National School of Statistics and Data Analysis (ENSAI)

Report

Stock Price Prediction

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

---

Supervised by:
Dr. Youssef Esstafa

Prepared by:
Boris Junios Fèmi AGOSSOU
Samuel MARTHELY
Fernand MBAIORNOM MBAIHODJI
Jules murphy aristote MOBELI
Sonokoli SORO

Class of 2025

January 2025

Contents

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. One of the most widely followed stock market indices is the *S&P 500*, which reflects the performance of the 500 largest companies listed on stock exchanges in the United States. In this project, we focus on modeling and forecasting the *S&P 500* using historical data of its adjusted closing prices.

The general approach involves transforming the raw price data into returns in order to achieve stationarity, a property that is crucial for the validity of most time series models. By working with log-returns instead of prices, we aim to stabilize the variance and reduce the influence of non-stationary behavior that typically characterizes raw financial series. Once the return series is obtained, standard diagnostic tools (such as stationarity tests) are applied to verify that the mean and variance of the process remain constant over time.

To gain insights into different modeling perspectives, we will explore three distinct classes of models:

- **ARIMA models**, which capture linear dependencies in the data through autoregressive and moving average components, while also addressing integration to enforce stationarity.
- **GARCH models**, which focus on estimating time-varying volatility patterns, commonly observed as volatility clustering in financial markets.
- **LSTM networks**, a type of recurrent neural network well-suited for sequence modeling, providing a more flexible and potentially nonlinear approach to capturing complex patterns in the data.

Our objective is not only to compare the forecast performance of these methods on the *S&P 500* returns, but also to highlight the underlying assumptions and potential advantages each model brings to time series analysis. This introduction lays the groundwork for our methodology, which will be discussed in detail in subsequent sections. There, we will examine the data preparation, the steps taken to check and ensure stationarity, and the model-building process for each of the three approaches. Finally, we will compare the forecasting accuracy of the models and provide a brief conclusion outlining the key findings of this study.

1 DATA AND METHODOLOGICAL FRAMEWORK

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

1.1 Exploratory analysis

1.1.1 Composition of the 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 data for the S&P 500 index (`GSPC`) from January 1, 2000, to December 31, 2023.

The dataset was clean and did not contain any missing values, which allowed us to proceed without additional imputation or data handling steps.

The dataset contains the following key information:

- **Date:** The trading date for each record.
- **Open:** The opening price of the S&P 500 index on the respective trading day.
- **High:** The highest price recorded during the trading day.
- **Low:** The lowest price recorded during the trading day.
- **Close:** The closing price of the index for the trading day.
- **Volume:** The total number of shares traded during the day.
- **Adjusted Close:** The closing price adjusted for dividends and splits.

1.2 Feature Choice

1.2.1 Adjusted Close

In this study, we selected the adjusted closing price of the S&P 500 index as the primary feature for analysis. This choice was motivated by several factors:

- **Relevance for Analysis:** The adjusted closing price reflects the final trading price of the index for a given day, adjusted for dividends and stock splits, providing a more accurate representation of the index's performance over time.
- **Market Trend Representation:** It serves as a reliable indicator for identifying long-term market trends and patterns.
- **Comparison Across Time:** By accounting for corporate actions such as splits and dividends, the adjusted closing price allows for consistent comparison across different time periods.

Using the adjusted closing price as our primary variable, we aim to capture meaningful insights into the historical trends and behavior of the S&P 500 index.

The plot below illustrates the historical adjusted closing price of the S&P 500 index from January 1, 2000, to December 31, 2023.

Figure 1: Historical Adjusted Closing Price of the S&P 500 Index (2000-2023)

Analysis of Time Series Stationarity

At first glance, the original time series of the S&P 500 adjusted closing prices does not appear to be stationary. This is evident from the increasing trend and the variability over time, which suggests that the mean and variance are not constant.

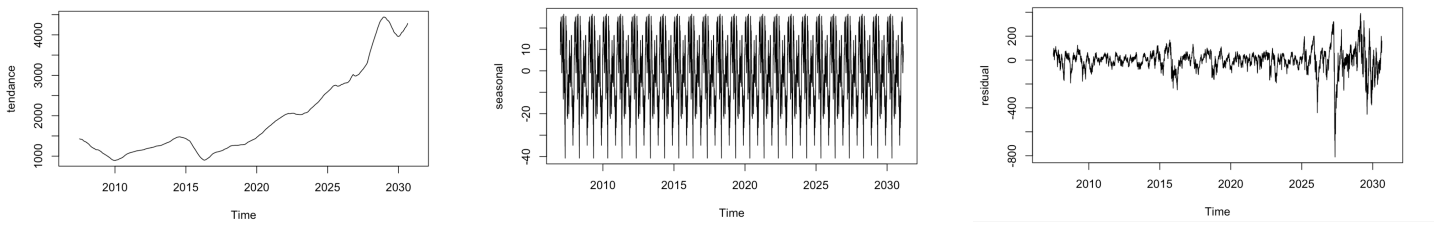
To further investigate this, we decomposed the series into its components: trend, seasonal, and residual. This decomposition provided the following insights:

- **Trend:** The trend component shows a clear upward movement over the years, reflecting the overall growth in the S&P 500 index.
- **Seasonal:** The seasonal component indicates repeating patterns over a fixed period, which could be attributed to cyclical market behavior or external factors.
- **Residual:** The residual component captures the irregular fluctuations that cannot be explained by the trend or seasonality.

The decomposition supports the initial observation that the series is not stationary due to the presence of a strong trend component.

To formally confirm this, we conducted an Augmented Dickey-Fuller (ADF) test on the time series. The test returned a p-value above the critical threshold (e.g., 0.05), confirming the null hypothesis that the series is non-stationary.

Below are the plots illustrating the original series and its decomposition:



(a) Trend Component (b) Seasonal Component (c) Residual Component

Figure 2: Decomposition of the Time Series into Trend, Seasonal, and Residual Components

1.2.2 Log return

As observed earlier, the Adjusted Closing Price series is non-stationary, evidenced by the clear upward trend in Figure 1. This non-stationarity is further confirmed by the Augmented Dickey-Fuller (ADF) test, which yields a p-value of 0.99, failing to reject the null hypothesis of a unit root.

This poses a challenge because classical time series models like ARIMA and GARCH assume that the underlying series is stationary. Without stationarity, the statistical properties required for these models do not hold, potentially leading to unreliable forecasts and inferences.

To address this issue, we transform the data to achieve stationarity. We choose to use logarithmic returns, which are commonly employed in financial time series analysis. Logarithmic returns are preferred over simple returns for several reasons:

- **Additivity over time:** Log returns are additive across time periods, simplifying the aggregation of returns over multiple intervals.
- **Variance stabilization:** They tend to stabilize the variance, making the series more suitable for models that assume homoscedasticity.
- **Symmetry:** Log returns treat upward and downward movements symmetrically, which is beneficial for modeling purposes.

The logarithmic return r_t at time t is calculated as:

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

where P_t is the adjusted closing price at time t.

After applying the log transformation, the resulting series appears stationary, as illustrated in Figure 4. Visually, the log returns fluctuate around a constant mean with relatively constant variance, lacking the trend present in the original series. Additionally, an ADF test on the log returns yields a p-value less than 0.01, allowing us to reject the null hypothesis of a unit root. This confirms that the transformed series is stationary.

Therefore, for the application of classical time series models such as ARIMA and GARCH, we will use the log returns of the Adjusted Closing Prices. This transformation ensures that the stationarity assumptions required by these models are satisfied, allowing for more reliable modeling and forecasting.

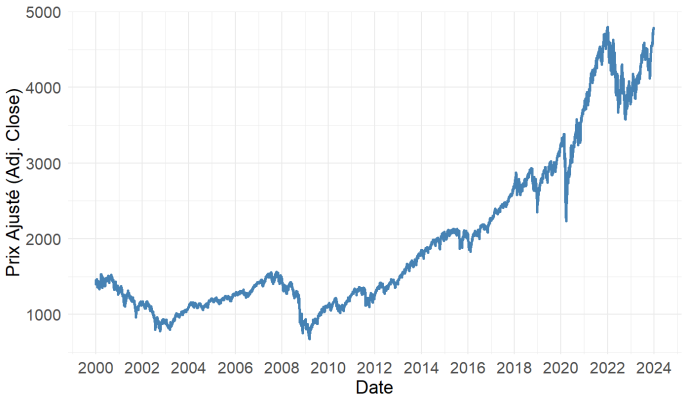


Figure 3: Adjusted Closing Price Over Time

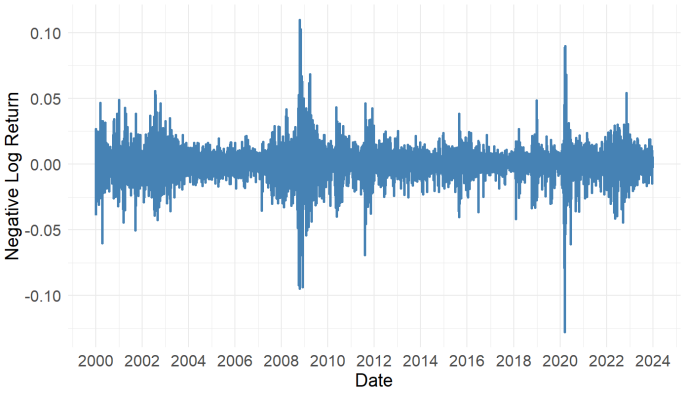


Figure 4: Log Returns Over Time

1.3 Feature Engineering

During the feature engineering process, we constructed several technical indicators, such as the 7-period Average True Range (ATR) to capture market volatility and the 5-period Chande Momentum Oscillator (CMO) to measure price momentum. Additionally, we incorporated exogenous economic variables, including GDP, inflation, and interest rates, to provide external context to our models.

However, when testing these features with models such as GARCH, we found that their inclusion did not improve predictive performance. In some cases, it even worsened predictions, likely due to increased noise or limited relevance to short-term market movements. Consequently, to maintain model simplicity and interpretability, we opted to exclude these external factors and focus on core features like log returns and selected technical indicators for our analysis.

2 PRESENTATION OF RESULTS

2.1 MODEL ESTIMATION AND EVALUATION

2.1.1 ARIMA Model

The Autoregressive Integrated Moving Average (ARIMA) model combines three key components: autoregression (AR), differencing (I), and moving averages (MA). The goal is to transform non-stationary data into a stationary process, enabling accurate modeling and forecasting.

2.1.1.1 Mathematical Formulation: The ARIMA(p, d, q) model is mathematically represented as:

$$\phi(B)(1-B)^d y_t = \theta(B)\epsilon_t$$

where: y_t is the observed value at time t, B is the backshift operator, such that $B^k y_t = y_{t-k}$, $(1-B)^d$ is the differencing operator of order d to achieve 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 represents a white noise process.

2.1.1.2 Assumptions of the Model: The ARIMA model relies on specific assumptions about the error term ϵ_t, which is assumed to follow a weak white noise process: 1. Zero Mean: $E[\epsilon_t] = 0$ for all $t \in \mathbb{Z}$. 2. Constant Variance: $E[\epsilon_t^2] = \sigma_\epsilon^2$, where σ_ϵ^2 is constant over time. 3. Uncorrelated Errors: $\text{Cov}(\epsilon_s, \epsilon_t) = 0$ for all $s \neq t$.

2.1.1.3 Steps in ARIMA Modeling: 1. **Identification**: Determine the appropriate orders p, d, and q by analyzing the properties of the data using tools like the autocorrelation function (ACF) for q and partial autocorrelation function (PACF) for p. 2. **Estimation**: Estimate the parameters ϕ_i and θ_j using techniques like maximum likelihood estimation (MLE). The likelihood expression is generally too complex to obtain an explicit maximum, so numerical algorithms (such as Newton's method) are used 3. **Diagnostic Checking**: Validate the model by analyzing the residuals to ensure they exhibit properties of white noise.

2.1.1.4 Relevance to this Study: In this work, the ARIMA model is applied to the adjusted closing prices of the SP500 index. The objective is to capture the temporal dynamics of the series and forecast future values. Results from the application of the ARIMA model are presented in the following sections.

The figures 5 and 6 below illustrate the autocorrelation function (ACF) and the partial autocorrelation function (PACF) of the log-returns of the time series, respectively, computed on the training data.

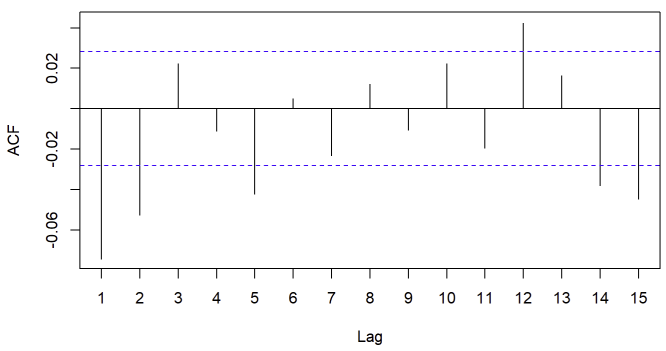


Figure 5: ACF of log-returns.

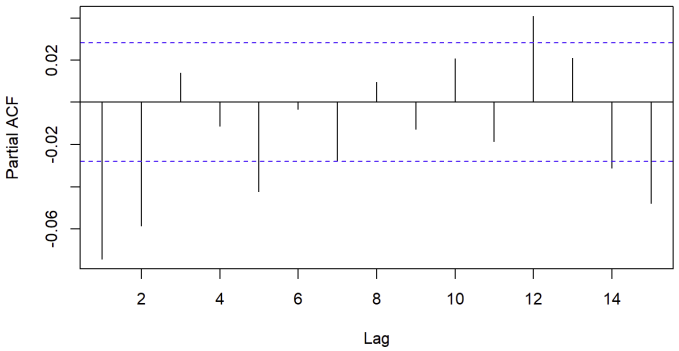


Figure 6: PACF of log-returns.

With the series already stationary, we tested all ARIMA$(p,0,q)$ models where $p \in \{0,1,\ldots,p_{\max}\}$ and $q \in \{0,1,\ldots,q_{\max}\}$. The selection of $p_{\max} = 2$ and $q_{\max} = 2$ was based on the Partial Autocorrelation Function (PACF) and Autocorrelation Function (ACF) plots shown in Figures 5 and 6.
These plots indicate that the significant lags are up to 2 for both the AR and MA components, as the values beyond lag 2 exhibit negligible correlations. This justified limiting the range of p and q for model testing.
Based on the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC), we identified the optimal model as ARIMA(2,0,0), as it minimizes both the AIC and BIC values. This choice ensures a balance between model complexity and performance.

To validate this model, we performed a diagnostic check of the residuals to ensure they behave as white noise. Specifically, we tested the hypothesis of uncorrelated residuals using the Box-Pierce test. The results of the test are as follows:

```
Box-Pierce Test:
data: arima200$resid
X-squared = 0.0046363, df = 1, p-value = 0.9457
```

These results confirm that the residuals are uncorrelated, validating the adequacy of the ARIMA(2,0,0) model.

The summary of the selected ARIMA model is presented below in Figure 7.

```
ARIMA(2,0,0) with zero mean

Coefficients:
          ar1      ar2
      -0.0786  -0.0584
s.e.   0.0144   0.0144

sigma^2 = 0.0001438:  log likelihood = 14510.66
AIC=-29015.31   AICc=-29015.31   BIC=-28995.86

Training set error measures:
                      ME       RMSE         MAE MPE MAPE      MASE         ACF1
Training set 0.0001559237 0.01198813 0.008007641 NaN  Inf 0.6684824 0.0008116146
```

Figure 7: Summary of the ARIMA(2,0,0) Model

2.1.1.5 Evaluation Metrics on the Test Data To evaluate the performance of the ARIMA(2,0,0) model on the test data, we computed several metrics including the Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). These metrics allow us to assess the accuracy of the model in predicting the log-returns. The metrics for the test data are summarized in Table 1. Additionally, we report the AIC and BIC values to provide insight into the overall goodness of fit of the model. By comparing these metrics with those obtained on the training data, we can verify whether the model generalizes well or suffers from overfitting.

Table 1: Evaluation Metrics for ARIMA(2,0,0)

Metric	**Training Data**	**Test Data**
MAE	0.008007641	0.008895012
MSE	0.0001438	0.0001862294
RMSE	0.01198813	0.01364659
AIC	-29015.31	-29014.14
BIC	-28995.86	-28988.21

The comparison of the metrics indicates that the model performs consistently across both the training and test datasets. The slight increase in MAE, MSE, and RMSE for the test data is expected and falls within an acceptable range. The close values of AIC and BIC between the training and test datasets further suggest that the model does not suffer from overfitting and generalizes well to unseen data.

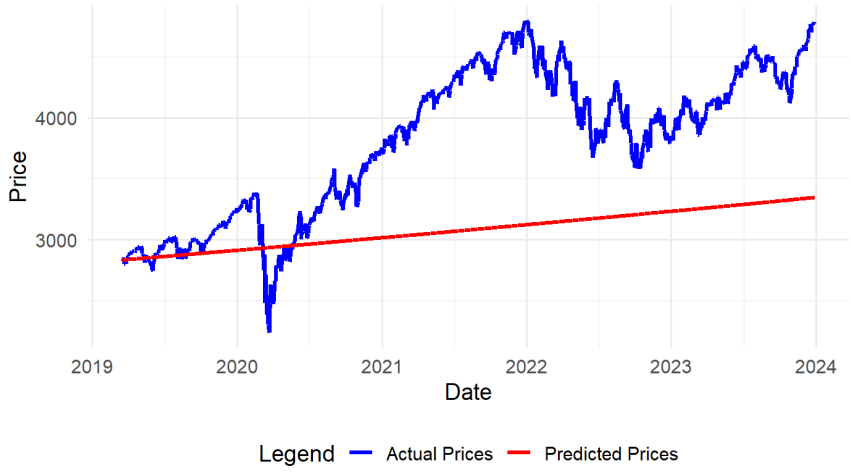


Figure 8: Comparison of actual and predicted prices on test data using the ARIMA(2,0,0) model

The figure illustrates the comparison between the actual prices (blue solid line) and the predicted prices (red solid line) over time using the ARIMA(2,0,0) model. While the predicted prices closely follow the actual prices at the beginning of the time series, the model's accuracy deteriorates as we move further into the future. The red line representing predictions deviates progressively from the actual prices, indicating that the ARIMA(2,0,0) model struggles to capture the underlying patterns or external factors influencing price fluctuations over longer time horizons.

Discussion of ARIMA Results

The ARIMA(2,0,0) model was applied to the log-returns of the adjusted closing prices of the S&P 500 index. While the model demonstrated reasonable performance metrics on both the training and test datasets, certain limitations were observed when analyzing the reconstructed prices derived from the predicted log-returns. Specifically, the predicted log-returns tend to converge closely to zero, failing to capture the fluctuations around this value. This behavior is reflected in the reconstructed prices, which exhibit a smooth, linear trend that diverges from the actual prices, as shown in Figure 8. The actual prices display significant volatility and non-linear movements, which are not captured by the ARIMA model.

This limitation can be attributed to several factors. First, log-returns for financial time series, particularly for large indices like the S&P 500, are generally very small in magnitude and close to zero. As a result, even minor errors in predicting log-returns can lead to significant deviations when reconstructing actual prices. Second, the ARIMA model, being linear in nature, relies heavily on past values and residuals for predictions. While this approach is effective for stationary series, it struggles to account for the inherent volatility and complex dependencies characteristic of financial markets. Finally, the reconstructed prices fail to reflect the high volatility observed in the actual prices, highlighting the ARIMA model's limitation in handling abrupt fluctuations.

These observations are not unexpected given the modeling approach and the nature of the data. Log-returns are primarily used to stabilize variance and achieve stationarity, making them suitable for capturing trends rather than short-term fluctuations. The smoothness of the predicted prices reflects the ARIMA model's tendency to prioritize overall trends over local variations.

These findings underscore the importance of aligning modeling objectives with the choice of features and model structures. While ARIMA models are useful for stationary series and trend forecasting, they may not be the best choice for highly volatile data or when accurate short-term predictions are required. To address these

challenges, future work could explore alternative approaches, such as combining ARIMA with GARCH models to account for volatility or leveraging nonlinear models like Long Short-Term Memory (LSTM) networks to capture complex patterns. Despite its limitations, the ARIMA(2,0,0) model provides a reasonable approximation of the overall trend in log-returns, but it highlights the need for more robust models to capture the nuances of financial time series data.

2.2 ARMA-GARCH MODEL

GARCH (Generalized Autoregressive Conditional Heteroskedasticity) models are an extension of the ARCH (Autoregressive Conditional Heteroskedasticity) models introduced by Engle in 1982 [2], and generalized by Bollerslev in 1986. They are mainly used in econometrics and finance to model and forecast the conditional volatility of financial time series, such as stock market returns.
GARCH models are particularly well suited to capturing the dynamic dependence of financial returns. Indeed, financial returns exhibit heteroskedastic volatility, i.e. the variance of returns varies over time.

2.2.1 Theorical fondations

Mathematical Formulation: The ARMA-GARCH model consists of two components:

1. The yield time series equation :

$$r_t = \mu + \epsilon_t$$

where:

- r_t is the yield or variable of interest at time t,
- μ is the conditional mean (often assumed to be constant),
- ϵ_t is the residual (conditionally heteroscedastic white noise).

2. The conditional variance equation :

$$\epsilon_t = \sigma_t \eta_t$$

$$\sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2$$

where:

- σ_t^2 is the conditional variance at time t,
- $(\eta_t)_t$ is a sequence of i.i.d. random variables, assumed to be distributed according to a centered normal distribution.
- $\omega > 0$, $\alpha_i \geq 0$, and $\beta_j \geq 0$ are the parameters to be estimated,
- α_i measures the ARCH effect (past shocks ϵ_{t-i}^2 on the variance),
- β_j measures the GARCH effect (past variances σ_{t-j}^2).

Assumptions of the Model: The ARMA-GARCH model relies on certain assumptions, which we can classify as follows:

1. **Residue structure assumptions**

- Conditional heteroskedasticity: The conditional variance σ_t^2 of the residuals ϵ_t varies over time and depends on past information : $\mathbb{E}[\epsilon_t^2 \mid \mathcal{F}_{t-1}] = \sigma_t^2$
- Absence of unconditional heteroscedasticity : The series must have a constant unconditional variance (stationarity in variance).

- Law of standardized residuals: The residuals η_t must be independent and identically distributed (i.i.d.) according to a centered normal distribution or a t-Student distribution.

2. **Stationarity assumptions** To guarantee second-order stationarity of the epsilon process, the coefficients of the GARCH components must satisfy the following constraint:

$$\sum_{i=1}^{q} \alpha_i + \sum_{j=1}^{p} \beta_j < 1$$

3. **Assumptions on data**

   - Series with no strong trend : The series must be stationary on average (no obvious trend).
   - Presence of volatility clusters: The data should show conditional volatility, i.e. periods of high volatility followed by periods of calm.
   - No autocorrelation in returns: The returns themselves should be non-autocorrelated (white noise), but their squares or absolute values may exhibit autocorrelation, signalling conditional heteroskedasticity.

Steps in ARMA-GARCH Modeling:

1. **Fitting an ARMA model for the mean:** An ARMA(p,q) model is fitted to the series to capture the dependence in the mean. The residuals ϵ_t from this model are extracted for further analysis.

2. **Identifying orders p and q for the GARCH model:** The squared residuals are analyzed as an ARMA process. The following tools are used:

   - The autocorrelation function (ACF) of ϵ_t^2 provides clues about q, the number of ARCH terms.
   - The partial autocorrelation function (PACF) of ϵ_t^2 helps identify p, the number of GARCH terms.

   Based on these diagnostics, upper bounds $p_{\max}$ and $q_{\max}$ are determined. A systematic grid search is then performed over all combinations of p and q, selecting the best model according to an information criterion (e.g., AIC or BIC).

3. **Parameter Estimation:** The parameters ϕ_i, θ_j, α_i, and β_j are estimated using maximum likelihood estimation (MLE), often with numerical optimization techniques.

4. **Model Diagnostics and Validation:** After fitting the ARMA-GARCH model, the standardized residuals $\eta_t = \epsilon_t/\sigma_t$ are analyzed. These residuals should exhibit properties of independent and identically distributed (i.i.d.) noise with constant variance. Ljung-Box tests on η_t and η_t^2 are conducted to ensure no remaining dependence in the mean or variance.

2.2.2 Selection of parameters

2.2.2.1 ARCH-LM Test :

The Autoregressive Conditional Heteroskedasticity Lagrange Multiplier (ARCH-LM) test is a statistical test first introduced in [2] used to detect the presence of ARCH effects in a time series. ARCH effects refer to time-varying volatility, where the variance of residuals is not constant but depends on past squared residuals. This

behavior is common in financial time series, such as returns, where periods of high and low volatility often occur in clusters.

- **Null Hypothesis (H_0)**: No ARCH effects are present (residuals have constant variance, or homoscedasticity).
- **Alternative Hypothesis (H_1)**: ARCH effects are present (variance depends on past squared residuals, or heteroscedasticity).

The results of the ARCH-LM test conducted on the residuals (`resid_arma`) of the ARMA(2,0) model are as follows:

- Chi-squared statistic: 1397.9
- Degrees of freedom (df): 12
- p-value: $< 2.2 \times 10^{-16}$

The p-value is extremely small (< 0.05), leading to a rejection of the null hypothesis H_0 of no ARCH effects. This indicates the presence of significant time-varying volatility (heteroscedasticity) in the residuals of the ARMA(2,0) model. Therefore, while the ARMA(2,0) model adequately captures the dependence in the mean, it fails to address the conditional heteroskedasticity in the series.

Given the presence of ARCH effects, we proceed to estimate an ARMA-GARCH model. The ARMA(2,0) component, already estimated in the previous step, will remain unchanged and will serve as the mean equation in the ARMA-GARCH model. The GARCH component will be added to capture the conditional heteroscedasticity of the residuals.

2.2.2.2 Estimation of garch orders :

To determine the appropriate orders p and q for the GARCH model, we analyzed the squared residuals, as suggested by the theoretical relationship between a GARCH(p,q) process and an ARMA(max(p,q),q) model . Specifically, we examined the autocorrelation function (ACF) and partial autocorrelation function (PACF) to identify potential lags for the ARCH (q) and GARCH (p) terms.

However, interpreting the ACF and PACF plots proved challenging, as they exhibited significant correlations across multiple lags. Consequently, we selected a reasonable range of $p,q \in \{0,1,2,3,4,5\}$ for testing and relied on the Akaike Information Criterion (AIC) to identify the best model.

Using this approach, the model selection process yielded the following result:

Best ARMA : ARMA(5,5)

Based on these results, we proceed to fit an ARMA(2,0)-GARCH(5,5) model, where:

- The ARMA(2,0) component, estimated previously, captures the dependence in the mean of the series.
- The GARCH(5,5) component is added to model the conditional heteroskedasticity in the variance.

This combined model accounts for both the dynamic structure in the mean and the time-varying volatility observed in the series.

2.2.2.3 Estimation of the model :

After fitting the ARMA(2,0)-GARCH(5,5) model, we evaluated its adequacy using several diagnostic tests. One of the key tests performed was the Ljung-Box test, which is designed to detect the presence of autocorrelation in a time series. Introduced by Ljung and Box in their seminal paper (Ljung Box, 1978) in [6], this test examines whether any group of autocorrelations up to a specified lag is significantly different from zero.

In our analysis, the Ljung-Box test was applied to the standardized residuals and squared residuals of the model. The results suggest no significant serial correlation in the mean or variance, as all p-values are above conventional significance levels (e.g., 0.05). This indicates that the ARMA-GARCH model has adequately captured the autocorrelation structure in the data.

The weighted ARCH LM test further confirms the absence of remaining ARCH effects, with p-values consistently above 0.05 across multiple lags. Additionally, the Nyblom stability test results show no evidence of parameter instability, as the joint and individual statistics remain below the critical thresholds.

Furthermore, we explored the inclusion of external economic factors to improve the model's performance. Various tests were conducted by incorporating these factors into the model; however, their impact was found to be negligible or even counterproductive, leading to a poorer fit. In light of these results, and following the principle of parsimony, we opted to retain the simpler ARMA(2,0)-GARCH(5,5) model without external factors.

In conclusion, the ARMA-GARCH(2,0,5,5) model successfully addresses the dynamic structure in both the mean and variance, as evidenced by the absence of significant residual correlations and ARCH effects. However, further refinements may be necessary to improve the overall fit, particularly in capturing extreme values or tail behavior.

2.2.3 Evaluation of the model

2.2.3.1 Metrics Now we will analyze metrics

Table 2: Evaluation Metrics for ARMA(2,0)-GARCH(5,5)

Metric	Value
MAE	0.008876137
MSE	0.0001861551
RMSE	0.01364387
AIC	-31275.1
BIC	-31184.35

Table 2 presents the evaluation metrics for the ARMA(2,0)-GARCH(5,5) model. The Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) values are notably low, reflecting the inherent characteristics of log-returns, which tend to have very small magnitudes. These metrics indicate that the model performs well in terms of predictive accuracy.

Compared to the ARIMA(2,0,0) model, the differences in MAE, MSE, and RMSE are minimal, as both models produce similarly low errors due to the small variance in log-returns data. However, the ARMA-GARCH model demonstrates a clear advantage in terms of the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC). The lower AIC and slightly improved BIC suggest that the ARMA-GARCH model provides a better fit to the data by capturing both the dependence in the mean and the conditional heteroskedasticity in the variance.

2.2.3.2 Visualisation :

In this section, we present and interpret the results of the ARMA-GARCH(2,0,5,5) model based on the provided data. We analyze the predictive performance and variance dynamics captured by the model and compare it to the simple ARMA model.

Comparison of Real and Predicted Prices

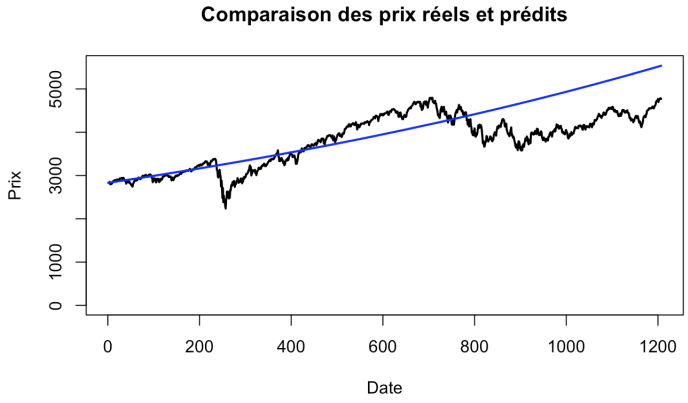


Figure 9: Comparison of Real and Predicted Prices for ARMA-GARCH(2,0,5,5)

Figure 9 shows the comparison between the real prices (black line) and the predicted prices (blue line) using the ARMA-GARCH(2,0,5,5) model. The predicted prices show a smoother trend compared to the actual prices, as expected, due to the inherent volatility in the real data. However, the ARMA-GARCH model appears to better capture the overall dynamics of the series compared to the simple ARMA model. The inclusion of the

GARCH component allows for better handling of periods of heightened volatility, resulting in predictions that adapt to the structure of the data over time.

Predicted Conditional Variance

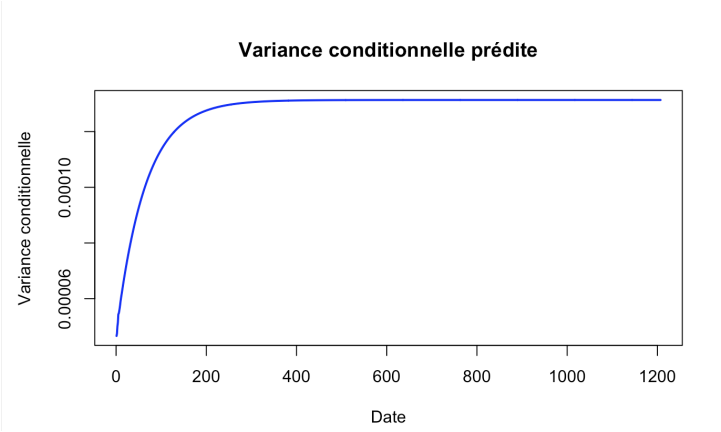


Figure 10: Predicted Conditional Variance from ARMA-GARCH(2,0,5,5)

Figure 10 shows the evolution of the conditional variance predicted by the ARMA-GARCH(2,0,5,5) model over the analyzed period. The conditional variance represents the dynamic nature of volatility in financial time series, which the GARCH component is specifically designed to capture. At the beginning of the series, the variance increases rapidly, reflecting the model's adaptation to the initial uncertainty and lack of information about the volatility dynamics. This is a common behavior in GARCH models, where the initial variance often starts low and stabilizes as the model processes more data. This behavior aligns well with the assumptions of the GARCH model and demonstrates its capability to adapt to changing volatility levels.

Observed vs. Predicted Variance During Training.

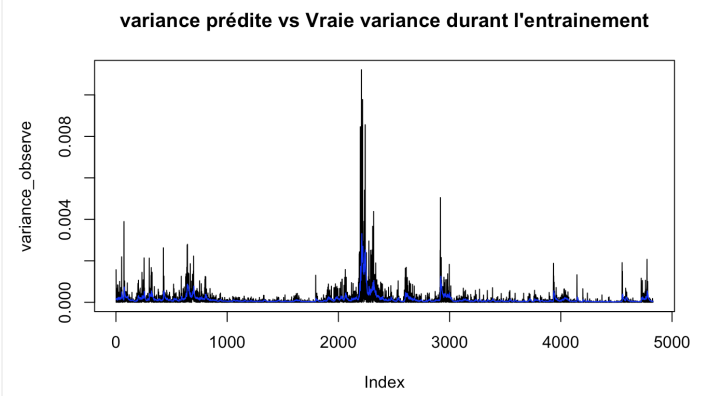


Figure 11: Observed Variance vs. Predicted Variance During Training

Figure 11 compares the observed variance (black line) with the predicted variance (blue line) during the training period. The ARMA-GARCH(2,0,5,5) model closely tracks the spikes in observed variance, particularly during periods of high volatility. This confirms the model's ability to adapt to changing market conditions and accurately predict volatility. The minor discrepancies observed in some areas are likely due to the limitations of the normal distribution assumption in the GARCH model, which could be addressed by using alternative distributions (e.g., Student-t or GED).

2.2.3.3 Discussion .

From the outset, the ARCH-LM test provided strong evidence of heteroskedasticity in the ARMA(2,0) residuals, thereby motivating the introduction of a GARCH component. By coupling ARMA and GARCH, the model not only preserves the ability of ARMA to capture autocorrelation in the mean process but also explicitly accounts for volatility clustering—a hallmark of financial time series.

The resulting ARMA(2,0)-GARCH(5,5) model displays several noteworthy improvements over the plain ARMA(2,0) approach. First, it yields lower AIC and BIC values, reflecting a better overall fit to the data.From a predictive standpoint, although the mean-squared error remains relatively small—largely because log-returns tend to exhibit low numerical magnitudes—the GARCH extension provides more realistic volatility forecasts. Visually, the predicted prices from ARMA-GARCH exhibit smoother dynamics than the raw market data but still respond to volatility fluctuations more effectively than the ARMA-only model (as evidenced by the close tracking of higher-volatility intervals).

Nonetheless, certain limitations persist. The normality assumption in standard GARCH can underestimate tail risk, suggesting that a heavier-tailed distribution (e.g., Student-t) might further improve performance. Likewise, while additional covariates or external economic indicators were tested, they did not enhance predictive power in this setting. Hence, parsimony guided the selection of ARMA(2,0)-GARCH(5,5) as a balanced model, capturing both the dependence in the mean and the clustering of volatility without unnecessary complexity.

2.3 CNN-LSTM MODEL

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) that can learn order dependencies in sequence prediction problems [1][4]. By storing information for an arbitrary duration, they address the limitations of traditional RNNs, enabling solutions to complex problems such as time-series forecasting, natural language processing, and speech recognition. For example, LSTMs excel at tasks like predicting future stock prices based on historical trends or generating text by understanding context over long sequences.

In the context of time series prediction, combining a Convolutional Neural Network (CNN) with an LSTM can be particularly beneficial. CNNs are adept at extracting high-level features from raw input data by detecting local patterns, trends, or anomalies within a time series. CNNs can identify short-term fluctuations. These extracted features are then passed to the LSTM, which specializes in learning temporal dependencies and relationships across these features.

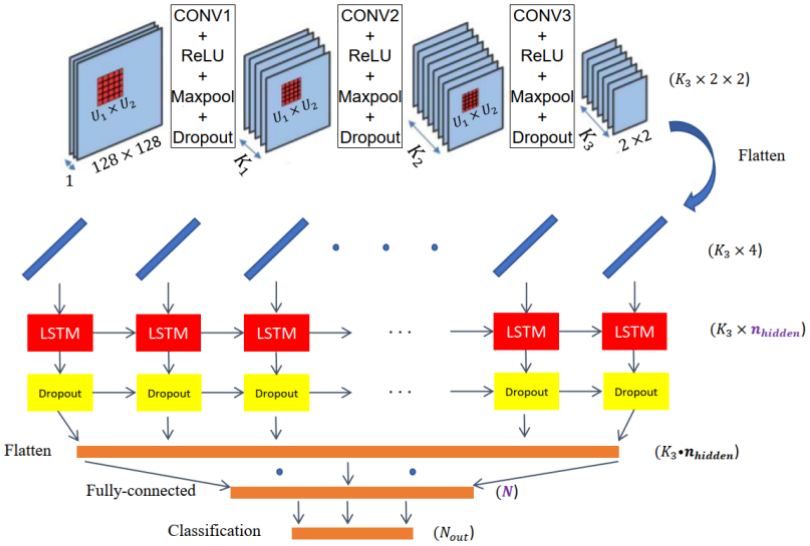


Figure 12: Architecture of CNN-LSTM model

Now we can see how the model performs in practical scenarios by evaluating its effectiveness on the *S&P 500* time-series datasets. The CNN-LSTM architecture, as illustrated in Figure 12, was tested on the datasets to forecast the adjusted closing price. Unlike the previous models, we do not need to use the stationary time series because LSTM can handle trends, seasonality and abrupt changes by learning long-term dependancies in the data and his ability to process repeating pattern across time [4]. The model trained here and the parameters are detailed in the section [4.1]

Visually, the model appears to capture the overall trend and variations in the time series very well. Figure **??** illustrates the observed versus predicted values, showing that the model can effectively forecast the adjusted closing price over time. The predicted values (represented by the red line) closely follow the observed data (represented by the orange line), which indicates that the model successfully learns the underlying patterns in the stock prices.

To quantify this performance, we evaluated the model using different metrics on the real prices, including Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). The results show that the CNN-LSTM model outperforms simpler models in terms of accuracy and generalization, especially in capturing both short-term fluctuations and long-term trends. Table 3 summarizes the comparison of

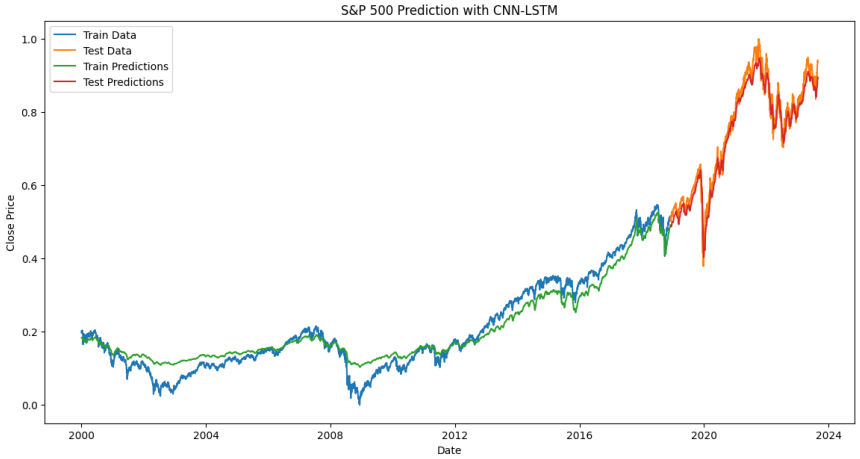


Figure 13: Comparison between Observed and predicted data

these metrics, highlighting the model's ability to minimize prediction errors.

Metric	Value
MSE	0.0021
RMSE	0.0454
MAE	0.0373

Table 3: Error metrics for the model on real prices

The predictions closely follow the fluctuations of the true data with small prediction errors, but machine learning (ML) or deep learning methods are not always suitable for all types of prediction problems. These models excel at capturing complex patterns but are not ideal when interpretability is needed. In some cases, understanding the factors influencing predictions (e.g., seasonality, trends) is crucial.

The key issue with ML methods is their lack of interpretability. While they provide good performance, they do not clearly explain the underlying reasons for fluctuations, unlike classical models such as ARIMA. Classical methods decompose predictions into identifiable components like trend and seasonality, which can be easily explained and analyzed. This transparency is important in fields like finance and where understanding why a prediction was made is necessary for decision-making.

In contrast, ML models tend to be black-boxes. They capture complex relationships but do not explicitly reveal how decisions are made. This lack of interpretability can create challenges in trust, fairness, and regulatory compliance, especially in high-stakes applications. Therefore, while ML methods offer powerful predictive capabilities, classical methods are still preferred when explanations of the predictions are needed for understanding underlying factors or for regulatory purposes.

Another part of this study can be comparing is multiple features CNN-LSTM can outperform classical models which is very probable as shown by Namin et al. [7]. There also is a research area under develop using both autoregression and deep learning networks at once [5]. This method can helps use the benefits of both models and reduce their disavantages. This area is still not develop enough.

3 General Discussion

Time-series forecasting is a critical task in finance, where accurate predictions can guide investment strategies and risk management. The choice of models depends on the trade-offs between interpretability, computational efficiency, and predictive performance. Classical statistical models, such as ARMA and ARIMA (sometimes augmented with GARCH for volatility modeling), have long been the cornerstone of financial econometrics. However, the advent of deep learning has introduced powerful alternatives, such as Long Short-Term Memory (LSTM) networks, that excel in capturing complex temporal dependencies and non-linear patterns. This project compares these two paradigms, focusing on their strengths, limitations, and applicability to financial time series, particularly the S&P 500 index.

Limitations of Classical Autoregressive Models.

Autoregressive models, including ARMA and ARIMA, are widely used for their simplicity, interpretability, and solid theoretical foundation. They are effective in capturing serial dependencies in the mean of a time series, while GARCH models extend their capability to model conditional heteroscedasticity. These features make them particularly valuable in regulated environments where explainability is a priority.

However, the comparison of the results reveals notable limitations. In the ARMA model's predictions, the trend fails to adapt to the observed dynamics of the time series, resulting in a poor fit to the actual stock prices. The predicted prices follow a nearly linear trend, which does not capture the volatility clustering or fluctuations in the real data. The GARCH model improves on this by incorporating a dynamic structure for volatility. While the GARCH model better aligns with periods of heightened or reduced market volatility, it still fails to fully capture the complexity of the series. The predicted prices exhibit smoother trends and moderate fluctuations compared to the real data, highlighting the limitations of relying on rigid parametric assumptions and the model's inability to address long-term dependencies.

Indeed, these models come with inherent limitations. First, their linear structure restricts their ability to capture non-linear dynamics and higher-order patterns often present in financial data. Second, they assume stationarity, which requires pre-processing steps such as differencing or detrending, potentially losing information on long-term trends and seasonality.

Another notable limitation is their forecast horizon. Autoregressive models are most effective for short-term predictions due to their reliance on recent past values. As the forecast range extends, these models struggle to maintain accuracy because they cannot effectively incorporate long-term dependencies or adapt to abrupt structural changes. Given that financial time-series forecasting often involves medium- to long-term horizons, this limitation reduces their utility in certain contexts.

Advantages of CNN-LSTM Models.

Deep learning models, such as CNN-LSTM architectures, address many of these shortcomings. By combining Convolutional Neural Networks (CNNs) with LSTMs, these models leverage the strengths of both architectures. CNNs excel at extracting localized patterns, such as short-term fluctuations or anomalies, from raw data, while LSTMs capture long-term dependencies and temporal relationships.

The CNN-LSTM model used in this project demonstrated strong predictive performance, as evidenced by metrics such as MSE, RMSE, and MAE calculated directly on the stock price. For instance, the RMSE for the CNN-LSTM model was approximately 0.0454, whereas for the ARIMA and GARCH models, it was in the range

of several hundreds. It effectively learned the overall trend and fluctuations in the S&P 500 index, outperforming traditional ARMA-based approaches in both short- and long-term forecasts. Unlike ARIMA, which requires stationarity, the LSTM component can handle non-stationary data, trends, and seasonality directly, thanks to its gating mechanisms which allow us to work directly on the stock prices itself and not the log returns. This flexibility is particularly valuable in financial applications where market conditions often evolve dynamically.

The visual comparison between observed and predicted values highlights the model's ability to closely follow market trends while adapting to periods of heightened volatility. Such adaptability makes CNN-LSTM models suitable for complex, high-frequency financial data where non-linear patterns dominate.

Challenges and Limitations of Deep Learning Models.

Despite their advantages, CNN-LSTM models are not without limitations. One key challenge is their *black-box* nature. Unlike ARIMA or GARCH, which provide clear and interpretable components (e.g., trend, seasonality, and volatility), deep learning models lack transparency. This makes it difficult to explain predictions, which can be a significant drawback in high-stakes domains like finance, where decisions must often be justified to regulators or stakeholders.

Another challenge is the data requirement. Deep learning models typically require large, high-quality datasets for training to avoid overfitting. In scenarios with limited data, classical models might outperform due to their lower data dependency and robustness. Additionally, deep learning models are computationally intensive, requiring significant resources for training and hyperparameter tuning. This can make them less practical for smaller organizations or real-time applications.

Finally, while CNN-LSTM models capture complex patterns effectively, their reliance on deep architectures can introduce overfitting, particularly when the data contains noise or outliers. Regularization techniques and cross-validation are necessary to mitigate this risk, but they add complexity to the modeling process.

Comparative Reflections and Future Directions.

The comparison between classical and deep learning models underscores the trade-offs inherent in each approach. Classical models like ARMA-GARCH remain valuable for their simplicity, interpretability, and established use in finance. They are particularly effective for short-term predictions and when understanding the underlying drivers of a time series is critical. However, their limitations in capturing non-linear patterns and long-term dependencies make them less suited for complex, high-frequency financial data.

On the other hand, CNN-LSTM models offer state-of-the-art predictive accuracy by leveraging their ability to model both short-term and long-term dynamics. They excel in capturing non-linear relationships and adapting to non-stationary environments. However, their lack of interpretability and high computational cost must be weighed against their performance gains.

Future research could explore hybrid approaches that combine the strengths of both paradigms. For example, ARIMA-LSTM or GARCH-LSTM [9] hybrids could model linear and non-linear components separately, leveraging the interpretability of classical models while benefiting from the flexibility of deep learning. Additionally, explainability techniques such as SHAP (SHapley Additive exPlanations) or attention mechanisms could enhance the transparency of deep learning models, making them more suitable for regulatory environments.

CONCLUSION

This project demonstrates that while CNN-LSTM models outperform classical ARMA-GARCH models in terms of predictive accuracy for the S&P 500 dataset, both approaches have their place depending on the specific requirements of the task. Classical models offer clarity and simplicity, making them ideal for short-term forecasts and regulated environments. Deep learning models, on the other hand, excel in capturing complex, long-term dynamics but require careful handling to address their interpretability and computational demands. Combining the two approaches or enhancing the explainability of deep learning models represents a promising direction for future research in financial time-series forecasting .

4 APPENDIX

4.1 Implementation of CNN-LSTM

Here is the implementation of the CNN-LSTM model:

```
class CNNLSTMModel(nn.Module):
    def __init__(self, input_size=1, hidden_size=50):
        super(CNNLSTMModel, self).__init__()
        # Two 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)
        # An LSTM layer
        self.lstm = nn.LSTM(input_size=64, hidden_size=hidden_size, num_layers=1,
            batch_first=True)

        self.fc = nn.Linear(hidden_size, 1)

        self.dropout = nn.Dropout(0.4)

    def forward(self, x):
        # Conv Layer 1
        x = F.relu(self.conv1(x))

        # Conv layer 2
        x = F.relu(self.conv2(x))

        x = self.maxpool(x)

        x = self.dropout(x)

        x = x.permute(0, 2, 1)      # [batch_size, sequence_length, features]

        out, _ = self.lstm(x)

        out = self.fc(out[:, -1, :])  # [batch_size, 1]
        return out
```

Listing 1: CNN-LSTM Model Implementation

References

[1] Y. Bengio, P. Simard, and P. Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166. DOI: 10.1109/72.279181.

[2] R. F. Engle. “Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of United Kingdom Inflation”. In: *Econometrica* 50.4 (1982), pp. 987–1007. DOI: 10.2307/1912773. URL: https://doi.org/10.2307/1912773.

[3] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[4] Dariusz Kobiela et al. “ARIMA vs LSTM on NASDAQ stock exchange data”. In: *Procedia Computer Science* 207 (2022). Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 26th International Conference KES2022, pp. 3836–3845. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2022.09.445. URL: https://www.sciencedirect.com/science/article/pii/S1877050922013382.

[5] Guokun Lai et al. “Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks”. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR ’18. Ann Arbor, MI, USA: Association for Computing Machinery, 2018, pp. 95–104. ISBN: 9781450356572. DOI: 10.1145/3209978.3210006. URL: https://doi.org/10.1145/3209978.3210006.

[6] Greta M. Ljung and George EP Box. “On a measure of lack of fit in time series models”. In: *Biometrika* 65.2 (1978), pp. 297–303. DOI: 10.1093/biomet/65.2.297.

[7] Sima Siami Namini and Akbar Siami Namin. “Forecasting Economics and Financial Time Series: ARIMA vs. LSTM”. In: (Mar. 2018). DOI: 10.48550/arXiv.1803.06386.

[8] Abdelmalek Toumi et al. “Un réseau hybride CNN-LSTM pour la classification de navires à partir d’une base frugale des images SAR”. In: *GRETSI - Groupe de Recherche en Traitement du Signal et des Images*. Grenoble, France, Aug. 2023. URL: https://ensta-bretagne.hal.science/hal-04320593.

[9] Yan Wang et al. “A Garlic-Price-Prediction Approach Based on Combined LSTM and GARCH-Family Model”. In: *Applied Sciences* 12.22 (2022). Submission received: 7 October 2022 / Revised: 31 October 2022 / Accepted: 3 November 2022 / Published: 9 November 2022, p. 11366. DOI: 10.3390/app122211366.