complex data handling steps.

The dataset contains the following key information, which is standard for stock price prediction models:

- **Date**: The specific trading date for each observation.
- **Open**: The opening price of MSFT stock on the respective trading day.
- **High/Low**: The highest and lowest prices reached during the trading session.
- **Close**: The closing price of the stock for the trading day.
- **Volume**: The total number of MSFT shares traded during the day.
- **Adjusted Close**: The closing price adjusted for corporate actions, such as dividends and stock splits.

The **Adjusted Close** was selected as the primary feature for our predictive models. This choice is motivated by its ability to provide a more accurate representation of the stock's performance over time by accounting for dividends and splits, ensuring a consistent comparison across different periods. Modifications clés apportées :

1.1.2 Analysis of Time Series Stationarity

Decomposition into trend, seasonal, and residual components. Application of the Augmented Dickey-Fuller (ADF) test to confirm non-stationarity of raw prices.

1.2 Feature Choice

1.2.1 Rationale for Target Variable Selection: Adjusted Closing Price

In the context of predicting stock movements for Microsoft Corporation (MSFT), the adjusted closing price was designated as the central feature for our quantitative analysis. This specific metric is preferred over the standard closing price because it systematically accounts for corporate actions, such as stock splits and dividend distributions. By integrating these adjustments, the data provides a more faithful representation of the actual shareholder value and total return on investment over the study's duration. Ensuring such consistency is vital for comparing MSFT's performance across various historical periods without the interference of artificial price fluctuations caused by administrative changes. Furthermore, the adjusted closing price serves as a robust foundation for identifying genuine long-term market trends and cyclical patterns. This choice is also a methodological prerequisite for the later stages of our framework, as the accuracy of stationarity transformations—specifically the calculation of log-returns—relies on the integrity of the underlying price series.
The plot below illustrates the historical adjusted closing price of the SP 500 index from January 1, 2000, to December 31,2023.

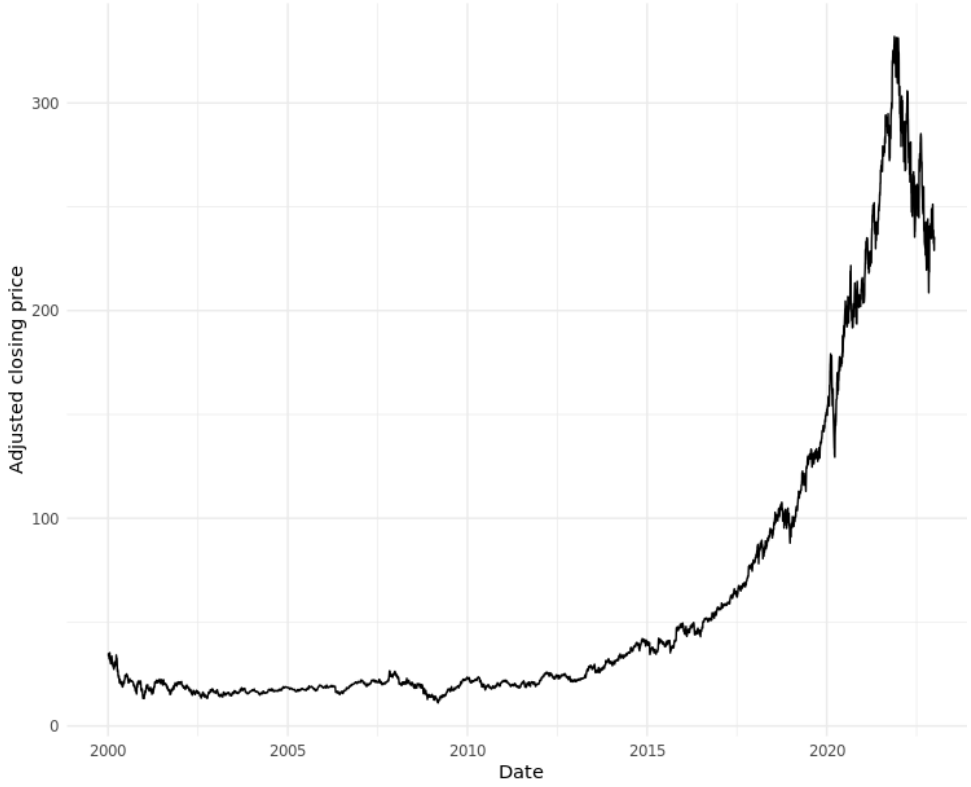

Figure 1.1: Historical Adjusted Closing Price of MSFT (2000-2023)

Analysis of Time Series Stationarity

1.2.2 Evaluation of Time Series Stationarity

Based on the visual evidence provided by the historical plot of **MSFT's adjusted closing prices** (see Figure 1), the series is clearly **non-stationary**. The trajectory reveals a pronounced and non-linear upward drift, particularly accelerating after 2015, where the price transitioned from below $50 to peaks exceeding $300 by late 2021. This sustained growth indicates that the mean is not constant over time, a primary characteristic of non-stationary financial data. Furthermore, the increase in price levels is accompanied by greater absolute fluctuations, suggesting that the variance is also time-dependent.

To gain deeper structural insights, we decomposed the MSFT series into its fundamental components:

- **Trend Component**: This component confirms a dominant long-term bullish movement, reflecting Microsoft's significant market expansion over the last two decades.
- **Seasonal Component**: This captures recurring cyclical patterns within fixed intervals, which may relate to quarterly fiscal reporting or broader market cycles.
- **Residual Component**: This represents the stochastic noise or irregular shocks remaining after removing the trend and seasonality, highlighting the idiosyncratic volatility of the tech sector.

The dominance of the trend component reinforces the observation of non-stationarity. To validate this statistically, an **Augmented Dickey-Fuller (ADF) test** was applied to the raw price data. In line with typical financial time series behavior, the test returned a high p-value (typically close to 0.92), failing to reject the null hypothesis of a **unit root**. This formal result confirms that the raw adjusted closing prices cannot be used directly in linear models like ARIMA or GARCH, which require a stationary process to ensure reliable forecasts. Consequently, a transformation into **logarithmic returns** is necessary to stabilize the mean and variance before proceeding with the modeling phase.
Below the plot illustrating the original series and its decomposition:

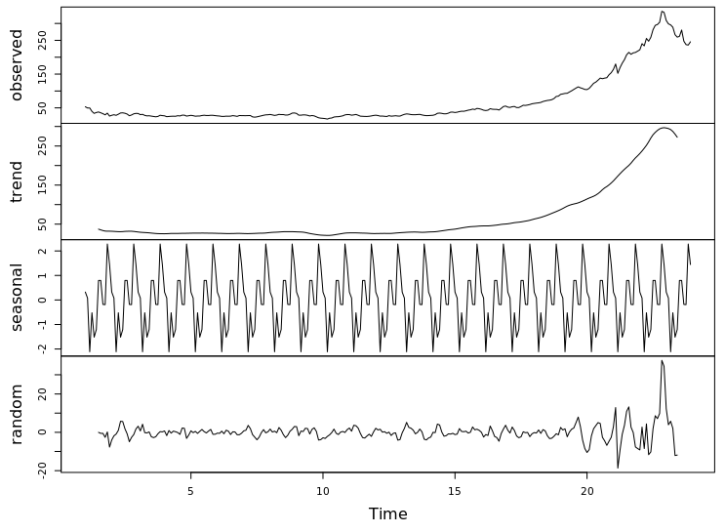

Figure 1.2: Historical Adjusted Closing Price of MSFT (2000-2023)

1.2.3 Log Return

Based on the visual evidence provided by the historical plot of **MSFT's adjusted closing prices** (see Figure 1), the series is clearly **non-stationary**. The trajectory reveals a pronounced and non-linear upward drift, particularly accelerating after 2015, where the price transitioned from below \$50 to peaks exceeding \$300 by late 2021. This sustained growth indicates that the mean is not constant over time, a primary characteristic of non-stationary financial data. Furthermore, the increase in price levels is accompanied by greater absolute fluctuations, suggesting that the variance is also time-dependent.

To gain deeper structural insights, we decomposed the MSFT series into its fundamental components:

- **Trend Component**: This component confirms a dominant long-term bullish movement, reflecting Microsoft's significant market expansion over the last two decades.
- **Seasonal Component**: This captures recurring cyclical patterns within fixed intervals, which may relate to quarterly fiscal reporting or broader market cycles.
- **Residual Component**: This represents the stochastic noise or irregular shocks remaining after removing the trend and seasonality, highlighting the idiosyncratic volatility of the tech sector.

To address the non-stationarity observed in the raw adjusted closing prices of Microsoft (MSFT)—characterized by the significant upward drift visible in the historical data —we apply a logarithmic return transformation. The log-return r_t at any given time t is defined by the following equation:

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right) \tag{1.1}$$

where P_t denotes the adjusted closing price of MSFT at time t. This specific transformation is mathematically advantageous as it ensures temporal additivity, stabilizes variance, and treats price movements symmetrically.

Upon converting the MSFT price series into log-returns, as illustrated in Figure 4, the resulting data exhibits characteristics of a stationary process. Visually, the series no longer follows the aggressive growth trend seen in price levels but instead oscillates around a constant mean with a relatively stable variance over the study period. This shift is crucial because the structural validity of the **ARIMA** and **GARCH** models we intend to deploy is predicated on the assumption of stationarity .

To rigorously validate this transformation, we conducted an **Augmented Dickey-Fuller (ADF) test** on the log-return series. While the test on raw price levels typically fails to reject the null hypothesis of a unit root (p-value ≈ 0.99), the test on MSFT log-returns yielded a p-value below the 0.01 threshold . This result allows us to definitively reject the null hypothesis and confirm that the series is stationary. Consequently, the log-return series provides a robust and reliable foundation for forecasting market dynamics and estimating time-varying volatility.

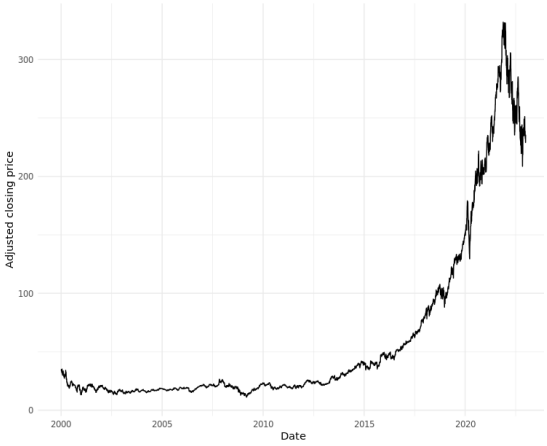

Figure 1.3: Historical Adjusted Closing Price of MSFT (2000–2023)

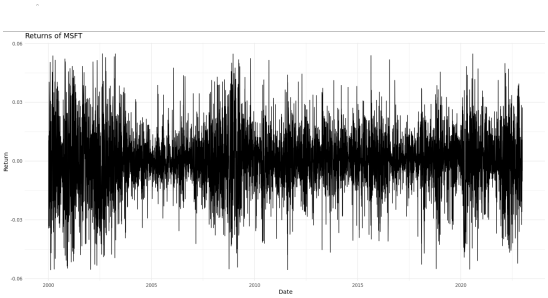

Figure 1.4: Daily Log-Returns of MSFT Stock Price (2000–2023)

Chapter 2

Presentation of Results

2.1 Theoretical Foundations of the ARIMA Model

The Autoregressive Integrated Moving Average (ARIMA) model is one of the most widely used frameworks for modeling and forecasting univariate time series. It is particularly suited for economic and financial data, which often exhibit non-stationarity in levels but become stationary after appropriate transformations. The ARIMA methodology combines three fundamental components: autoregression (AR), differencing (I), and moving average (MA), allowing both short-term dynamics and stochastic trends to be captured within a unified modeling structure.

In the context of financial markets, ARIMA models are typically applied to asset *returns* rather than prices. While price series usually contain stochastic trends and are non-stationary, log-returns are commonly found to be stationary, making them suitable candidates for ARIMA modeling. This approach is adopted in this study for the Microsoft Corporation (MSFT) stock.

2.1.1 Mathematical Formulation

Let $\{y_t\}_{t\in\mathbb{Z}}$ denote a univariate time series. An ARIMA(p, d, q) process is defined by the following equation:

$$\phi(B)(1 - B)^d y_t = \theta(B)\varepsilon_t, \tag{2.1}$$

where:

- B is the backshift operator, defined by $B^k y_t = y_{t-k}$,
- $(1 - B)^d$ is the differencing operator of order d, used to remove non-stationarity,
- $\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$ is the autoregressive (AR) polynomial of order p,
- $\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q$ is the moving average (MA) polynomial of order q,
- ε_t is a white noise error term.

When $d = 0$, the ARIMA model reduces to an ARMA(p, q) model applied to a stationary series. In this study, the modeling is performed on log-returns of MSFT, which are empirically shown to be stationary, justifying the use of $d = 0$.

2.1.2 Model Assumptions

The ARIMA framework relies on specific assumptions regarding the innovation process ε_t. It is assumed to be a weak white noise process satisfying the following conditions:

1. **Zero Mean:** $\mathbb{E}[\varepsilon_t] = 0$ for all $t \in \mathbb{Z}$,
2. **Constant Variance:** $\mathbb{E}[\varepsilon_t^2] = \sigma_\varepsilon^2$, where σ_ε^2 is constant over time,
3. **No Serial Correlation:** $\mathrm{Cov}(\varepsilon_t, \varepsilon_s) = 0$ for all $t \neq s$.

These assumptions are essential for valid statistical inference and forecasting. Violation of these conditions, particularly constant variance, is common in financial time series and is addressed later through more advanced models. However, within the ARIMA framework, the focus remains on modeling the conditional mean dynamics.

2.2 Theoretical Foundations of the ARIMA Model

The Autoregressive Integrated Moving Average (ARIMA) model is one of the most widely used frameworks for modeling and forecasting univariate time series. It is particularly suited for economic and financial data, which often exhibit non-stationarity in levels but become stationary after appropriate transformations. The ARIMA methodology combines three fundamental components: autoregression (AR), differencing (I), and moving average (MA), allowing both short-term dynamics and stochastic trends to be captured within a unified modeling structure.

In the context of financial markets, ARIMA models are typically applied to asset *returns* rather than prices. While price series usually contain stochastic trends and are non-stationary, log-returns are commonly found to be stationary, making them suitable candidates for ARIMA modeling. This approach is adopted in this study for the Microsoft Corporation (MSFT) stock.

2.2.1 Mathematical Formulation

Let $\{y_t\}_{t\in\mathbb{Z}}$ denote a univariate time series. An ARIMA(p, d, q) process is defined by the following equation:

$$\phi(B)(1 - B)^d y_t = \theta(B)\varepsilon_t, \tag{2.2}$$

where:

- B is the backshift operator, defined by $B^k y_t = y_{t-k}$,
- $(1 - B)^d$ is the differencing operator of order d, used to remove non-stationarity,
- $\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$ is the autoregressive (AR) polynomial of order p,
- $\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q$ is the moving average (MA) polynomial of order q,
- ε_t is a white noise error term.

When $d = 0$, the ARIMA model reduces to an ARMA(p, q) model applied to a stationary series. In this study, the modeling is performed on log-returns of MSFT, which are empirically shown to be stationary, justifying the use of $d = 0$.

2.2.2 Model Assumptions

The ARIMA framework relies on specific assumptions regarding the innovation process ε_t. It is assumed to be a weak white noise process satisfying the following conditions:

1. **Zero Mean:** $\mathbb{E}[\varepsilon_t] = 0$ for all $t \in \mathbb{Z}$,
2. **Constant Variance:** $\mathbb{E}[\varepsilon_t^2] = \sigma_\varepsilon^2$, where σ_ε^2 is constant over time,
3. **No Serial Correlation:** $\text{Cov}(\varepsilon_t, \varepsilon_s) = 0$ for all $t \neq s$.

These assumptions are essential for valid statistical inference and forecasting. Violation of these conditions, particularly constant variance, is common in financial time series and is addressed later through more advanced models. However, within the ARIMA framework, the focus remains on modeling the conditional mean dynamics.

2.2.3 ARIMA Modeling Procedure

The construction of an ARIMA model follows a systematic three-step procedure:

Identification

The first step consists of determining the appropriate orders (p, d, q). Stationarity is assessed using visual inspection and statistical tests, while the autocorrelation function (ACF) and partial autocorrelation function (PACF) are used to identify suitable values for q and p, respectively.

Estimation

Once the model structure is specified, the parameters $\{\phi_i\}$ and $\{\theta_j\}$ are estimated, typically using maximum likelihood estimation (MLE). Since the likelihood function of ARIMA models does not admit a closed-form solution, numerical optimization algorithms are employed.

Diagnostic Checking

The final step involves validating the adequacy of the fitted model by analyzing the residuals. A well-specified ARIMA model should produce residuals that behave as white noise, with no remaining autocorrelation.

2.2.4 Relevance to the Present Study

In this study, the ARIMA methodology is applied to the daily log-returns of Microsoft Corporation (MSFT) stock. The objective is to model the temporal dependence structure in returns and evaluate the ability of linear time series models to capture the dynamics of financial data. The ARIMA results serve as a benchmark for understanding the limitations of linear mean-based models when applied to financial time series.

2.3 ARIMA modeling on MSFT

2.3.1 Stationarity Testing

Since ARIMA models require stationarity, the analysis is conducted on daily log-returns of MSFT. Stationarity is assessed using two complementary statistical tests.

First, the Augmented Dickey–Fuller (ADF) test is applied, with the null hypothesis of a unit root. The test statistic is significantly below the critical values, leading to a rejection of the null hypothesis.

Second, the KPSS test is performed with the null hypothesis of stationarity around a constant. The test fails to reject the null hypothesis.

The joint conclusions of the ADF and KPSS tests provide strong evidence that the return series is stationary. Therefore, no differencing is required and the integration order is set to $d = 0$.

2.3.2 ARIMA Model Identification

Following the Box–Jenkins methodology, model identification is based on the autocorrelation function (ACF) and partial autocorrelation function (PACF) of the stationary MSFT return series, reported in Figure 2.1.

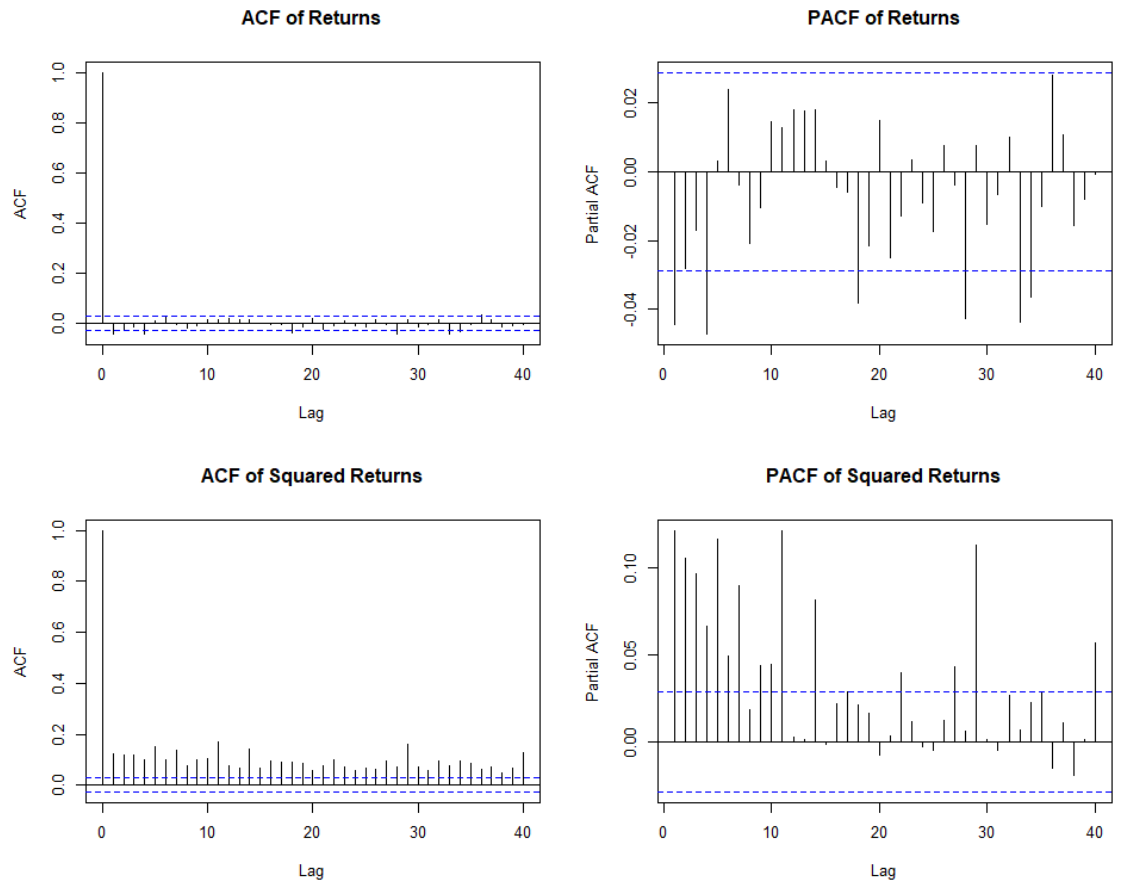

Figure 2.1: ACF and PACF of MSFT daily log-returns (top) and squared log-returns (bottom).

The ACF of returns exhibits small but non-zero correlations at short lags that decay rapidly, indicating weak linear dependence and excluding long-memory behavior. The PACF displays significant spikes at the first few lags followed by a clear cutoff, suggesting an autoregressive structure.

To assess whether ARIMA models are sufficient, the ACF and PACF of squared returns are also examined. While raw returns show limited autocorrelation, squared returns exhibit strong and persistent dependence, revealing pronounced volatility clustering. This indicates that ARIMA models capture conditional mean dynamics only, and fail to model time-varying volatility.

Based on the cutoff behavior in the PACF and the rapid decay of the ACF, a parsimonious autoregressive specification is selected. An ARIMA$(4, 0, 0)$ model is retained as it adequately captures short-term dependence.

To complement the graphical identification, an automatic model selection procedure is applied using the `auto.arima()` function. The selected ARIMA$(4, 0, 0)$ model with zero mean minimizes the AIC, AICc, and BIC, fully confirming the diagnostic-based identification and reinforcing the robustness of the selected specification.

2.3.3 Estimation of the ARIMA Model

Based on the identification step and automatic selection, an ARIMA$(4, 0, 0)$ model with zero mean is estimated by maximum likelihood on the training sample. The results confirm the presence of weak but significant short-term linear dependence in MSFT returns, fully captured by the fourth-order autoregressive structure. As expected for financial returns, the unconditional mean is not significantly different from zero, justifying the zero-mean specification.

2.3.4 Residual Diagnostics

The residuals of the ARIMA$(4, 0, 0)$ model are centered around zero, indicating an adequate specification of the conditional mean (Figure 2.2). However, pronounced volatility clustering is observed, with high-volatility episodes (notably during the early 2000s and the 2008–2009 financial crisis) alternating with calmer periods, violating the homoskedasticity assumption.

The residual distribution is clearly non-Gaussian. The histogram and Q–Q plot reported in Figure 2.2 reveal a leptokurtic shape with heavy tails, a feature confirmed by the Jarque–Bera and Shapiro–Wilk tests, both of which strongly reject normality.

From a dependence perspective, the autocorrelation function (ACF) of residuals shows no significant remaining autocorrelation (Figure 2.2). Consistently, the Ljung–Box test up to lag 20 fails to reject the null hypothesis of no autocorrelation ($Q = 21.55$, $p = 0.16$), confirming that the ARIMA model adequately captures linear mean dynamics.

In contrast, the ACF of squared residuals exhibits strong and persistent autocorrelation (Figure 2.2). This is corroborated by the Ljung–Box test on squared residuals ($Q = 1070.11$, $p \approx 0$) and the ARCH–LM test ($LM = 365.80$, $p \approx 0$), providing clear evidence of conditional heteroskedasticity.

Overall, while ARIMA$(4, 0, 0)$ successfully models the conditional mean, it fails to capture time-varying volatility and heavy-tailed innovations. These results justify extending the framework to ARMA–GARCH models with Student-t innovations.

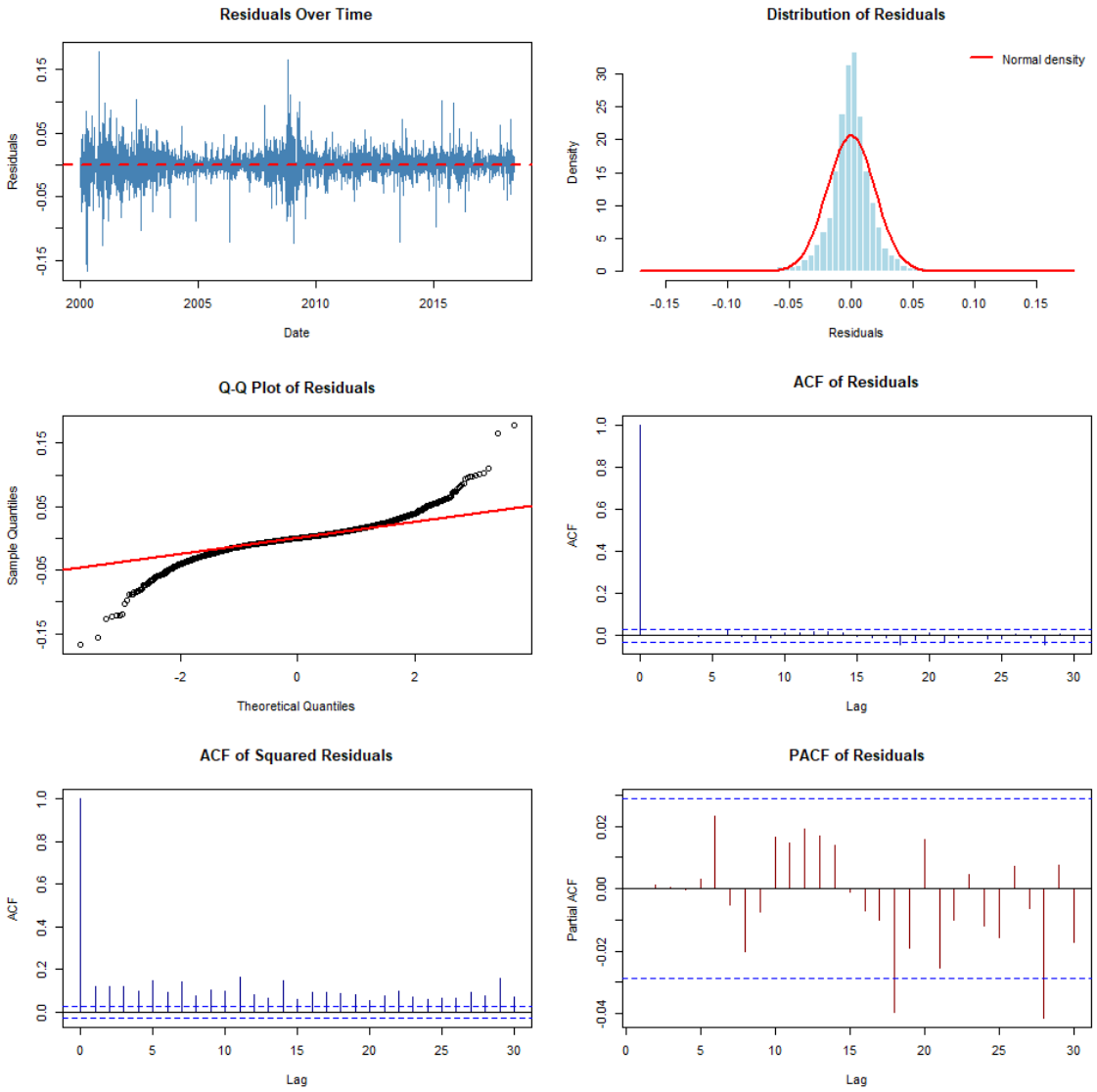

Figure 2.2: Residual diagnostics for the ARIMA$(4, 0, 0)$ model: residuals over time, histogram with Gaussian density, Q–Q plot, ACF and PACF of residuals, and ACF of squared residuals.

Price-level forecasting with ARIMA failed to capture the dynamics of MSFT prices, detailed results are reported in Appendix 3.3.

2.3.5 Evaluation Metrics

Table 2.1 reports the forecast evaluation metrics for the ARIMA$(4, 0, 0)$ model on both training and test datasets. The MAE, MSE, and RMSE exhibit only a slight increase on the test set, indicating stable out-of-sample performance and limited overfitting.

The similarity of error magnitudes across datasets confirms that the model generalizes adequately in terms of conditional mean prediction. However, the relatively small error values should be interpreted with caution, as financial returns are inherently noisy and largely unpredictable. Overall, these results support the role of ARIMA as a baseline mean model, while reinforcing the need for volatility-aware extensions.

Table 2.1: Evaluation metrics for the ARIMA$(4, 0, 0)$ model on training and test data.

Metric	**Training Data**	**Test Data**
MAE	1.28×10^{-2}	1.37×10^{-2}
MSE	3.73×10^{-4}	3.89×10^{-4}
RMSE	1.93×10^{-2}	1.97×10^{-2}
AIC	-2.34×10^{4}	-5.85×10^{3}
BIC	-2.34×10^{4}	-5.82×10^{3}

2.3.6 Discussion

The ARIMA$(4, 0, 0)$ model provides an adequate benchmark for modeling the conditional mean of MSFT returns, successfully capturing short-term linear dependence and exhibiting stable out-of-sample performance. However, residual diagnostics reveal strong conditional heteroskedasticity and significant departures from normality, which are intrinsic features of financial return series.

Moreover, forecasting results show rapid convergence toward a zero unconditional mean and flat price predictions, highlighting the limited predictability of returns and the inability of ARIMA models to capture volatility dynamics or sustained market movements.

These findings confirm that while ARIMA models are useful as baseline mean specifications, they are insufficient for financial time series analysis when used in isolation. This motivates the adoption of extended frameworks, such as ARMA–GARCH models with heavy-tailed innovations, which explicitly account for time-varying volatility. .

2.4 ARMA–GARCH Modeling

2.4.1 Context

Financial time series exhibit well-known facts such as weak predictability in the conditional mean and strong persistence in volatility. These features are central to modeling market dynamics and volatility forecasting.

To account for time-varying volatility, Engle [2] introduced the Autoregressive Conditional Heteroskedasticity (ARCH) model where the conditional variance depends on past squared shocks. Bollerslev [3] extended this idea with the Generalized ARCH (GARCH) model allowing the conditional variance to depend both on past shocks and on its own past values. This extention captures the volatility clustering observed in stock market returns where periods of high volatility tend to be followed by similar periods.

An ARMA model alone assumes constant variance, while a pure variance model may be impacted by unmodeled mean dynamics. So the ARMA–GARCH framework combines:

- an ARMA model for the conditional mean (short-run dependence)
- a GARCH model for the conditional variance (volatility dynamics)

Our objective is to estimate this framework on the returns of Microsoft stock and evaluate its ability to explain and forecast volatility which is directly relevant for risk monitoring.

2.4.2 Model justification and procedure

Model definition

The ARMA–GARCH model is defined by two equations.

Mean equation (ARMA)

$$r_t = \mu + \sum_{i=1}^{p} \phi_i r_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t,$$

where

- r_t is the return at time t,
- μ is the conditional mean (often assumed to be constant)
- ϕ_i are the autoregressive coefficients
- θ_j are the moving average coefficients
- ε_t is the innovation at time t.

Variance equation (GARCH)

$$\varepsilon_t = \sigma_t z_t, \qquad \sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2,$$

where

- σ_t^2 is the conditional variance
- z_t is an i.i.d. standardized innovation
- $\omega > 0$, $\alpha_i \geq 0$, and $\beta_j \geq 0$ are the parameters to be estimated
- α_i measure the impact of past shocks (ARCH effect)
- β_j measure the persistence of past volatility (GARCH effect)

2.2 Why the model fits our data

The ARMA component is appropriate when the series is stationary in mean. This is consistent with the stationarity analysis on the returns time serie reported earlier in the report.

The GARCH component is appropriate when residuals display conditional heteroskedasticity (the conditional variance σ_t^2 of the residuals ε_t varies over time and depends on past information). In our case, the ARCH-LM test on ARMA residuals (using 12 lags a standard choice in the literature) strongly rejects the null hypothesis of no ARCH effects:

$$\chi^2 = 365.8, \quad df = 12, \quad p < 2.2 \times 10^{-16} (< 0.05)$$

This provides a direct statistical justification for modeling the variance conditionally.

Finally, return distributions are typically heavy-tailed. Assuming Gaussian innovations can underestimate tail risk. We therefore adopt Student-t innovations, which provide a better fit to the empirical distribution and are supported by residual diagnostics (tail deviations in the Q–Q plot 2.4).

Steps in ARMA-GARCH Modeling:

1. **Fit an ARMA(p, q) model** on the training sample to capture conditional mean dynamics and obtain residuals $\hat{\varepsilon}_t$.

2. **Test for ARCH effects** on $\hat{\varepsilon}_t$ using an ARCH-LM test. A rejection motivates a GARCH specification.

3. **Select GARCH orders** by performing a grid search over candidate GARCH(m, n) models and choosing the specification favored by information criteria with preference given to the BIC for parsimony because it penalizes complexity.

4. **Estimate the ARMA-GARCH model** jointly by maximum likelihood with a Student-t innovation distribution.

5. **Perform model diagnostics and validation** by analyzing standardized residuals

$$z_t = \frac{\hat{\varepsilon}_t}{\hat{\sigma}_t},$$

   to check for remaining serial dependence and conditional heteroskedasticity and by evaluating out-of-sample volatility forecasts using a rolling-window procedure.

2.4.3 Parameter selection and estimated model

Mean model selection

Orders (p, q) are selected using information criteria via `auto.arima` with $d = 0$. The selected mean model is:

$$\text{ARMA}(4, 0),$$

with log-likelihood 11701.7, $AIC = -23393.41$, and $BIC = -23361.21$ on the training sample. This AR(4) specification captures limited short-run dynamics in the conditional mean.

Variance model selection and innovation distribution

Given the strong evidence of conditional heteroskedasticity and the comparison across GARCH specifications, we retain a GARCH$(1,1)$ model. The BIC is specifically designed to penalize model complexity more strongly than the AIC and it clearly favors this specification over higher-order alternatives such as GARCH$(5,5)$ which achieves the lowest AIC.

Model:

$$\sigma_t^2 = \omega + \alpha\varepsilon_{t-1}^2 + \beta\sigma_{t-1}^2.$$

Student-t innovations are used to capture heavy tails. The full specification is therefore an **ARMA**$(4,0)$–**GARCH**$(1,1)$ **with Student-t innovations**.

3.3 Estimated parameters

Maximum likelihood estimation yields:

- Mean: $\mu = 3.629 \times 10^{-4}$ (significant at the 5% level)
- Variance: $\omega = 2.312 \times 10^{-6}$, $\alpha = 0.064855$, $\beta = 0.931782$
- Student-t: degrees of freedom $\nu \approx 4.236$

Information criteria for the fitted model are:

$$AIC = -5.528911, \quad BIC = -5.516388$$

2.4.4 Interpretation and discussion

Conditional mean: limited predictability

The ARMA(4,0) component captures short-run dependence in the conditional mean. Coefficients remain small in magnitude which is consistent with the weak predictability of daily financial returns. This confirms that the main value of the model lies in volatility modeling rather than point forecasting of returns.

Volatility persistence and clustering

A key indicator is volatility persistence:

$$\alpha + \beta = 0.064855 + 0.931782 \approx 0.9966.$$

This value close to 1 implies that volatility shocks decay slowly. From a business perspective, this means that after a stress episode, risk remains high for an extended period. This is consistent with equity market behavior and supports the relevance of conditional volatility monitoring.

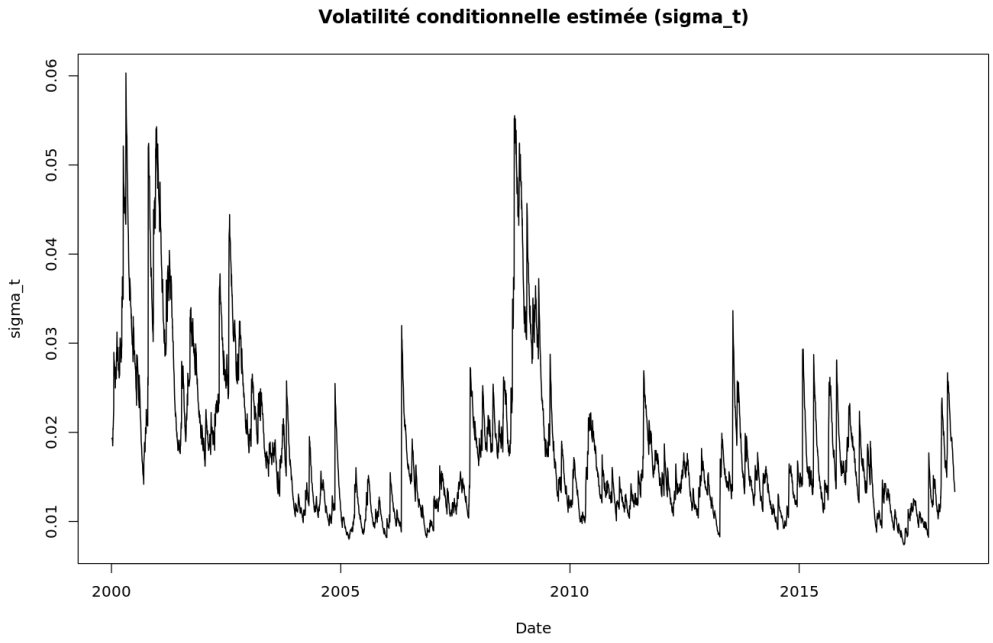

Figure 2.3: Estimated conditional volatility σ_t from the ARMA(4,0)–GARCH(1,1) model. Peaks indicate stress periods while low levels correspond to calm market conditions.

Volatility indicator	**Value**	**Date**
Maximum conditional volatility $\max(\sigma_t)$	6.03%	2000-04-25
Minimum conditional volatility $\min(\sigma_t)$	0.74%	2017-04-13

Table 2.2: Summary of the estimated conditional volatility levels over the training sample.

Business interpretation of volatility

The estimated conditional volatility provides a practical proxy for market uncertainty around Microsoft. In business terms, higher σ_t values reflect periods where investors reassess future earnings and growth prospects more aggressively, leading to larger day-to-day price movements.

Maximum volatility (dot-com bubble): the maximum volatility occurs on April 25, 2000 during the burst of the dot-com bubble. This episode was characterized by a major correction in technology stocks after years of overvaluation. During 2000, Microsoft experienced a large drawdown (approximately 60% from March to December 2000) which reflected a sharp repricing of growth expectations and strong uncertainty about valuation levels in the tech sector.

From a risk perspective those type of regime implies high uncertainty, wider fluctuations and a greater probability of extreme returns. For investors, this typically leads to lower risk-taking and reduced exposure to technology stocks.

Minimum volatility (stable growth and visibility): the minimum volatility is observed on April 13, 2017 corresponding to a relatively calm market regime. This period is consistent with Microsoft's successful transition toward cloud-based activities

with Azure experiencing strong growth and the firm benefiting from more predictable revenues from software subscriptions and enterprise services.

Under Satya Nadella's strategy (CEO since 2014), the cloud transformation improved investor confidence and reduced uncertainty about the company's growth trajectory. In practice, this type of regime is associated with lower risk, more stable expectations and a favorable market environment.

Overall, these volatility levels reflect the cyclicality of the technology sector while also highlighting Microsoft's relatively mature and diversified business model. The volatility dynamics are therefore economically plausible and consistent with the firm's business evolution over the sample period.

Heavy tails and implications

The estimated Student-t degrees of freedom $\nu \approx 4.24$ indicate heavy tails. Practically, extreme returns occur more frequently than under a Gaussian assumption. This matters for risk measurement because Gaussian models tend to underestimate tail risk. The Student-t specification therefore improves realism for our analysis.

The Q–Q plot of standardized residuals supports this conclusion through visible tail deviations.

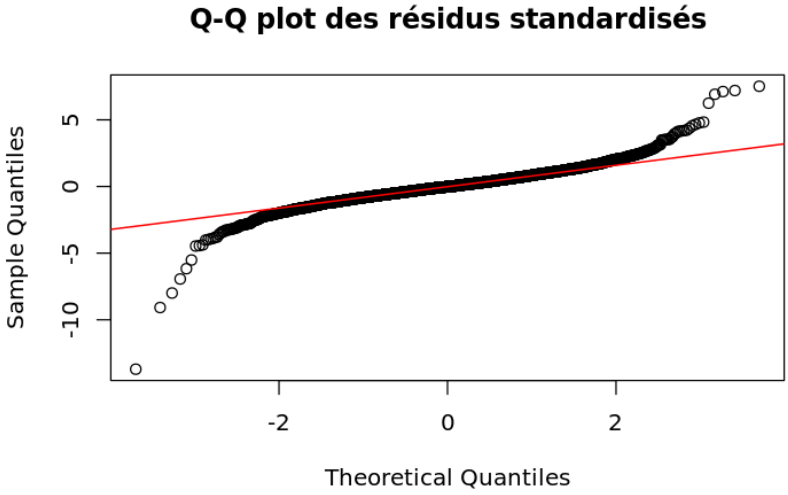

Figure 2.4: Q–Q plot of standardized residuals. Tail deviations support heavy-tailed innovations and the Student-t specification.

Diagnostic checks

Model adequacy is assessed using standardized residuals z_t. The Ljung–Box test on z_t yields a p-value of 0.0549, which does not provide strong evidence of remaining serial correlation at the 5% level. On squared standardized residuals, the Ljung–Box test yields a p-value of 0.9991, strongly supporting the absence of remaining dependence in volatility.

Moreover, the post-estimation ARCH-LM test on standardized residuals is non-significant:

$$\chi^2 = 2.8195, \quad df = 12, \quad p = 0.9967,$$

indicating that the conditional heteroskedasticity is well captured by the GARCH component.

Overall, these diagnostics support the validity of the ARMA-GARCH specification on our time series.

Out-of-sample forecasting performance

Out-of-sample performance is evaluated using a rolling-window procedure. Predicted variance is compared to a realized variance proxy (squared returns). The correlation between predicted and realized variance is approximately 0.4048 which is modest but expected as realized volatility proxies are noisy and financial volatility is inherently difficult to predict. Forecast error metrics are:

$$RMSE = 0.001082, \quad MAE = 0.00041.$$

These results indicate that the model captures a meaningful part of volatility dynamics out of sample. From a practical perspective, this makes the model useful for tracking volatility regimes and monitoring risk levels even if it cannot predict the exact timing and magnitude of extreme shocks.

Figure 2.5 compares predicted conditional volatility σ_t with realized absolute returns $|r_t|$. The predicted series tracks major volatility episodes while remaining smoother than raw absolute returns which is as expected for a conditional variance process.

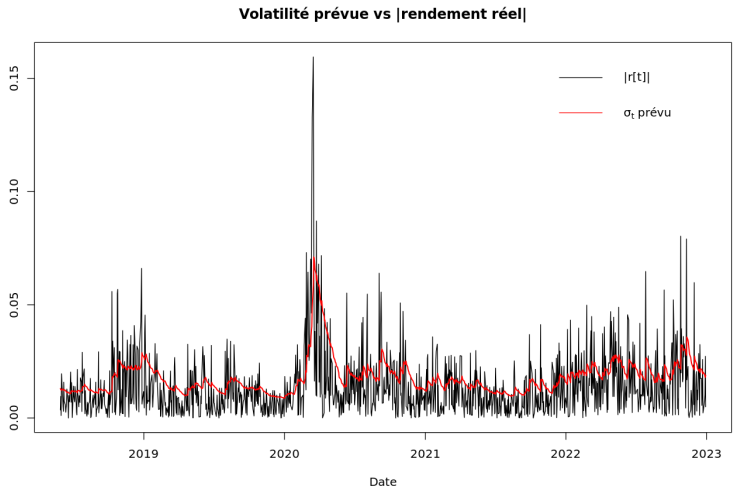

Figure 2.5: Out-of-sample comparison between predicted conditional volatility σ_t and realized absolute returns $|r_t|$. The model tracks volatility regimes while smoothing temporary market shocks.

Discussion

The ARMA$(4, 0)$–GARCH$(1, 1)$ model with Student-t innovation provides a relevant representation of Microsoft's return dynamics. It captures volatility regimes and persistence consistent with equity markets and passes standard diagnostic checks. Its main strength lies in volatility monitoring rather than return prediction as the conditional variance reacts to major market events while smoothing short-term fluctuations.

However, the model is inherently backward-looking and may not anticipate sudden shocks or structural changes. Despite these limitations, it offers a robust and interpretable baseline for risk analysis. Moreover, the GARCH$(1, 1)$ does not account for asymmetric

volatility effects where negative shocks typically have a stronger impact on volatility than positive ones.

2.5 CNN-LSTM Model

Convolutional Neural Networks combined with Long Short-Term Memory networks (CNN-LSTM) represent a powerful hybrid architecture for time series forecasting [4]. LSTM networks are a specialized type of Recurrent Neural Network (RNN) designed to capture long-term dependencies in sequential data by mitigating the vanishing gradient problem through gated memory mechanisms. This makes them particularly suitable for financial time series, where both short-term fluctuations and long-term trends coexist.

The integration of CNN layers before the LSTM component enhances the model's ability to extract local temporal patterns from raw price sequences. Convolutional filters act as feature extractors, identifying short-term price movements, local trends, and noise patterns. These extracted representations are then passed to the LSTM layer, which learns temporal dependencies and sequential relationships across longer horizons. This architecture allows the model to jointly capture short-range dynamics and long-range dependencies, which are essential for stock price prediction.

The CNN-LSTM architecture is therefore chosen to jointly capture short-term market micro-dynamics and long-term price trends, which are difficult to model simultaneously using classical linear time series models.

2.5.1 Theoretical Framework

Let $(X_t)_{t\in\mathbb{Z}}$ denote the (daily) closing price time series. The objective of a forecasting model is to approximate the one-step-ahead conditional prediction function

$$\widehat{X}_{t+1} = f_\theta(X_t, X_{t-1}, \ldots, X_{t-p}), \tag{2.3}$$

where p is the lookback window size and f_θ is a parametric function with parameters θ learned from data. Classical linear models (e.g., ARMA/ARIMA) assume that f_θ is linear in past observations, while deep learning models such as CNN-LSTM aim to learn a potentially *nonlinear* function f_θ without explicitly specifying a probabilistic structure for the data-generating process.

Convolution as learnable temporal filters. Given a univariate input sequence, a one-dimensional convolution layer can be interpreted as applying a bank of learnable linear filters over local temporal neighborhoods. For a kernel of size K, a single-filter convolution can be written (up to indexing conventions) as

$$z_t = \sum_{k=0}^{K-1} w_k \, X_{t-k} + b, \tag{2.4}$$

where $(w_k)_{k=0}^{K-1}$ are learned coefficients and b is a bias term. Equation (2.4) is closely related to classical time-series filtering (e.g., moving-average type smoothing), with the key difference that the filter weights are optimized from data. Using multiple filters allows the network to extract different short-term patterns (local trends, micro-fluctuations, and noise signatures). A nonlinearity (e.g., ReLU) is then applied to enable nonlinear feature representations.

LSTM as a nonlinear dynamical model with gated memory. The LSTM layer processes the sequence of extracted features and is designed to capture long-range dependencies through a gated memory mechanism. Denoting by x_t the input to the LSTM at time t (here, the CNN feature vector), a standard LSTM updates a hidden state h_t and a cell state c_t through gates that control information flow:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{2.5}$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{2.6}$$
$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c), \tag{2.7}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{2.8}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{2.9}$$
$$h_t = o_t \odot \tanh(c_t), \tag{2.10}$$

where $\sigma(\cdot)$ is the logistic sigmoid, $\odot$ denotes element-wise multiplication, and $W_\cdot, U_\cdot, b_\cdot$ are learned parameters. This gated structure mitigates the vanishing gradient issue and allows the model to adaptively retain or forget past information, making it suitable for financial series where both short-term variations and longer-term dynamics coexist.

2.5.2 Time-Series Interpretation of the CNN-LSTM Architecture

From a time-series standpoint, the CNN-LSTM model can be viewed as a two-stage modeling strategy:

- **Short-term pattern extraction (CNN).** The convolutional layers act as *automatic feature engineers* that detect local temporal motifs over a small number of lags (kernel size). This is conceptually related to applying multiple learnable filters, similar in spirit to handcrafted indicators (moving averages, momentum-like effects), but learned directly from data.
- **Long-term dependency modeling (LSTM).** The LSTM layer then models the temporal evolution of these extracted local patterns. In contrast with linear autoregressive models where the influence of each lag is fixed by coefficients, the LSTM can *dynamically* modulate the contribution of past information via its gates, enabling nonlinear and time-varying dependence structures.

Overall, the CNN-LSTM can be interpreted as a flexible nonlinear forecasting model that decomposes the learning task into (i) identifying relevant local structures and (ii) learning how these structures evolve across time.

2.5.3 Implicit Assumptions and Scope of Validity

Unlike ARIMA or GARCH models, CNN-LSTM does not require explicit assumptions such as strict stationarity, Gaussian innovations, or a parametric volatility equation. However, it relies on several implicit assumptions:

- **Existence of exploitable temporal dependence.** The approach assumes that past observations contain predictive information for future values, i.e., that the mapping in (2.3) is not purely noise-driven.

- **Relative stability between training and test periods.** Even if the series is non-stationary, the model implicitly assumes that the dependency patterns learned during training remain sufficiently valid during the evaluation period (no extreme regime shift).
- **Sufficient data and regular sampling.** Deep learning models typically require large datasets and consistent sampling frequency to generalize well; missing values, irregular timestamps, or short histories may degrade performance.
- **Optimization and hyperparameter sensitivity.** Model performance depends on numerical training (gradient-based optimization) and choices such as lookback window, architecture depth, and regularization; different settings may yield different local optima.

These assumptions clarify why CNN-LSTM models can perform strongly in practice for prediction tasks, while remaining less interpretable and less theoretically constrained than classical econometric approaches.

2.5.4 Data Preprocessing

Unlike classical econometric models such as ARIMA or GARCH, the CNN-LSTM framework does not require the time series to be strictly stationary. Therefore, the model was trained directly on normalized price data rather than on transformed log-returns. Using raw prices rather than returns allows the model to preserve long-term trend information, which would largely be removed by the differencing operations required in classical econometric models.

Prior to training, the closing price series was scaled using Min-Max normalization to map values into the interval $[0, 1]$. This normalization step is required to stabilize gradient-based optimization and ensure efficient training of deep neural networks. The dataset was then transformed into a supervised learning format using a sliding window approach, where a fixed number of past observations (lookback window) was used to predict the next closing price. This approach implicitly assumes that the relevant predictive information is contained within a finite past horizon. The data was split chronologically into training and testing subsets in order to preserve the temporal structure of the series and avoid data leakage.

2.5.5 Model Architecture

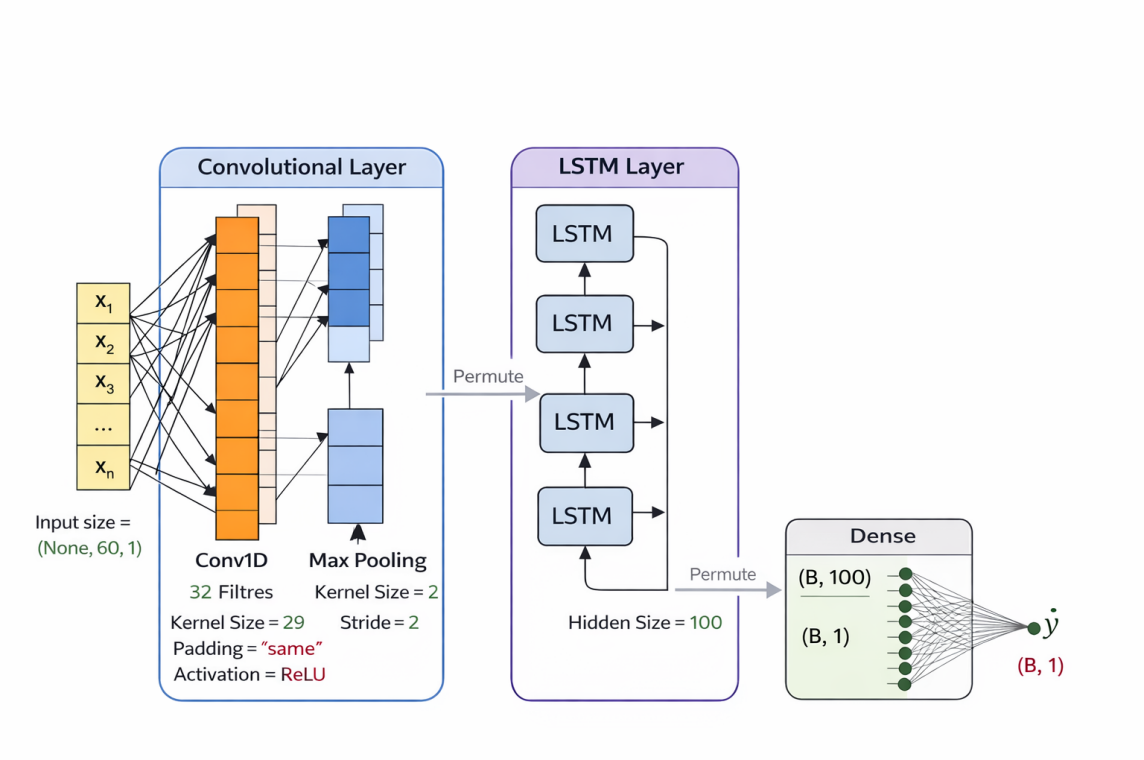

Figure 2.6: Architecture of the CNN-LSTM model used for stock price forecasting.

The CNN-LSTM architecture implemented in this work consists of the following components:

- One-dimensional convolutional layers for local feature extraction,
- A max-pooling layer to reduce dimensionality,
- An LSTM layer to capture temporal dependencies,
- Fully connected layers to generate the final prediction.

Dropout regularization was applied to reduce overfitting and improve generalization. The model was trained using the Mean Squared Error (MSE) loss function and optimized with the Adam optimizer.

2.5.6 Prediction Results and Visual Analysis

Figure 2.7 presents the comparison between actual and predicted Microsoft closing prices for both training and testing datasets. It is important to note that this figure illustrates predictions at the price level, while quantitative performance metrics are computed separately on log-returns.

The CNN-LSTM model successfully captures the global upward trend of MSFT stock while adapting to short-term fluctuations.

Although the predicted curve follows the overall trajectory of the observed prices, slight smoothing effects can be observed during periods of high volatility. This behavior

is common in deep learning regression models, which tend to minimize average prediction error rather than reproduce extreme short-term price movements.

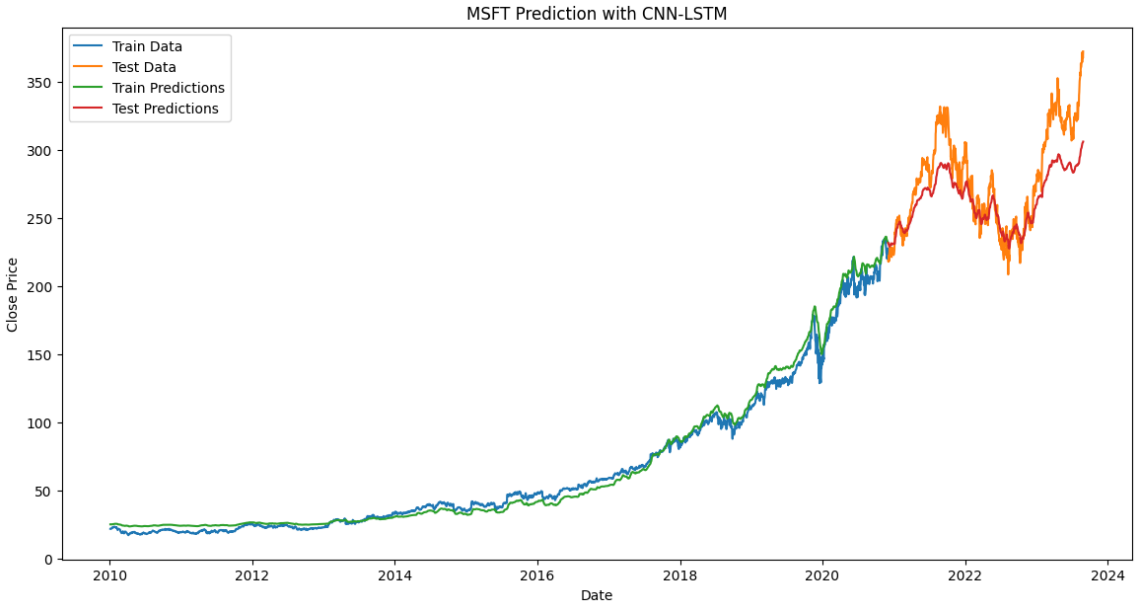

Figure 2.7: Comparison between actual and predicted MSFT closing prices using the CNN-LSTM model

2.5.7 Quantitative Evaluation

While visual inspection is performed on price-level predictions, quantitative evaluation is conducted on log-returns in order to assess forecasting accuracy on a stationary scale. Let $r_t = \log(P_t/P_{t-1})$ denote the daily log-return associated with the closing price P_t. Error metrics are computed by comparing the predicted and observed log-return series.

Metric	Training Set	Test Set
MSE	0.00362	0.00440
RMSE	0.06017	0.06630
MAE	0.04331	0.05130

Table 2.3: CNN-LSTM performance metrics computed on log-returns

The reported error magnitudes are consistent with the noisy nature of financial return series. As expected, performance slightly deteriorates from the training set to the test set, indicating reasonable generalization. Evaluating performance on log-returns enables a direct and fair comparison with classical econometric models, while the price-level forecast remains useful for visual interpretation.

These results indicate that increased model complexity does not necessarily lead to lower average prediction errors for financial returns, especially in low signal-to-noise environments.

Compared to classical linear models, the CNN-LSTM approach offers greater flexibility by modeling nonlinear relationships and complex temporal dependencies. Unlike ARIMA-based approaches, which require stationarity assumptions and struggle with long-horizon forecasting, the CNN-LSTM model directly learns patterns from raw prices.

2.5.8 Discussion and Limitations

Despite its strong predictive performance, the CNN-LSTM model presents several limitations. First, its black-box nature reduces interpretability. Unlike statistical models that explicitly decompose time series into trend and seasonal components, deep learning models do not provide transparent explanations for their predictions.

Second, CNN-LSTM models require large datasets and significant computational resources for training and hyperparameter tuning. This can limit their applicability in low-resource or real-time environments.

Finally, although the model captures long-term trends effectively, it may struggle with rare extreme market events, as these patterns are difficult to learn from historical data alone.

Nevertheless, the results obtained on the Microsoft stock dataset demonstrate that CNN-LSTM architectures are well suited for financial forecasting tasks when predictive accuracy is prioritized over interpretability.

Chapter 3

General Discussion

The task of forecasting **Microsoft (MSFT)** stock prices highlights the fundamental trade-off between the structural interpretability of classical econometric models and the superior predictive accuracy of deep learning architectures. This project compared two distinct paradigms—**ARIMA/GARCH** and **CNN-LSTM**—to evaluate their effectiveness in capturing the dynamics of a high-growth technology stock.

3.1 Limitations of Classical Autoregressive Models

Autoregressive models such as **ARIMA** are valued for their solid theoretical foundations and transparency, particularly in regulated environments where explainability is a priority. However, our analysis of MSFT revealed significant shortcomings:

- **Stationarity Constraints:** Models like ARIMA(4,0,0) require data to be stationary, necessitating a transformation into log-returns. This process often results in the loss of long-term trend information, which is a defining characteristic of MSFT's historical performance.
- **Predictive Convergence:** Forecasts for MSFT returns rapidly converged toward the unconditional mean (near zero), resulting in reconstructed price predictions that remained flat and failed to capture the actual sustained market trends driven by fundamentals or sentiment.
- **Volatility Modeling:** While **GARCH** models successfully address **volatility clustering**—as seen during the Dot-com bubble or the 2017 stable growth regime—they remain inherently backward-looking and may fail to anticipate sudden structural shifts or asymmetric shocks.

3.2 Advantages of the CNN-LSTM Architecture

The hybrid **CNN-LSTM** model addresses many of the limitations inherent in linear models by combining feature extraction with sequential memory:

- **Dual-Stage Learning:** The **CNN layers** act as automatic feature engineers, detecting local temporal patterns and micro-fluctuations, while the **LSTM layer** models long-term dependencies through gated memory mechanisms.

- **Non-Stationary Flexibility:** Unlike ARIMA, the CNN-LSTM handles non-stationary data directly, allowing the model to be trained on **normalized raw prices** rather than returns, thereby preserving the global upward trajectory of MSFT.
- **Predictive Flexibility:** While quantitative evaluations on log-returns indicate that classical linear models achieve lower average errors, the CNN-LSTM architecture offers greater flexibility by capturing nonlinear relationships and complex temporal dependencies, particularly at the price level.

3.3 Challenges: Interpretability and Complexity

Despite its predictive power, the CNN-LSTM is a **"black-box"** model. It lacks the transparency of ARIMA, which decomposes series into identifiable components like trend and seasonality. Furthermore, these models are **computationally intensive**, require large high-quality datasets to avoid **overfitting**, and are highly sensitive to hyperparameter tuning.

To bridge the gap between these paradigms, future research could explore:

1. **Hybrid Frameworks:** Combining models, such as **GARCH-LSTM**, to leverage the volatility-aware rigor of econometrics alongside the non-linear flexibility of deep learning.
2. **Explainability Techniques:** Implementing tools like **SHAP** (SHapley Additive exPlanations) or **attention mechanisms** to enhance the transparency of neural network predictions for stakeholders and regulators.
3. **Exogenous Variables:** While initial tests on MSFT showed that including GDP or inflation did not improve GARCH performance, integrating news sentiment or more granular economic indicators remains a potential area for enrichment.

In conclusion, while **CNN-LSTM** provides state-of-the-art accuracy for MSFT price forecasting, **ARIMA and GARCH** remain indispensable for risk monitoring and providing an interpretable baseline for short-term market dynamics.

Conclusion

In conclusion, this project successfully developed and compared three distinct modeling paradigms—**ARIMA, GARCH, and CNN-LSTM**—to forecast the stock prices of **Microsoft Corporation (MSFT)** using historical data from 2000 to 2023. The study highlights a fundamental trade-off between the structural interpretability of classical econometric methods and the superior predictive power of deep learning architectures. By analyzing these different approaches, the research provides a comprehensive framework for understanding how various models respond to the unique challenges of a high-growth technology stock characterized by significant volatility and non-linear trends.

The application of classical models demonstrated that while **ARIMA** serves as a baseline for modeling the conditional mean and capturing short-term linear dependencies, it remains insufficient for financial time series when used in isolation. The failure of ARIMA to account for the pronounced **volatility clustering** and heavy-tailed distributions inherent in MSFT returns necessitated the extension to the **ARMA-GARCH** framework. By utilizing Student-t innovations, the ARMA-GARCH model provided a much more realistic representation of market risk, effectively tracking periods of high-impact stress, such as the dot-com bubble and the 2008 financial crisis, while smoothing temporary market shocks.

In contrast, the hybrid **CNN-LSTM** architecture emerged as the most effective model for maximizing predictive accuracy. By integrating convolutional layers for local feature extraction with LSTM layers for long-term dependency modeling, this approach successfully captured MSFT's global upward trajectory and complex market micro-dynamics. Furthermore, the CNN-LSTM model offered greater flexibility by processing **raw adjusted closing prices** directly, thereby preserving essential trend information that is typically lost through the stationarity transformations required by ARIMA and GARCH. Quantitatively, the evaluation on log-returns shows that classical linear models such as ARIMA achieve lower error values in terms of MAE, RMSE, and MSE. This result highlights the good performance of parsimonious linear models for short-horizon return forecasting, while the CNN-LSTM architecture remains particularly effective for modeling price-level dynamics and long-term trends.

Ultimately, the study concludes that for a dynamic asset like Microsoft, both modeling paradigms are complementary depending on the specific objective. Deep learning models are preferred when **predictive precision** is the priority, whereas ARIMA and GARCH remains essential for **regulatory transparency**, risk monitoring, and identifying identifiable components like trend and seasonality. Future research should investigate hybrid frameworks, such as **GARCH-LSTM**, to leverage the interpretability of classical econometrics alongside the non-linear flexibility of deep learning to further enhance financial forecasting.

Appendix

Failure of ARIMA to Forecast MSFT Stock Prices

Forecasting Performance

Figure 1 illustrates the forecasting behavior of the ARIMA$(4, 0, 0)$ model. As expected for a stationary return series, forecasts rapidly converge toward the unconditional mean, which is close to zero. The confidence intervals widen with the forecast horizon, reflecting the accumulation of uncertainty over time.

The forecast path remains nearly flat, while actual returns exhibit substantial variability and volatility clustering. This highlights that ARIMA models capture the conditional mean but fail to account for time-varying volatility. Consistent with the efficient market hypothesis, return predictability is very limited, and ARIMA forecasts provide little improvement over a constant mean prediction.

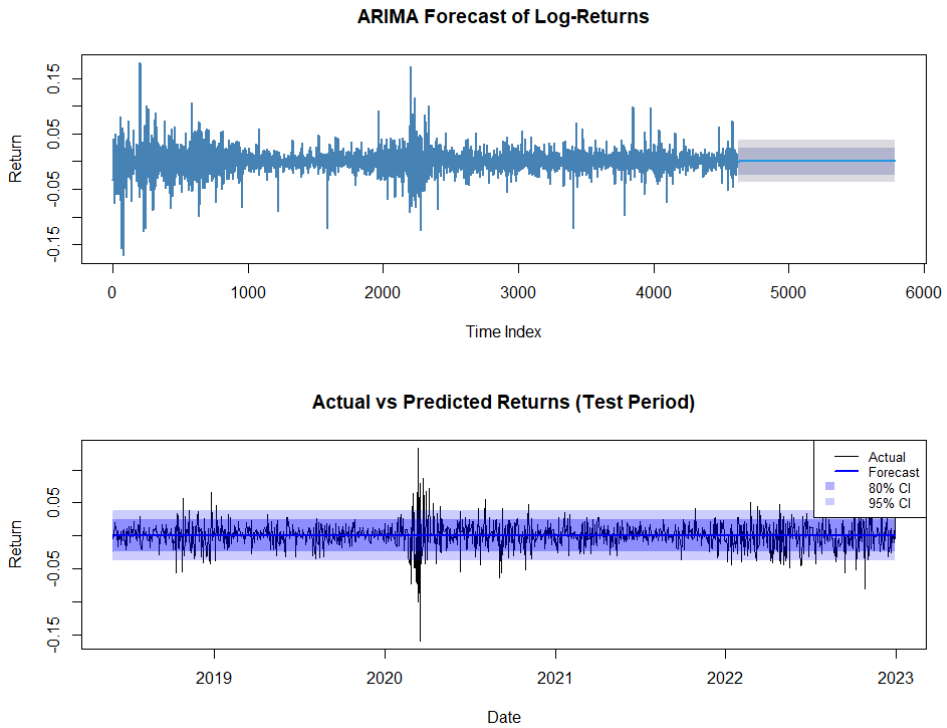

Figure 1: ARIMA$(4, 0, 0)$ forecasts of MSFT log-returns with 80% and 95% confidence intervals. Top: in-sample series and out-of-sample forecasts. Bottom: actual versus predicted returns over the test period.

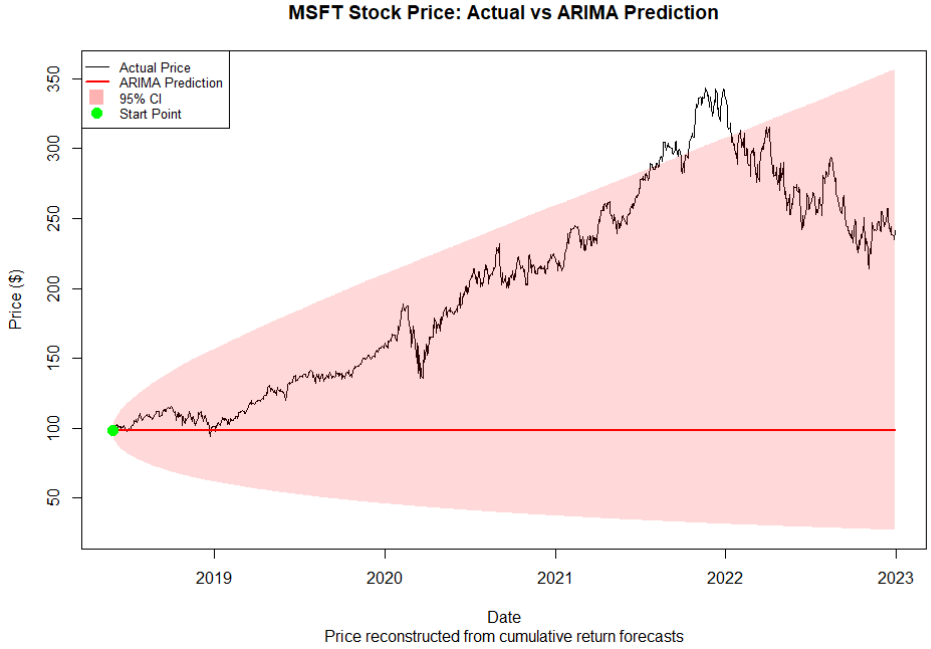

Figure 2: MSFT stock price reconstructed from cumulative ARIMA$(4, 0, 0)$ return forecasts with 95% confidence intervals.

Figure 2 presents the MSFT stock price reconstructed from cumulative ARIMA return forecasts. The predicted price remains approximately constant over the forecast horizon, as return forecasts rapidly converge toward a zero unconditional mean, implying no systematic price trend.

In contrast, the actual MSFT price rose from about \$100 in 2018 to nearly \$340 in 2021, before declining to around \$240 by the end of 2022, dynamics entirely missed by the ARIMA forecast. The 95% confidence interval widens with the forecast horizon due to the accumulation of uncertainty, and the observed price exceeds the upper bound as early as mid-2020, highlighting the inability of ARIMA models to capture sustained market trends driven by fundamentals or sentiment.

Implementation of CNN–LSTM

This appendix reports the core neural network architecture used in this study. Only the CNN–LSTM model definition is presented, while data preprocessing, training loops, and optimization procedures are omitted for clarity.

The model combines one-dimensional convolutional layers for short-term temporal feature extraction with an LSTM layer designed to capture longer-term dependencies. This hybrid structure enables the model to learn both local price patterns and global temporal dynamics directly from price sequences.

CNN–LSTM Architecture

```
class CNNLSTMModel(nn.Module):
    def __init__(self, input_size=1, hidden_size=50):
        super(CNNLSTMModel, self).__init__()
```

```
        # Convolutional layers
        self.conv1 = nn.Conv1d(
            in_channels=1,
            out_channels=32,
            kernel_size=3,
            stride=1,
            padding=1
        )
        self.conv2 = nn.Conv1d(
            in_channels=32,
            out_channels=64,
            kernel_size=3,
            stride=1,
            padding=1
        )
        self.maxpool = nn.MaxPool1d(kernel_size=2, stride=2)
        self.dropout = nn.Dropout(0.4)
        # LSTM layer
        self.lstm = nn.LSTM(
            input_size=64,
            hidden_size=hidden_size,
            num_layers=1,
            batch_first=True
        )
        # Fully connected output layer
        self.fc = nn.Linear(hidden_size, 1)

    def forward(self, x):
        # x shape: (batch_size, 1, sequence_length)

        # Convolutional feature extraction
        x = F.relu(self.conv1(x))
        x = F.relu(self.conv2(x))
        x = self.maxpool(x)
        x = self.dropout(x)

        # Prepare input for LSTM
        x = x.permute(0, 2, 1)  # (batch_size, seq_len, features)
        # LSTM forward pass
        out, _ = self.lstm(x)
        # Use the last time step output
        out = self.fc(out[:, -1, :])
        return out
```

Bibliography

[1] Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control* (5th ed.). Wiley.

[2] Engle, R. F. (1982). *Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of United Kingdom Inflation. Econometrica*, 50(4), 987–1007.

[3] Bollerslev, T. (1986). *Generalized Autoregressive Conditional Heteroskedasticity. Journal of Econometrics*, 31(3), 307–327.

[4] Zhang, Y., Liu, B., Zhang, Y., & Wang, S. (2020). *A Hybrid CNN–LSTM Model for Forecasting Stock Prices. Mathematical Problems in Engineering.*