Forecasting Daily Silver Prices Using ARIMA and XGBoost: A Comparative Analysis

Rouaba Mohammed *

Laboratory for the development of Algerian economic

institutions, Ibn Khaldoun University of Tiaret- Algeria

mohammed.rouaba@univ-tiaret.dz

Received: 21/05/2025 Accepted: 24/08/2025 Date de publication: 22 /11/2025

Abstract:

This study aims to forecast the daily closing prices of silver by employing two distinct approaches: the traditional Autoregressive Integrated Moving Average (ARIMA) model and the modern Extreme Gradient Boosting (XGBoost) algorithm. The dataset covers the period from January 3, 2023, to May 21, 2025, obtained from the Yahoo Finance platform. The ARIMA model was specified using the Box–Jenkins methodology, while the XGBoost model incorporated multiple technical indicators, including simple and exponential moving averages, RSI, and MACD. Model performance was evaluated using Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). The results indicate that the ARIMA model achieved higher predictive accuracy than XGBoost for this dataset, though XGBoost demonstrated the ability to capture certain nonlinear patterns.

***Keywords**: Forecasting; Silver Prices; ARIMA; XGBoost; Investment Decision.*

Jel Classification Codes : C58, G17, E37.

* Corresponding author.

1. Introduction:

Forecasting the prices of precious metals, such as silver, is a critical task in financial markets, given their role as both investment assets and economic indicators. Accurate forecasts can support investors, policymakers, and businesses in making informed decisions regarding trading strategies, portfolio diversification, and risk management.

Time series forecasting methods can generally be classified into two categories: statistical models and machine learning algorithms. Statistical models, such as the Autoregressive Integrated Moving Average (ARIMA), have been widely used for decades due to their strong theoretical foundations and interpretability. ARIMA is particularly effective for modeling linear relationships in stationary time series and remains a benchmark method in many forecasting applications.

In recent years, advances in machine learning have introduced powerful alternatives capable of handling complex and nonlinear patterns in data. One such algorithm is Extreme Gradient Boosting (XGBoost), which has gained popularity for its high predictive performance in various domains, including finance. By combining decision trees in a boosting framework, XGBoost can capture nonlinear dependencies that traditional models may overlook.

1.1. Research Problem:

The research problem can be formulated through the following question:

'Which model -ARIMA or XGBoost- provides more accurate forecasts for the daily closing prices of silver over the period from January 3, 2023, to May 21, 2025?'

1.2. Significance of the Study:

This study is significant for several reasons. First, it addresses the practical need for accurate forecasting of silver prices, a commodity that plays a key role as both an investment asset and a hedge against inflation and market volatility. Reliable price forecasts can support investors, traders, and policymakers in making informed decisions, optimizing portfolio strategies, and managing financial risks.

Second, the research contributes to the academic literature by conducting a direct comparison between a well-established statistical method (ARIMA) and a modern machine learning algorithm (XGBoost). While ARIMA remains a standard benchmark in time series analysis, XGBoost has gained recognition for its ability to model complex nonlinear relationships. Evaluating their relative performance in the context of silver price forecasting provides valuable insights for both practitioners and researchers.

Finally, the study enhances methodological understanding by illustrating the strengths and limitations of each model, offering guidance for selecting appropriate forecasting techniques in financial and commodity markets.

1.3. Objectives of the Study:

The main objectives of this study are:

1. To build and evaluate an ARIMA model for forecasting the daily closing prices of silver over the period from January 3, 2023, to May 21, 2025.
2. To build and evaluate an XGBoost model using selected technical indicators as input features for forecasting silver prices over the same period.
3. To compare the forecasting performance of ARIMA and XGBoost based on standard evaluation metrics, namely Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).
4. To identify the strengths and limitations of each approach in the context of precious metal price forecasting.
5. To provide practical recommendations for investors, traders, and researchers on selecting suitable forecasting methods for silver and similar commodities.

2. Theoretical Framework of the Study :

2.1. ARIMA Models:

This methodology was introduced by GEORGE E.P. BOX and GWILYM M. JENKINS in their famous book on time series analysis titled "Time Series Analysis Forecasting & Control" in 1976 (Peter & Silvia, 2012, p. 136). Despite the passage of many years, this methodology is still considered modern and is one of the most commonly used in time series analysis, particularly for short-term forecasting. It relies on autocorrelation and partial autocorrelation functions of the observed variable (Latif et al., 2023, p. 259).

Models Used in ARIMA (Taneja et al., 2016, p. 3):

1- Autoregressive Models (AR):

In this model, the dependent variable y_t depends on its previous values up to period p as follows:

$$y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \delta + \varepsilon_t$$

Where ϕ_i are the autoregressive coefficients, and p is the order of the model.

2- Moving Average Model (MA):

The model takes the following form:

$$y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \delta + \varepsilon_t$$

Where θ_j are the moving average coefficients, and q is the order of the model.

3- Autoregressive Moving Average Model (ARMA (p, q)):

This model is a combination of the two previous models, where the series y_t is a linear function of both the variables $y_{t-1}, y_{t-2}, \ldots, y_{t-p}$ and the errors $\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-q}$ as follows:

$$y_t = \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \delta + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \quad \ldots \theta_q \varepsilon_{t-q}$$

The Box-Jenkins methodology for estimating ARIMA models involves four stages:

1 .Identification: This stage involves determining the appropriate orders p, d, and q, where p represents the order of the autoregressive part, d represents the degree of differencing needed to make the time series stationary, and q represents the order of the moving average part. This stage uses autocorrelation and partial autocorrelation functions and unit root tests (Abonazel & Darwish, 2022, p.7)

2 .Estimation: After identifying the orders p, d, and q, this stage involves estimating the model parameters.

3 .Diagnostic Checking: To ensure that the estimated model is suitable for forecasting and free from any standard problems that could affect the forecasting process, residual diagnostics are performed (Drebee et al., 2025).

4 .Forecasting: After estimating the model and ensuring it is free from any standard problems, future values of the time series are forecasted (Gujarati, 2014, pp. 303-304).

2.2 XGBoost Model:

XGBOOST is one of the most popular and widely used boosting algorithms. It is similar to the gradient boosting algorithm but has some additional features that make it more powerful, and it has a fast-scaling method so that it can be parallelized or distributed across clusters.

This algorithm was developed in 2016 by Chen and Gestrin,(Wang et al., 2021) winning first place in the Kaggle data analysis competition and publishing a research paper on how it works, which proved to be very successful in a short period of time.

The basic framework of XGBoost (Extreme Gradient Boosting) is boosting, and its main basic idea comes from decision tree and integrated learning, which is widely used in classification and regression analysis. The idea of the XGBoost algorithm is: sort all the values of the features and store them as a block structure, then traverse all the splitting points to find the optimal splitting point of a feature. The goal is to find a set of trees that can be used as sample prediction values, and the algorithm makes the sample prediction error, i.e., the objective function, reach the minimum, while also having a certain degree of robustness and generalisation ability. The formula derivation process is as follows :

The first step is to construct the objective function Obj, which consists of the model loss function + regularisation term, where the model is prevented from overfitting by adding a regular term (Chen et al., 2019).

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{i=1}^{t} \Omega(f_i) \quad (1)$$

Where Ω is the regular term, l is the loss function, n is the sample size, y_i is the actual value of the sample, $\hat{y}_i$ is the output value of the ith sample of the model, t is the number of decision trees, and f_i is the mapping used by the t^{th} tree to the leaf points.

In the second step, a Taylor expansion of the loss function l is performed, denoted as:

$$L\left(y_i, y_i^{(t-1)}\right) = g_i f_t(x_i) + 0.5 h_i f_t^2(x_i) \quad (2)$$

Where $f_t(x) = \omega_{q(x)}, \omega \in R^{\mathrm{T}}$, $q: R^d \to \{1,2,\ldots,T\}$ is used to define a tree, T is the total number of leaf nodes (Chen & Guestrin, 2016), and $\Omega(f_k)$ is a regular term to define the complexity of a tree (Deng et al., 2023):

$$\Omega(F_k) = \gamma T + 0.5\lambda \sum_{j=1}^{T} \omega_j^2 \quad (3)$$

In the third step, the kernel function of the objective function for grouping leaf nodes is:

$$\sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_j \right) \omega_j + 0.5 \left(\sum_{i \in I_j} h_i + \lambda \right) \omega_j^2 \right] + \gamma T \quad (4)$$

Define $G_j = \sum_{i \in I_j} g_j$, $H_j = \sum_{i \in I_j} h_i$, then its optimal point is:

$$\omega_j^* = -\frac{G_j}{H_L + \lambda}, \quad Obj = \frac{-1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \lambda T \quad (5)$$

In the fourth step, the dataset is split into two groups and the respective losses of the dataset after splitting are denoted as L_1 and L_2, and the gain after splitting is calculated:

$$\text{Gain} = L_1 + L_2 - L = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} + \frac{(G_R + G_L)^2}{H_L + H_R + \lambda} \right] - \gamma \quad (6)$$

The optimal cut-point partitioning algorithm is then used to traverse all the decision tree nodes to find the objective function.

There are two common XGBoost optimal cut-point partitioning algorithms: a greedy algorithm and an approximation algorithm(Guang, 2021). The greedy algorithm starts from the tree with depth 0. After all the samples at each node are sorted in ascending order according to the eigenvalue of each feature, the optimal splitting point is found and the splitting Gain is recorded, and the feature with the largest Gain is selected as the splitting feature and used as the splitting position. And associate its sample set with the two newly split leaf nodes on that leaf node. Repeat this until the condition is satisfied. Since the greedy algorithm cannot access the memory to find the optimal solution when the amount of data is too large. To address this shortcoming, the approximation algorithm only considers the split point of each feature and uses the split number strategy to greatly reduce the complexity of the computation to find the optimal solution.

2.3 Model Evaluation Criteria:

There are several error metrics that can be used to measure forecasting performance, including Mean Absolute Error (MAE), Mean Square Error (MSE), and Root Mean Square Error (RMSE). These are defined as follows (Willmott & Matsuura, 2005, p. 80):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|e_i|$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}|e_i|^2$$

$$\text{RMSE} = \sqrt{MSE}$$

Where e_i represents the residuals of the model, and n is the number of observations.

3. Practical Framework of the Study:

3.1. Study Data:

The study relied on the daily closing prices of silver, obtained using the 'yfinance' library available in Python, covering the period from 03/01/2023 to 21/05/2025. Table (01) below shows a portion of this data in Python.

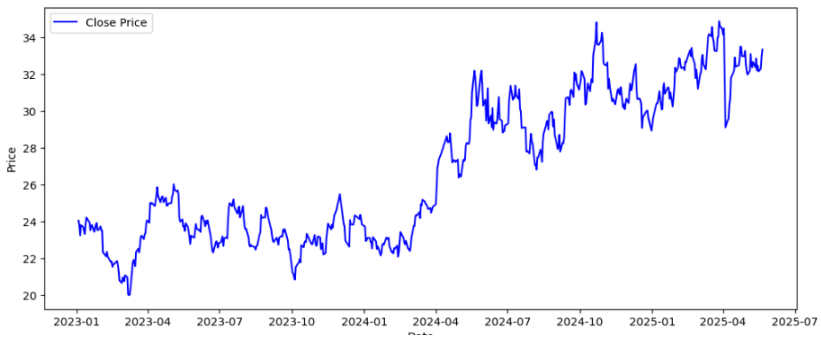
Table N°1: Study Data

Price	Close	High	Low	Open	Volume
Date					
2025-05-15	32.48	32.48	32.48	32.48	2
2025-05-19	32.31	32.57	32.21	32.44	217
2025-05-20	32.98	33.06	32.07	32.12	217
2025-05-21	33.35	33.37	33.12	33.28	11562

Source: Prepared by the researchers based on Python outputs.

Figure (01) below shows the historical evolution of the daily closing prices of silver during the period from 03/01/2023 to 21/05/2025.

Fig N° 1: Historical Evolution of Daily Closing Prices of Silver



Source: Prepared by the researchers based on Python outputs.

The study data was divided into two parts: the first part represents the training dataset, comprising 80% of the total observations for the study period, while the second part represents the testing dataset, comprising 20%. The training data was used to build the study model, while the testing data was used to evaluate the model's performance and its ability to generalize results to unobserved data. Table (02) below provides detailed information on the data split.

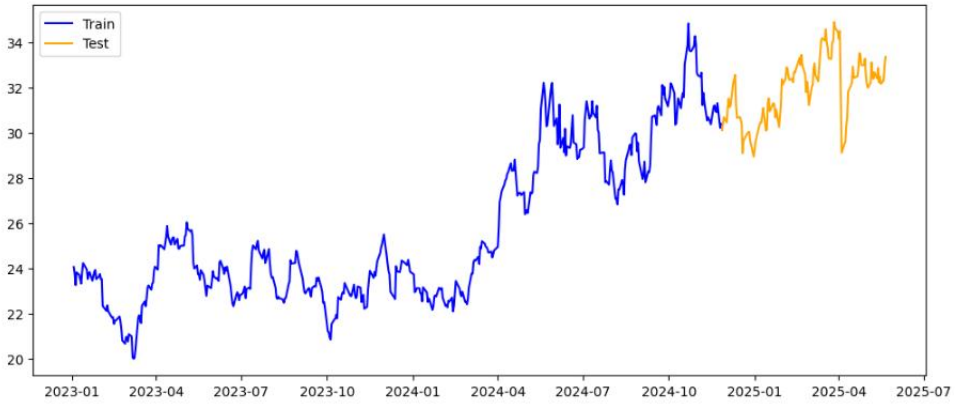
Table N°2: Data Split

	Training data	Test data
Split ratio	80%	20%
Period	03/01/2023 – 26/11/2024	27/11/2024 – 21/05/2025
Observations	479	120

Source: Prepared by the researchers based on Python outputs.

Figure (02) below shows how the time series of daily closing prices of silver was split into training and testing data.

Fig N° 2: Training and Testing Data



Source: Prepared by the researchers based on Python outputs.

3.2. Estimating the ARIMA Model:

To estimate the ARIMA model, the following steps were followed:

First Stage: Identification Stage

This is the most challenging stage in building time series models, as a large number of alternative models can be considered. This stage can be divided into two parts: using unit root tests to determine the degree of integration of the series, and then using autocorrelation and partial autocorrelation functions to determine the orders of the ARIMA model.

Stationarity Test (Unit Root Test):

To test the degree of stationarity of the time series, the Augmented Dickey-Fuller (ADF) test was used. The results of this test are shown in Tables (03) and (04) below:

Table N°3: Results of the ADF Test for the Time Series at Level

Metric	Value
ADF Statistic	-0.972603
p-value	0.763125
Lags Used	8.000000
Number of Observations Used	590.000000
Critical Values (1%)	-3.441482
Critical Values (5%)	-2.866451
Critical Values (10%)	-2.569386

Source: Prepared by the researchers based on Python outputs.

Table N°4: Results of the ADF Test for the Time Series at First Difference

Metric	Value
ADF Statistic	-9.807949
p-value	0.000000
Lags Used	7
Number of Observations Used	590
Critical Values (1%)	-3.441482
Critical Values (5%)	-2.866451
Critical Values (10%)	-2.569386

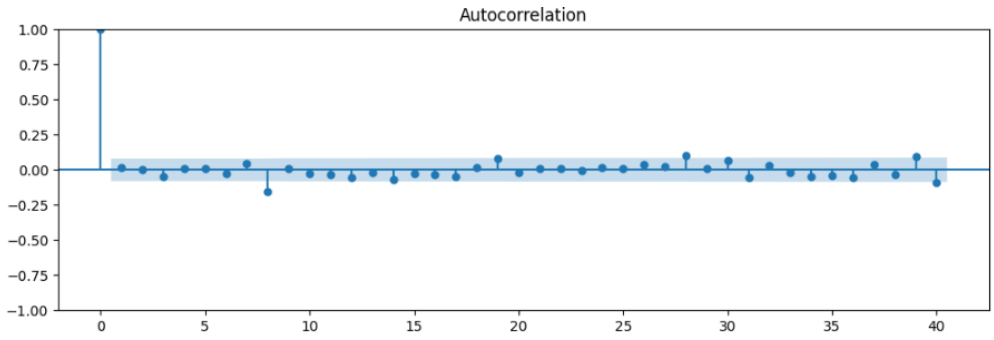
Source: Prepared by the researchers based on Python outputs.

From the results of the ADF test shown in Tables (03) and (04), it is observed that the time series is not stationary at the level, but it becomes stationary after taking the first difference, indicating that the series is integrated of order 1.

Autocorrelation Function and Partial Autocorrelation Function:

Figure (03) below shows the autocorrelation function, where it is observed that the autocorrelation values differ significantly from zero at some lags.

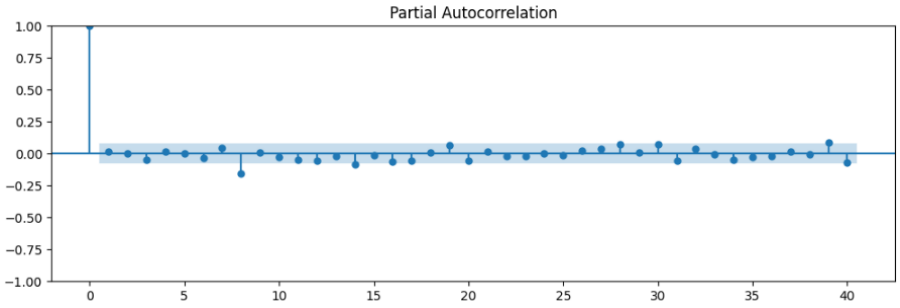
Fig N° 3: Autocorrelation Function



Source: Prepared by the researchers based on Python outputs.

Figure (04) below shows the partial autocorrelation function, where it is observed that the partial autocorrelation values also differ significantly from zero at some lags.

Fig N° 4: Partial Autocorrelation Function

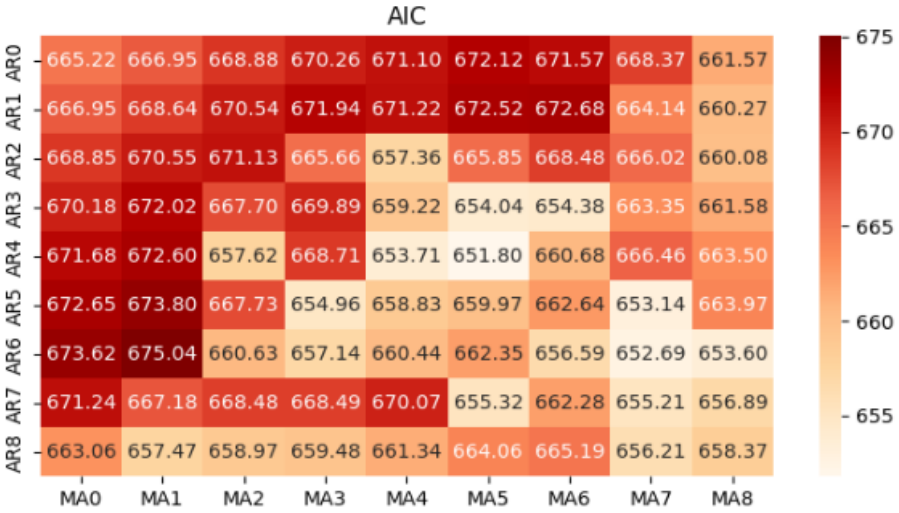


Source: Prepared by the researchers based on Python outputs.

Determining the Order of the ARIMA Model:

Python was used to determine the autoregressive and moving average orders for the best predictive model based on the AIC criterion. The best predictive model is the one that gives the lowest AIC value. Table (05) below shows the AIC values for different ARIMA model orders.

Table N°5 : AIC Values for Different ARIMA Model Orders



Source: Prepared by the researchers based on Python outputs.

From Table (05), it is clear that the best predictive model is the ARIMA (4,1,5) model, which gives the lowest AIC value of 651.80.

Second Stage: Estimation Stage

After identifying the degree of integration of the time series and determining the optimal ARIMA (4,1,5) model, the next stage is to estimate the parameters of this model. Table (06) below shows the estimation results.

Table N°6: Estimation Results of the ARIMA Model

SARIMAX Results

Dep. Variable:	SI=F	No. Observations:	479
Model:	SARIMAX(4, 1, 5)	Log Likelihood	-315.901
Date:	Wed, 21 May 2025	AIC	651.802
Time:	06:10:32	BIC	693.498
Sample:	0	HQIC	668.195
	- 479		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.5051	0.044	-11.475	0.000	-0.591	-0.419
ar.L2	0.7936	0.039	20.560	0.000	0.718	0.869
ar.L3	-0.4148	0.038	-10.819	0.000	-0.490	-0.340
ar.L4	-0.8570	0.039	-21.701	0.000	-0.934	-0.780
ma.L1	0.4432	0.185	2.398	0.017	0.081	0.806
ma.L2	-0.8646	0.113	-7.661	0.000	-1.086	-0.643
ma.L3	0.4886	0.283	1.726	0.084	-0.066	1.044
ma.L4	0.9396	0.229	4.103	0.000	0.491	1.388
ma.L5	0.0355	0.048	0.736	0.462	-0.059	0.130
sigma2	0.2151	0.050	4.298	0.000	0.117	0.313

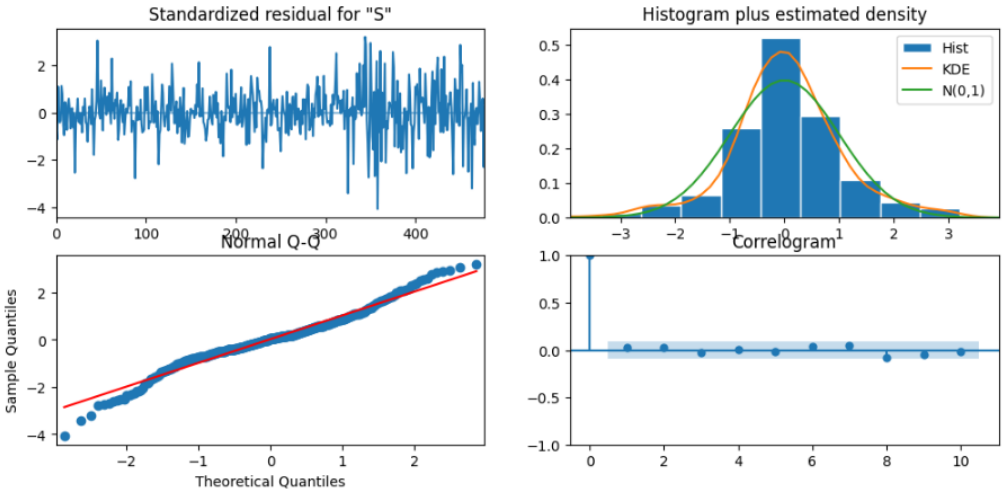
Ljung-Box (L1) (Q):	0.36	Jarque-Bera (JB):	53.36
Prob(Q):	0.55	Prob(JB):	0.00
Heteroskedasticity (H):	2.84	Skew:	-0.10
Prob(H) (two-sided):	0.00	Kurtosis:	4.62

Source: Prepared by the researchers based on Python outputs.

Third Stage: Diagnostic Checking Stage

Figure (05) below contains four diagnostic plots for the residuals.

Fig N° 5: Diagnostic Plots for Residuals



Source: Prepared by the researchers based on Python outputs.

From Figure (05), the top-left plot shows the residual plot, while the plot to its right shows the distribution of residuals, which is observed to be close to a normal distribution. This is also confirmed by the Q-Q plot in the bottom-left corner.

Figure (05) also shows the autocorrelation coefficients of the residual series, where it is observed that these coefficients do not differ from zero and are within the confidence interval (5%), indicating that the error terms of the residuals are independent and not autocorrelated. This is further confirmed by Table (07), which shows the results of the Durbin-Watson test, where the test statistic value is 2, indicating no autocorrelation in the residual series.

Table N°7: Results of the Durbin-Watson Test

Durbin-Watson test on residuals	2.028177
Test for all AR roots outside unit circle (>1)	True
Test for all MA roots outside unit circle (>1)	True

Source: Prepared by the researchers based on Python outputs.

Fourth Stage: Forecasting Stage

Figure (06) below shows the degree of fit between the actual values and the predicted values during the testing period. It is observed that there is a high degree of convergence between the original series curve and the estimated series curve, which is confirmed by Table (08) below, which shows the performance evaluation metrics for this methodology.

Fig N° 6: Fit Between Predicted and Actual Values for the ARIMA Model During the Testing Period

Source: Prepared by the researchers based on Python outputs.

Table N°8: Performance Evaluation Metrics for the ARIMA Model

Criterion	MAE	MSE	RMSE
Value	0.4502	0.3802	0.6166

Source: Prepared by the researchers based on Python outputs

3.3. Estimating the XGBoost Model:

- **Target feature:** We took as a target the daily closing price of Silver.
- **Input features:** The following table shows the features used to predict the target feature.

Table N°9: Input Features

SMA_5, SMA_10, SMA_15, SMA_30	**Simple Moving Average (5, 10, 15, 30 days)**
EMA_12, EMA_26	12 and 26-day exponential moving average
RSI	Relative Strength Index
MACD	Convergence/Divergence of Moving Averages

Source: Prepared by the researchers based on Python outputs

Algorithm parameterization:

When using the XGBoost algorithm, we need to adjust the parameters of this algorithm according to the characteristics of the price of silver, these parameters are shown below (Zhang, 2022, p. 598):

n_estimators	[100, 200, 300, 400]
Learning_rate	[0.001, 0.005, 0.01, 0.05]
Max_depth	[3, 5, 8, 10, 12, 15]
Gamma	[0.001, 0.005, 0.01, 0.2]
Random_state	[42, 43, 44, 45]

Where: n_estimators: Number of trees in the model.

Learning rate: Controls the speed of learning.

Max_depth: Maximum tree depth to control complexity.

Gamma: Controls minimum split improvement.

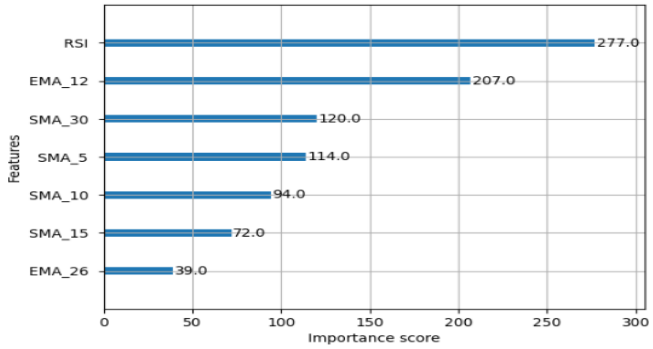
Random_state: Ensures reproducible results.

The optimal values of these parameters obtained after applying the XGBoost algorithm are:

n_estimators	200
Learning_rate	0.05
Max_depth	3
Gamma	0.2
Random_state	42

Figure 7 shows us the importance of input features in predicting the daily closing price of Silver.

Fig N° 7: Importance of input features in predicting the target feature

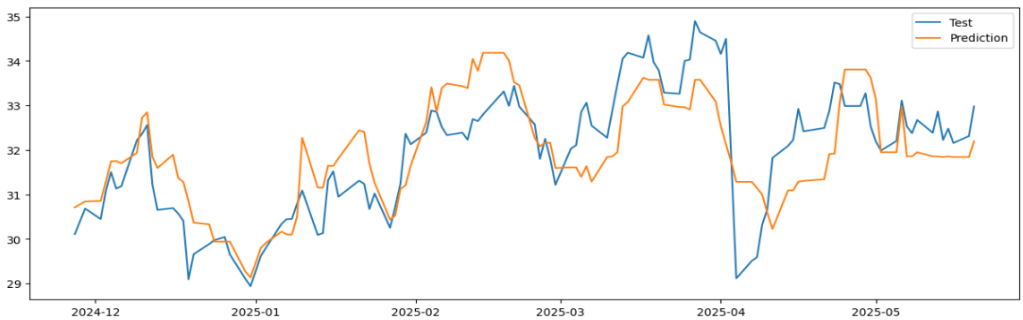


Source: Prepared by the researchers based on Python outputs.

Prediction and verification of model accuracy: After completing the model estimation process, the model must be tested on data outside the training sample to ensure its generalization efficiency so that the test sample is estimated on which the model has never been trained.

Figure 8 below shows us the actual and predicted values of the test data using the XGBoost algorithm.

Fig N° 8: Fit Between Predicted and Actual Values for the XGBoost Model During the Testing Period



Source: Prepared by the researchers based on Python outputs

Table N°10: Performance Evaluation Metrics for the XGBoost Model

Criterion	MAE	MSE	RMSE
Value	0.7284	0.7869	0.8871

Source: Prepared by the researchers based on Python outputs

4. Results and discussion

The forecasting results for both the ARIMA and XGBoost models were evaluated using the test dataset covering the period from November 27, 2024, to May 21, 2025. Three standard error metrics (MAE, MSE, and RMSE) were used to measure the predictive performance of each model.

For the ARIMA (4,1,5) model, the evaluation yielded an MAE of 0.4502, an MSE of 0.3802, and an RMSE of 0.6166. These relatively low error values indicate a high level of predictive accuracy, with the model closely tracking the actual daily closing prices of silver during the testing period. The residual diagnostics confirmed that the error terms were normally distributed and free from autocorrelation, validating the model's suitability for forecasting.

In contrast, the XGBoost model, which incorporated multiple technical indicators such as SMA, EMA, RSI, and MACD, produced an MAE of 0.7284, an MSE of 0.7869, and an RMSE of 0.8871. While the model demonstrated the ability to capture some nonlinear patterns in the data, its overall accuracy was lower than that of the ARIMA model for this dataset. This performance gap may be attributed to the relatively short time series, which could limit the advantage of complex nonlinear models that require larger datasets to generalize effectively.

The comparative analysis shows that, for the given period and dataset, the ARIMA model outperformed XGBoost in terms of predictive accuracy. However, this does not diminish the potential of XGBoost in contexts where nonlinearities are more pronounced or where larger datasets are available. Future research could explore hybrid models that combine the strengths of ARIMA in capturing linear dependencies with the nonlinear modeling capabilities of XGBoost, potentially leading to improved forecasting performance.

5. Conclusion:

This study examined the forecasting performance of two distinct approaches (ARIMA and XGBoost) in predicting the daily closing prices of silver over the period from January 3, 2023, to May 21, 2025. The ARIMA (4,1,5) model demonstrated superior predictive accuracy, achieving lower error metrics (MAE, MSE, RMSE) compared to the XGBoost model. Residual analysis confirmed that the ARIMA model satisfied the assumptions required for reliable forecasting, while XGBoost, despite its capability to capture nonlinear patterns, yielded less accurate predictions for this dataset.

The findings suggest that, for relatively short time series with predominantly linear structures, traditional statistical models like ARIMA may outperform more complex machine learning algorithms. However, XGBoost remains a valuable tool in scenarios involving larger datasets or more pronounced nonlinearities.

For practitioners and researchers in commodity and financial markets, these results highlight the importance of matching the forecasting method to the nature of the data. Future studies could explore hybrid approaches that integrate ARIMA's strength in modeling linear components with XGBoost's ability to capture nonlinear patterns, potentially leading to improved forecasting performance.

6. List of references:

1. bonazel, M. R., & Darwish, N. M. (2022). Forecasting confirmed and recovered COVID-19 cases and deaths in Egypt after the genetic mutation of the virus: ARIMA Box-Jenkins approach. Commun. Math. Biol. Neurosci., 2022, Article ID 17.

2. Drebee, H. A., Abd Ali, M. K., Obayes, F. A., & Jarallah, R. S. (2025). Forecasting Agricultural Unemployment in Iraq: A Box-Jenkins Approach for 2024-2028. IOP Conference Series: Earth and Environmental Science,

3. Gujarati, D. (2014). Econometrics by example. Bloomsbury Publishing.

4. Latif, N., Selvam, J. D., Kapse, M., Sharma, V., & Mahajan, V. (2023). Comparative performance of LSTM and ARIMA for the short-term prediction of bitcoin prices. Australasian Accounting, Business and Finance Journal, 17(1), 256-276.

5. Peter, D., & Silvia, P. (2012). ARIMA vs. ARIMAX--which approach is better to analyze and forecast macroeconomic time series. Proceedings of 30th international conference mathematical methods in economics.

6. Taneja, K., Ahmad, S., Ahmad, K., & Attri, S. (2016). Time series analysis of aerosol optical depth over New Delhi using Box--Jenkins ARIMA modeling approach. Atmospheric Pollution Research, 7(4), 585-596.

7. Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. Climate research, 30(1), 79-82.

8. Chen, M., Liu, Q., Chen, S., Liu, Y., Zhang, C.-H., & Liu, R. (2019). XGBoost-based algorithm interpretation and application on post-fault transient stability status prediction of power system. IEEE Access, 7, 13149-13158.

9. Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining,

10. Deng, S., Huang, X., Zhu, Y., Su, Z., Fu, Z., & Shimada, T. (2023). Stock index direction forecasting using an explainable eXtreme Gradient Boosting and investor sentiments. The North American Journal of Economics and Finance, 64, 101848.

11. Guang, Y. (2021). Generalized xgboost method. arXiv preprint arXiv:2109.07473.

12. Wang, P., Zhang, G., Yu, Z.-G., & Huang, G. (2021). A deep learnING And XGBoost-based method for predicting protein-protein interaction sites. Frontiers in Genetics, 12, 752732.

13. Zhang, Y. (2022). Stock price prediction method based on XGboost algorithm. 2022 International Conference on Bigdata Blockchain and Economy Management (ICBBEM 2022),