 ISSN:2302-8149
e-ISSN:2540-9719

Bitcoin Price Forecasting using Seasonal Log-Differenced XGBoost with 2014–2025 Data

[1]Muhammad Akbar*
[1]Master of Information Technology, Computer Science Faculty, University of Indonesia, Jakarta, Indonesia
*e-mail: muhammad.akbar57@ui.ac.id

(***received*:** 17 July 2025, ***revised*:** 14 August 2025, ***accepted*:** 15 August 2025)

Abstrak

Bitcoin, as a decentralized digital currency, experiences significant price fluctuations, making accurate price forecasting a complex yet valuable challenge. Price forecasting is essential in economic decision-making, serving as the foundation for portfolio construction, risk analysis, and investment strategy development. Bitcoin's high volatility makes it an attractive asset for investors but also poses significant risks, necessitating sophisticated forecasting tools and models to mitigate uncertainty. The XGBoost model in regression is widely known and effectively applied to handle time series data. This model can capture complex nonlinear relationships in Bitcoin price data, providing more accurate forecasts than traditional statistical models. The research methodology includes data collection, data preprocessing, stationarity checking, differencing, feature engineering, data division into training and testing sets, XGBoost model training, prediction and evaluation, and result visualization. The research results show that the XGBoost model achieves a Mean Absolute Error of 8.26% and an RMSE of 9.87%, indicating excellent forecasting accuracy. The implications of this research could potentially assist investors and traders in improving their strategies and risk management.

Keywords: forecasting, bitcoin, XGBoost model, MAE, digital currency

1 Introduction

Currently, the global financial landscape is undergoing a revolutionary transformation due to the emergence of cryptocurrencies, with Bitcoin as its pioneering entity. As a leading digital asset, Bitcoin has garnered significant attention from investors, academics, and regulators worldwide [1]. Bitcoin's high price volatility presents both challenges and attractive investment opportunities, driving the imperative to develop precise and reliable prediction models [2].

The cryptocurrency market has unique characteristics that distinguish it from conventional financial markets. The absence of clear seasonal patterns and a number of unrealistic constraints complicate accurate forecasting using traditional statistical methods [3]. Therefore, Bitcoin price prediction requires a more in-depth approach that can identify complex patterns, temporal dependencies, and external elements that can influence price changes [4].

Bitcoin's inherent volatility distinguishes it from conventional investment assets, necessitating the adoption of more sophisticated forecasting methodologies [5]. Current research is frequently constrained to the utilization of sentiment indices and price data, thereby establishing a notable research deficiency in the integration of more varied and advanced data sources and techniques to augment prediction precision [4].

Previous research has documented the successful utilization of time series models, such as ARIMA, and machine learning models, like Support Vector Regression, for stock price prediction [6], [7]. However, these conventional models might exhibit limitations in capturing the intricate patterns and nonlinear dependencies inherent in Bitcoin's price data. Additionally, artificial neural networks have demonstrated considerable promise in forecasting applications. Several studies indicate that recurrent neural networks provide superior accuracy and robustness compared to standard neural networks [8]. Furthermore, Long Short-Term Memory, a variation of recurrent neural networks, has been employed for stock price prediction, achieving commendable RMSE results.

This research endeavors to tackle these complexities by investigating the efficacy of XGBoost, a potent machine learning algorithm, for forecasting Bitcoin prices. The approach integrates time series-derived features with Fourier-based decomposition to effectively capture seasonal variations. Machine

 ISSN:2302-8149
e-ISSN:2540-9719

learning methodologies offer several benefits over traditional statistical techniques, including their capacity to handle nonlinear and high-dimensional datasets, discern intricate relationships, and adapt to evolving market dynamics. Specifically, XGBoost has emerged as a valuable tool for forecasting due to its inherent gradient boosting, regularization capabilities, and proficiency in managing missing data.

The structure of this research consists of an introduction that outlines the background and motivation of the study, followed by a literature review that discusses the concepts of Bitcoin, decentralization, and the XGBoost model. The methodology section describes the implementation stages of the XGBoost algorithm, time series features, and Fourier-based decomposition. The results and discussion section presents interpretations of empirical results, experiments, and in-depth analysis. Finally, the conclusion summarizes the main findings, research implications, and potential directions for future research.

2 Literature Review

The literature review is structured to cover the following key areas: Bitcoin and Decentralization, Machine Learning, XGBoost for Time Series Forecasting, Feature Engineering for Time Series Data, Fourier Transform for Seasonality Decomposition, Model Evaluation and Performance Metrics, Differencing, Resampling Methods for Imbalanced Data, and Dimensionality Reduction. Each of these areas will be discussed in detail below:

2.1 Bitcoin and Decentralization

Bitcoin, introduced in 2008 by the anonymous entity Satoshi Nakamoto, represents an innovative decentralized peer-to-peer digital currency system that facilitates online transactions without the mediation of a central authority. Bitcoin's decentralized architecture is implemented through blockchain technology, a distributed public ledger that records all transactions on various computer nodes [8]. Blockchain ensures transparency, security, and immutability, making it highly resistant to censorship and manipulation. Bitcoin's decentralization allows it to operate independently of conventional financial institutions and governmental authorities, giving users a higher degree of control and autonomy over their assets.

Blockchain technology, which forms the foundation for Bitcoin, plays a central role in the creation of cryptocurrencies [9]. This system offers a decentralized and transparent approach to recording and verifying transactions [10].

Blockchain technology has key features including transparency, security, and decentralization. Blockchain records all transactions publicly and permanently, enhancing financial security and transparency. The distribution of blockchain technology across various computers eliminates the need for a centralized authority, further reducing the risk of fraud and manipulation.

Bitcoin's decentralization has attracted significant interest and driven widespread adoption for several reasons. First, Bitcoin offers an alternative to conventional financial systems regulated by central banks and financial institutions. Second, Bitcoin provides a sensor-resistant and permissionless means of conducting transactions, allowing individuals to participate in the global economy without restrictions. Bitcoin has stimulated discussions among economists regarding its ability to disrupt existing payment and monetary systems, as well as providing extensive data on the behavior of agents and the Bitcoin system [11].

2.2 Machine Learning

Machine learning, an area within artificial intelligence, is dedicated to creating systems capable of learning from data without requiring explicit programming. Its algorithms are widely employed for tasks such as classification, regression, clustering, and dimensionality reduction. The ability of machine learning algorithms to learn from experience and emulate human cognitive processes has significantly increased interest in both machine learning and artificial intelligence [12]. Generally, these algorithms are divided into supervised, unsupervised, and reinforcement learning categories. Supervised learning algorithms are trained using labeled datasets, which consist of corresponding inputs and outputs. The algorithm then learns to map these inputs to their respective outputs, enabling it to make predictions on new, unseen data [13].

Unsupervised learning algorithms are trained on unlabeled datasets, requiring them to discern inherent patterns and structures. In contrast, reinforcement learning algorithms learn through interaction with an environment, receiving feedback via rewards or penalties. The rapid evolution of

 ISSN:2302-8149
e-ISSN:2540-9719

machine learning in recent years has been propelled by the increasing availability of data, advancements in computational power, and the development of novel algorithms [14].

2.3 XGBoost for Time Series Forecasting

XGBoost, an acronym for Extreme Gradient Boosting, is a highly effective and powerful machine learning algorithm employed for classification and regression challenges. It operates on the principle of gradient boosting, an ensemble methodology that synergistically combines weak models, typically decision trees, in a sequential manner to systematically reduce errors generated by preceding models. XGBoost is favored for its speed, accuracy, and scalability, establishing it as a preferred choice for data scientists and machine learning specialists.

To prevent overfitting and improve generalization performance, the XGBoost algorithm implements a series of techniques. These techniques involve L1 and L2 regularization, which penalize the objective function to limit model complexity, as well as tree pruning, which limits the depth and complexity of individual trees. Furthermore, XGBoost supports the handling of missing values, eliminating the need for data imputation and allowing the algorithm to work directly with incomplete datasets.

One of the main advantages of XGBoost is its ability to capture nonlinear interactions and complex dependencies in data [15]. Through additive model construction, XGBoost can learn complex relationships between features and target variables, which is often difficult for traditional linear models to achieve.

In various domains such as finance, economics, and meteorology, XGBoost has been extensively applied to time series forecasting tasks. For example, the algorithm has been used to predict stock prices, energy demand, and retail sales volume. XGBoost's performance in these tasks is often superior, frequently surpassing traditional statistical methods and other machine learning algorithms [16].

XGBoost has emerged as a highly effective algorithm for forecasting, often outperforming ARIMA models in terms of accuracy and its ability to capture complex relationships. By employing machine learning algorithms within a gradient boosting framework, XGBoost facilitates parallel tree boosting, thereby providing fast and accurate solutions for various data science problems [17]. To prevent overfitting, XGBoost utilizes regularization techniques, such as LASSO and Ridge, to penalize more complex models [18].

Optimizing essential parameters such as gamma, subsample, nrounds, max_depth, eta, colsample_bytree, and min_child_weight using the "xgboost" package, as well as 10-fold cross-validation, has been shown to significantly improve the performance of predictive models [19]. In the context of machine learning models, XGBoost has become a major competitor, consistently achieving optimal performance in various applications [20]. XGBoost not only serves as a stand-alone prediction tool but is also frequently integrated into actual production workflows to predict ad click-through rates. Moreover, XGBoost is a top choice in ensemble methods and has been used in various competitions, including the Netflix challenge.

2.4 Feature Engineering for Time Series Data

Time series data contains valuable information that can be extracted through feature engineering. Lag features, which incorporate historical values from the time series, are crucial for capturing temporal dependencies and patterns in the data. As stated in [21], the XGBoost algorithm demonstrates superior accuracy and stability compared to other algorithms, making lag features a valuable tool for enhancing the predictive performance of models.

In addition to lag features, rolling statistics such as moving averages and standard deviations can provide additional insights into the trends and volatility of the time series [22]. Feature engineering involves leveraging domain expertise to derive meaningful features from raw data through data transformation [23]. This process is critical, as the performance of machine learning algorithms is significantly influenced by how data is presented to them, impacting the effectiveness of the trained model [23].

Furthermore, the incorporation of Fourier-based features can aid in capturing seasonal patterns in time series data. Rolling statistics, including moving averages and standard deviations, can provide valuable insights into trends and volatility in time series, thus providing a basis for identifying and quantifying trends and changes in the data. The combination of these features with powerful machine learning techniques such as XGBoost results in comprehensive and accurate forecasting models [24].

 ISSN:2302-8149
e-ISSN:2540-9719

2.5 Fourier Transform for Seasonality Decomposition

Fourier transformation is a robust mathematical tool for analyzing and representing signals in the frequency domain [25]. This transformation decomposes signals into distinct frequency components, facilitating the identification and quantification of periodic patterns.

In the context of time series forecasting, Fourier transformation can be employed to deconstruct time series into distinct seasonal components. Through calculating high-frequency components and dominant periods from time series samples derived from various benchmark datasets, Fourier analysis indicates that real-world datasets often rely on more than one seasonal pattern [26].

Fourier coefficients, resulting from the Fourier transformation, serve to measure both the amplitude and phase of each frequency component. These coefficients can then be utilized as features in machine learning models to capture potential seasonal effects [27]. Integrating these seasonal components into forecasting models has the potential to significantly enhance prediction accuracy.

Furthermore, spectral analysis, enabled by Fourier transformation, facilitates the identification of dominant frequencies along with their relative strengths. This enables a thorough understanding of seasonal behavior within the data.

In practical implementations, the Fourier transformation is applied to time series data, and the resulting Fourier coefficients are utilized as additional features in the XGBoost model. The combination of XGBoost's ability to capture nonlinear relationships and the effectiveness of Fourier transformation in decomposing seasonality results in a robust and flexible forecasting model. By applying a moving average filter, noise can be reduced through the use of average values of the data obtained [28].

2.6 Model Evaluation and Performance Metrics

Model evaluation and the selection of appropriate performance metrics are critical in the time series forecasting process. Several commonly used metrics include Mean Absolute Error, Mean Squared Error, and Root Mean Squared Error. These metrics provide quantitative measures of the predictive accuracy of a model by quantifying the magnitude of errors between predicted and actual values.

Mean Absolute Error is the average of the absolute differences between the forecasted values and the actual values, offering a clear and interpretable measure of forecast error. MAE indicates the average magnitude of errors and is less sensitive to outliers compared to squared error metrics [29], [30]. The formula for Mean Absolute Error is presented as shown in equation (1)

$$MAE = \frac{1}{n}\sum_{i=0}^{n}\left|X_i - \hat{X}_i\right| \qquad (1)$$

Mean Absolute Percentage Error expresses accuracy as a percentage, facilitating ease of interpretation and comparison across various time series. Mean Absolute Percentage Error is particularly useful for assessing forecasting performance in financial applications where relative errors are more informative than absolute errors [29]. The formula for Mean Absolute Percentage Error is presented as given in equation (2)

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\frac{|X_i - \hat{X}_i|}{X_i} \qquad (2)$$

Mean Squared Error is a loss function calculated as the average of the squared differences between target and predicted values. Root Mean Squared Error, which is the square root of the MSE value, is also widely utilized. Mean Squared Error measures the average of the squares of the errors, providing a quadratic scoring rule that penalizes larger errors more heavily [29][30]. The formula for Mean Absolute Percentage Error is presented as equation (3)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{X}_i)^2} \quad (3)$$

The coefficient of determination, Mean Squared Error, Root Mean Squared Error, and Mean Absolute Error are some of the metrics used to evaluate the performance of each model. In addition to

 ISSN:2302-8149
e-ISSN:2540-9719

error metrics, graphical visualizations such as actual versus predicted plots and residual plots can provide valuable insights into model performance and identify potential areas for improvement [29].

2.7 Differencing

Differencing is a technique used to make time series stationary by removing changes in the level of the series, thus stabilizing the mean of the time series and reducing or eliminating trends and seasonality. If the data is not stationary with respect to the mean, then differencing is performed [30].

Differencing calculates the difference between successive observations, which is often effective in stabilizing the mean of the time series and reducing or eliminating trends and seasonality. The use of non-stationary data requires a differencing or transformation process to ensure that the data becomes stationary, thus meeting the assumptions required for ARIMA modeling [31]. However, differencing can cause the loss of some important information, especially if applied excessively.

Differencing consists of 3 types: simple differencing, seasonal differencing, and double differencing. Simple differencing is a differencing method that involves subtracting the value of the previous period from a series by the current value of the series.[34] This process is particularly effective for removing linear trends[35]. The formula for simple differencing can be expressed as given in equation (4)

$$Y_t' = Y_t - Y_{t-1} \quad (4)$$

Seasonal differencing is employed to mitigate seasonal variations and ascertain the time period. This technique involves subtracting an observation from a previous observation from the same season of the previous year. This method is crucial when dealing with time series that exhibit recurring patterns at fixed intervals, such as monthly or quarterly seasonality[35]. The formula for seasonal differencing is expressed as given shown in equation (5)

$$Y_t' = Y_t - Y_{t-s} \quad (5)$$

Double differencing entails differencing the data twice, rendering it highly effective when dealing with time series exhibiting both trends and seasonality. Double differencing can remove both a linear trend and a seasonal trend that might otherwise obscure underlying patterns [36]. However, it is crucial to avoid over-differencing, as this can lead to the loss of valuable information and result in a fitted model that deviates significantly from the true underlying data generating process [34] [35]. The formula for double differencing can be expressed as show in equation (6)

$$Y_t'' = Y_t' - Y_{t-1}' = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) \quad (6)$$

2.8 Resampling Methods for Imbalanced Data

Resampling data can be grouped into three types: oversampling, undersampling, and a combination of both. In scenarios where data is imbalanced, resampling techniques can be used to address this issue. Oversampling involves adding copies of minority class examples, while undersampling involves removing majority class examples [33]. Resampling methods can help improve model performance by balancing class distribution and preventing models from becoming biased towards the majority class.

2.9 Dimensionality Reduction

Dimensionality reduction can be implemented as a preprocessing step to enhance classification models [34]. It involves reducing the number of features while retaining essential information [34]. Principal Component Analysis is a method employed for dimensionality reduction during preprocessing [35].

Principal Component Analysis is a dimensionality reduction technique applicable for reducing the number of features in microarray data [36]. Data reduction is a technique used to decrease the amount or dimensions of a dataset, enhancing the efficiency of model learning, improving model performance, preventing overfitting, and rectifying skewed data distributions [37].

3 Proposed Method

This section outlines the methodology employed for forecasting Bitcoin prices, utilizing XGBoost in conjunction with time series features and Fourier-based seasonality. It encompasses an

 ISSN:2302-8149
e-ISSN:2540-9719

understanding of the data, data preprocessing steps, the model architecture, and evaluation metrics. The research methodology is visually depicted in Figure 1.

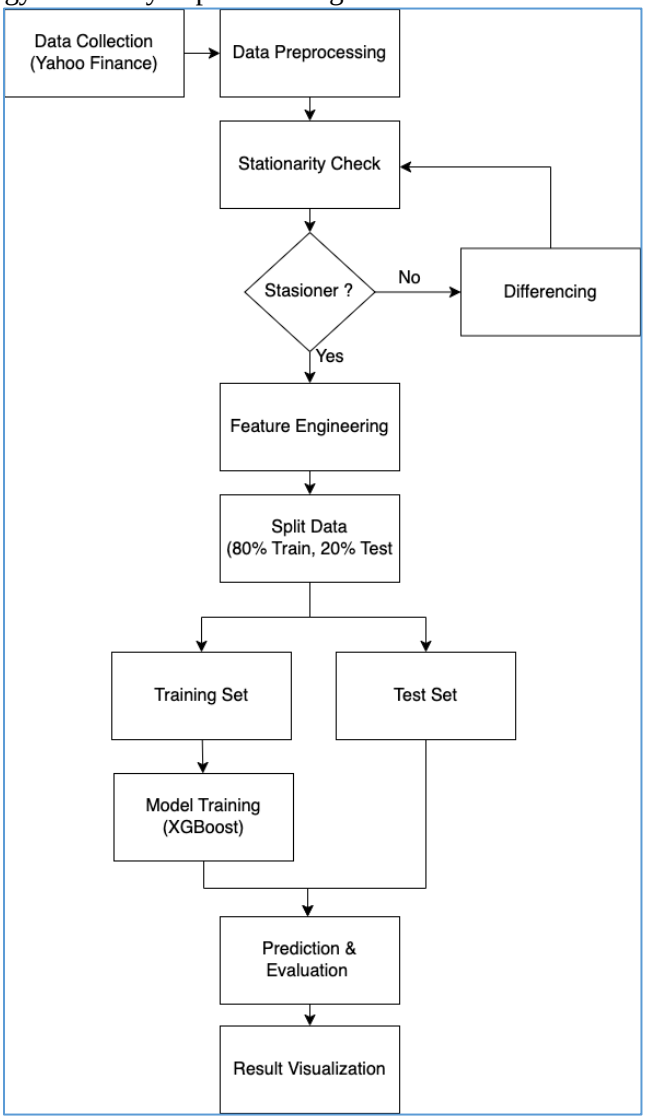

Fig. 1. Research methodology

Data Collection, Bitcoin price information was obtained from Yahoo Finance, covering the period from September 17, 2014, to July 15, 2025. This historical data consists of opening prices, high prices, low prices, closing prices, and daily trading volume. Yahoo Finance is an online platform that provides financial data, news, and analysis, including stock price quotes, press releases, and financial reports. This data is then processed through data cleaning techniques, including handling missing values, outliers, and inconsistencies. The dataset consists of 'date' and 'close' attributes, with the time column set as the index.

ACF Plot are used to determine the MA order, while PACF graphs are used to determine the AR order. Through the ACF and PACF graphs, it can be determined whether the data is stationary or not. Data that is not stationary requires a differencing process.

Data that does not meet stationarity criteria requires transformation or a differencing process to ensure that stationarity assumptions are met. In this study, the data underwent single differencing to achieve stationarity. However, it should be noted that in certain situations, differencing can be performed repeatedly if stationarity has not been achieved after the first application. Feature engineering is done through several techniques such as Fourier transformation, lag features, and statistical window features to extract relevant information from time series data.

Lag features are very useful for time series datasets because they can be used to observe the correlation between current and previous values. Lag features are created by shifting historical data by

 ISSN:2302-8149
e-ISSN:2540-9719

several periods. The lags used are one, two, three hundred sixty-five, and one thousand four hundred sixty. Fourier Terms, to include seasonal components in the model,

Fourier terms are generated using Fourier analysis. Because bitcoin has annual seasonality and halving season, the periods are set to three hundred sixty-five days and one thousand four hundred sixty days, and several pairs of sine and cosine terms are calculated to capture seasonal patterns.

Rolling Features, rolling statistics such as moving averages and standard deviations are also included as features to provide information about Bitcoin price trends and volatility. Moving averages can be used to identify trends in time series data. Data is rolled with a window of 7 and 30 days.

Date parts, to help the model learn different patterns over time, additional features such as day of the week, month of the year, and quarter of the year are extracted from the date column. To accurately evaluate model performance, the data is partitioned into training and testing sets.

The data is divided into two parts, with 80% used for model training and 20% allocated for testing. This division ensures that the model is evaluated on unseen data to assess its ability to generalize to new data points.

The XGBoost model is trained using the training data. Model parameters are optimized through cross-validation techniques, including learning rate, tree depth, number of estimators, subsample, gamma, and random_state.

Evaluate Model, Model performance is evaluated on the test set using appropriate metrics such as Mean Squared Error, Root Mean Squared Error, and R-squared. The coefficient of determination, Mean Square Error, Root Mean Square Error, and Mean Absolute Error are some of the metrics used to evaluate the performance of each model.

In addition to error metrics, graphical visualizations such as actual versus predicted plots and residual plots can provide valuable insights into model performance and identify potential areas for improvement [38].

4 Results and Analysis

This section presents the experimental results of forecasting Bitcoin prices using XGBoost with time series features and Fourier-based seasonality. Bitcoin price data were sourced from Yahoo Finance, spanning from September 17, 2014, to July 15, 2025. The data were obtained through web scraping using the pandas datareader and yfinance packages. This historical data encompasses opening prices, high prices, low prices, closing prices, and daily trading volume.

```
>>>  import yfinance as yf
>>>  import pandas as pd
>>>  btc_data=yf.download("BTC-USD",start="2010-01-01",end="2025-07-15")
>>>  btc_data = btc_data[['Close']]
>>>  btc_data.index.name = 'Date'
```

The Bitcoin data obtained from the program consists of columns such as date, high price, low price, opening price, closing price, and volume. This historical data is then processed through data preprocessing techniques, including handling missing values, outliers, and inconsistencies.

In Data Preprocessing, missing values are addressed through imputation using previous values. In each data entry within the Bitcoin historical price table, the date column is set as an index to simplify the handling of empty values. Subsequently, transformations are applied to ensure stationarity, given that time series models often assume stationary conditions. The differencing process is carried out by subtracting the previous value from each data point.

To identify the Moving Average and Autoregressive orders in the historical Bitcoin price data, Autocorrelation Function and Partial Autocorrelation Function plots are used. The ACF plot helps in identifying the MA order, while the PACF plot is used to determine the AR order. Through these two plots, it can be determined whether the analyzed data is stationary or not. The ACF and PACF plots can be visualized using the plot_acf and plot_pacf functions from the statsmodels.graphics.tsaplots package.

```
>>>  from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
>>>  plt.figure(figsize=(14, 6))
>>>  plt.subplot(1, 2, 1)
>>>  plot_acf(df['Close'].dropna(), lags=60, ax=plt.gca())
>>>  plt.title('Autocorrelation (ACF) - Close')
```

 ISSN:2302-8149
e-ISSN:2540-9719

```
>>>  plt.subplot(1, 2, 2)
>>>  plot_pacf(df['Close'].dropna(),lags=60,method='ywm', ax=plt.gca())
>>>  plt.title('Partial Autocorrelation (PACF) - Close')
>>>  plt.tight_layout()
>>>  plt.show()
```

Based on the aforementioned function, the Autocorrelation Function values exhibit a gradual decline without any statistically significant values beyond the confidence limits, suggesting a strong but decelerating correlation. This condition indicates that the data is non-stationary due to the presence of long-term trends, where the influence of previous prices remains persistent. The Partial Autocorrelation Function values show a statistically significant spike at lag 1, followed by a decrease approaching zero. Movements at lag 2 and beyond are not significant, indicating a correlation between the current and previous data. The results from these plots can be observed in Figure 2.

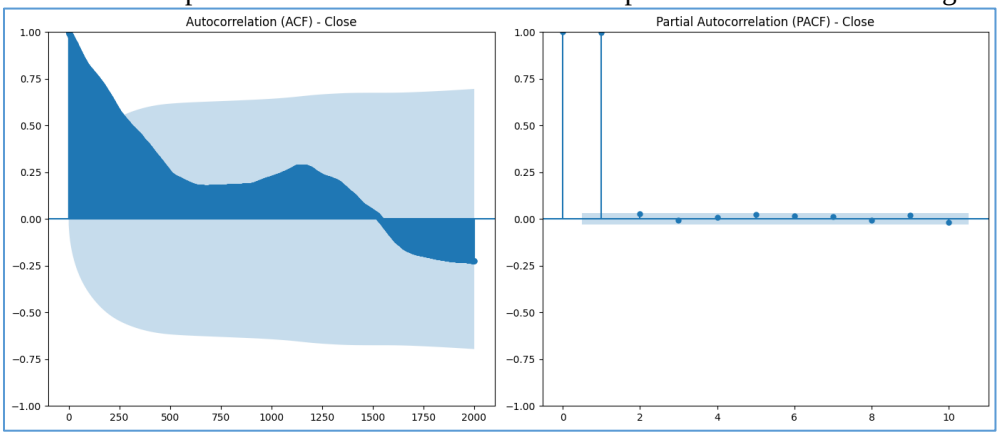

Fig. 2. Plot ACF and PACF

To render the data stationary, differencing was performed once. Utilizing Python, a logarithmic transformation is applied using close_log = np.log, followed by differencing the transformed data with close_log_diff = close_log.diff(). Subsequently, any resulting null values are removed using the dropna() function.

```
>>>  btc_data['Close_log]=np.log(btc_data['Close'])
>>>  btc_data['diff_log']=btc_data['Close_log'].diff()
>>>  btc_data.dropna(inplace=True)
```

Following confirmation of data stationarity and the absence of null values, the subsequent stage involves feature engineering. This process yields four primary features: Lag Features, Fourier Terms, Rolling Features, and Date Parts.

Lag Features, The PACF results, indicating lags of 1 and 2, form the basis for establishing lag values. Furthermore, considering Bitcoin's unique characteristic as an asset with halving events occurring every four years, lags corresponding to the 365th and 1460th days are also incorporated [39].

```
>>>  btc_data['lag_1']   = btc_data['diff_log'].shift(1)
>>>  btc_data['lag_2']   = btc_data['diff_log'].shift(2)
>>>  btc_data['lag_365'] = btc_data['diff_log'].shift(365)
>>>  btc_data['lag_1460']= btc_data['diff_log'].shift(1460)
```

Fourier Terms, to incorporate seasonal components into the model, Fourier terms are generated using Fourier analysis. Given that Bitcoin exhibits annual seasonality and halving seasons, the periods are set to 365 days and 1460 days, and several pairs of sine and cosine terms are computed to capture seasonal patterns.

```
>>>  btc_data['sin_365'] =np.sin(2*np.pi* btc_data.index.dayofyear/365)
>>>  btc_data['cos_365'] =np.cos(2*np.pi* btc_data.index.dayofyear/365)
>>>  btc_data['sin_1460']=np.sin(2*np.pi* btc_data.index.dayofyear/1460)
>>>  btc_data['cos_1460']=np.cos(2*np.pi* btc_data.index.dayofyear/1460)
```

 ISSN:2302-8149
e-ISSN:2540-9719

Rolling statistics, such as moving averages and standard deviations, are incorporated as features to provide insights into Bitcoin price trends and volatility. Specifically, moving averages with windows of 7 and 30 days are computed to capture short-term and medium-term trends in the price data.

```
>>>  btc_data['roll_mean_7'] = btc_data['diff_log'].rolling(7).mean()
>>>  btc_data['roll_std_7']  = btc_data['diff_log'].rolling(7).std()
>>>  btc_data['roll_mean_30']= btc_data['diff_log'].rolling(30).mean()
>>>  btc_data['roll_std_30'] = btc_data['diff_log'].rolling(30).std()
```

Date Parts: Additional features such as the day of the week, month of the year, and quarter of the year are extracted from the date column to aid the model in learning distinct temporal patterns.

```
>>>  btc_data['dayofweek'] = btc_data.index.dayofweek
>>>  btc_data['month']     = btc_data.index.month
>>>  btc_data['day'].      = btc_data.index.day
```

Subsequently, the data is partitioned into two variables, X and y. The X variable encompasses the core data, excluding the 'Close', 'Close_log', and 'diff_log' columns, and focuses on variables derived from feature engineering. The y variable stores the difference log and serves as the target variable for prediction.

```
>>>  X=btc_data.drop(columns=['Close','Close_log','diff_log'])
>>>  y = btc_data['diff_log']
```

Following feature engineering, the data undergoes partitioning into training and testing sets. The variables X and y are divided accordingly, yielding X_train, X_test, y_train, and y_test, which are then prepared for input into the XGBoost model. Additionally, the variable last_close_log is initialized to retain the final value from the close log, facilitating inverse scaling during evaluation.

```
>>>  split = int(0.80 * len(btc_data))
>>>  X_train, X_test = X[:split], X[split:]
>>>  y_train, y_test = y[:split], y[split:]
>>>  last_close_log=btc_data['Close_log'].iloc[split-1]
```

Following data preparation, the XGBoost model is trained employing the subsequent parameters: n_estimators=1000, learning_rate=0.05, max_depth=5, subsample=0.8, colsample_bytree=0.8, and random_state=42. The n_estimators parameter, set at 1000, signifies the construction of 1000 trees within the ensemble. A learning_rate of 0.05 is implemented to modulate the contribution of each tree to the final model, aiming to mitigate overfitting. A max_depth of 5 restricts the depth of each tree to manage model complexity. A subsample of 0.8 indicates that 80% of the training data will be utilized for training each tree, also with the intention of reducing overfitting. A colsample_bytree value of 0.8 suggests that 80% of the features will be employed to train each tree, further assisting in the reduction of overfitting. The parameter random_state=42 is employed to ensure reproducible results. Utilizing the XGBoost library, the model is fitted using X_train and y_train.

```
>>>  import xgboost as xgb
>>>  model = xgb.XGBRegressor(
         n_estimators    =1000,
         learning_rate   =0.01,
         max_depth       =4,
         Subsample       =0.8,
         colsample_bytree=0.8,
         Gamma           =0.1,
         reg_alpha       =0.1,
         reg_lambda      =1.0,
         random_state    =42,
         Verbosity       =1
     )
>>>  model.fit(X_train, y_train)
```

 ISSN:2302-8149
e-ISSN:2540-9719

Following the training of the model, the subsequent phase involves generating predictions on the test set to derive the y_pred_logdiff variable. Given that the model predicts the logarithmic differences in price, these predictions are then converted back to the original price scale, utilizing the last logarithmic price value from the training data.

The variable y_pred_price forecasts the exponential price values. These forecasts are subsequently compared against the actual prices in the test set to assess the model's performance. The evaluation is conducted using Root Mean Squared Error and Mean Absolute Error.

```
>>> from sklearn.metrics import mean_squared_error, mean_absolute_error
>>> y_pred_logdiff = model.predict(X_test)
>>> y_pred_price   = np.exp(np.cumsum(y_pred_logdiff) + last_close_log)
>>> y_true_price   = btc_data['Close'].iloc[split:]
>>> rmse       =np.sqrt(mean_squared_error(y_true_price, y_pred_price))
>>> mae        =mean_absolute_error(y_true_price, y_pred_price)
>>> print(f"RMSE (Price): {rmse:,.2f}")
>>> print(f"MAE  (Price): {mae:,.2f}")
```

The Python execution yielded a Root Mean Squared Error of 11,823.11 and a Mean Absolute Error of 9,891.17. A validation stage was implemented to assess the model's accuracy [40]. These findings suggest that the proposed model exhibits a capacity to forecast Bitcoin prices, with a mean deviation of 9,891.17 and a root mean square error of 11,823.11, providing a quantitative framework for interpreting prediction accuracy within the context of Bitcoin price fluctuations.

Given the error values of 9,891.17 and 11,823.11, and considering that the weekly Bitcoin price on July 15, 2025, ranged from US$ 116,144.02 to US$ 123,237.83, the relative percentage concerning the actual price can be calculated accordingly.

$$Mean\ Price = \frac{123{,}237.83\ +\ 116{,}144.02}{2} = \$\,119{,}690.93$$

$$RMSE\ = \frac{11{,}823.11}{119{,}690.93}\ x\ 100\ \approx 9.87\ \%$$

$$MAE = \frac{9{,}891.17}{119{,}690.93}\ x\ 100\ \approx 8.26\ \%$$

The observed Mean Absolute Error, registering below 10%, may be cautiously interpreted as adequate for forecasting within a medium-term horizon. However, its utility diminishes in applications demanding heightened short-term precision, where this magnitude of error may be considered relatively significant. In light of Bitcoin's inherent price volatility, these error values may be deemed tolerable, particularly when contextualized within broader trend analysis or strategic decision-making frameworks.

Within the context of portfolio rebalancing strategies, a forecast exhibiting an 8–9% error margin may still offer valuable directional insights, particularly for establishing entry and exit thresholds or fine-tuning crypto-to-cash ratios. Analogously, in scenarios such as monthly investment allocation or risk-adjusted position sizing, a 10% error tolerance is frequently deemed acceptable to facilitate high-level decision-making processes without necessitating intraday precision. Conversely, high-frequency trading systems typically demand substantially lower error tolerances.

Figure 3 presents a compelling visual comparison between the actual and predicted time series data, revealing a strong concordance between the two trajectories. This alignment highlights the efficacy of the XGBoost model in discerning the complex, non-linear dynamics inherent in cryptocurrency markets, particularly when augmented with comprehensive time series features and Fourier-based seasonal components [29]. The visualization facilitates a qualitative assessment of the model's predictive capabilities and identifies areas for potential refinement. Given the stochastic nature and inherent volatility of Bitcoin as a high-risk asset, the observed error margins of 9,891.17 and 11,823.11 are considered within an acceptable threshold.

 ISSN:2302-8149
e-ISSN:2540-9719

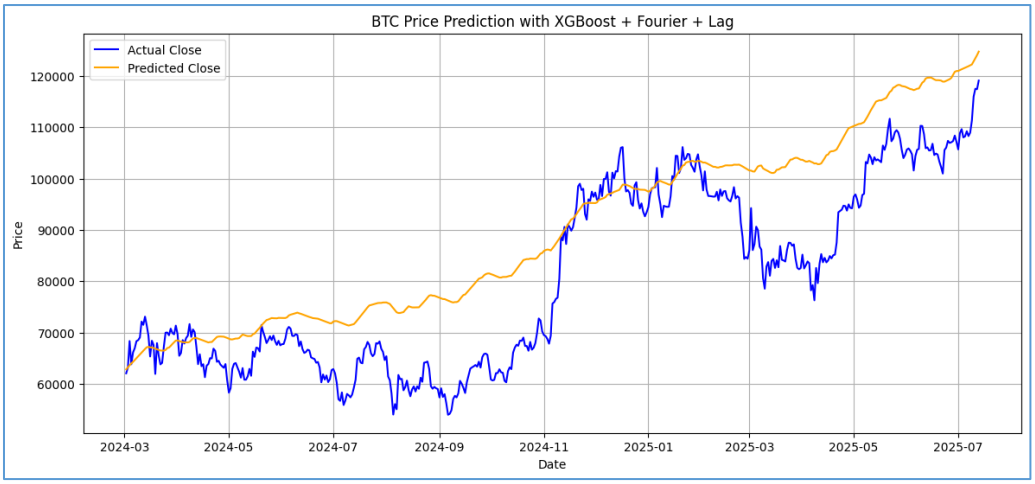

Fig. 3. Visualization of time series actual data vs. predicted data

5 Conclusion

Based on the Bitcoin price forecasting study using XGBoost that integrates time series and Fourier-based seasonal features, it can be concluded that the XGBoost model demonstrates significant potential in predicting Bitcoin prices, as indicated by an RMSE of 11,823.11 and an MAE of 9,891.17. These findings indicate the model's ability to generate Bitcoin price predictions with a moderate level of accuracy; nevertheless, there remains room for further improvement. The implementation of time series features, including rolling features and date parts, as well as Fourier-based seasonality, proves effective in enhancing the performance of the XGBoost model. These features contribute to the model's ability to capture temporal patterns and trends in Bitcoin price data, resulting in more accurate predictions. In addition, the volume of data can affect the level of accuracy. For further research, it is recommended that the level of error significance be minimized to below 5% through the addition of more detailed Fourier terms, or through a hybrid approach with SARIMA or LSTM models.

Reference

[1] Q. Guo, S. Lei, Q. Ye, and Z. Fang, "MRC-LSTM: *A Hybrid Approach of Multi-scale Residual CNN and LSTM to Predict Bitcoin Price*," 2022 International Joint Conference on Neural Networks (IJCNN), p. 1, Jul. 2021, DOI: 10.1109/ijcnn52387.2021.9534453.

[2] M. Iqbal, M. H. Iqbal, F. H. Jaskani, K. Iqbal, and A. Hassan, "*Time-Series Prediction of Cryptocurrency Market using Machine Learning Techniques*," EAI Endorsed Transactions on Creative Technologies, Vol. 8, No. 28, p. 170286, Jul. 2021, DOI: 10.4108/eai.7-7-2021.170286.

[3] A. Bouteska, M. Z. Abedin, P. Hájek, and K. Yuan, "*Cryptocurrency Price Forecasting – A Comparative Analysis of Ensemble Learning and Deep Learning Methods*," International Review of Financial Analysis, Vol. 92, p. 103055, Dec. 2023, DOI: 10.1016/j.irfa.2023.103055.

[4] H. S. Jung, J. H. Kim, and H. Lee, "*Decoding Bitcoin: Leveraging Macro-and Micro-Factors in Time Series Analysis for Price Prediction*," PeerJ Computer Science, Vol. 10, Sep. 2024, DOI: 10.7717/peerj-cs.2314.

[5] S. Rikli, D. N. Bigler, M. Pfenninger, and J. Osterrieder, "*Wasserstein GAN: Deep Generation Applied on Bitcoins Financial Time Series*," arXiv (Cornell University), Jan. 2021, DOI: 10.48550/arxiv.2107.06008.

[6] A. L. Putra and A. Kurniawati, "Analisis Prediksi Harga Saham PT. Astra International Tbk menggunakan Metode *Autoregressive Integrated Moving Average* (ARIMA) dan *Support Vector Regression* (SVR)," Jurnal Ilmiah Komputasi, Vol. 20, No. 3, Sep. 2021, DOI: 10.32409/jikstik.20.3.2732.

[7] G. Cohen and A. Aiche, "*Predicting the Bitcoin's Price using AI*," Frontiers in Artificial Intelligence, Vol. 8, Apr. 2025, DOI: 10.3389/frai.2025.1519805.

[8] D. R. Chandranegara, R. A. Afif, C. S. K. Aditya, W. Suharso, and H. Wibowo, "Prediksi Harga Saham Jakarta Islamic Index menggunakan Metode *Long Short-Term Memory*," Jurnal Edukasi

 ISSN:2302-8149
e-ISSN:2540-9719

dan Penelitian Informatika (JEPIN), Vol. 9, No. 1, p. 129, Apr. 2023, DOI: 10.26418/jp.v9i1.57561.

[9] S. Sudaryono, Q. Aini, N. Lutfiani, F. Hanafi, and U. Rahardja, "*Application of Blockchain Technology for iLearning Student Assessment*," IJCCS (Indonesian Journal of Computing and Cybernetics Systems), Vol. 14, No. 2, p. 209, Apr. 2020, DOI: 10.22146/ijccs.53109.

[10] Muntafiah, S. Bakri, and A. Rais, "Tinjauan Transaksi *Crypto Currency* berbasis Keabsahan Kontemporer Syariah" Academica Journal of Multidisciplinary Studies, Vol. 6, No. 2, p. 335, Dec. 2022, DOI: 10.22515/academica.v6i2.5931.

[11] R. Böhme, N. Christin, B. Edelman, and T. Moore, "*Bitcoin: Economics, Technology, and Governance*," The Journal of Economic Perspectives, Vol. 29, No. 2, p. 213, Apr. 2015, DOI: 10.1257/jep.29.2.213.

[12] D. Shah, D. Patel, J. Adesara, P. Hingu, and M. Shah, "*Integrating Machine Learning and Blockchain to Develop a System to Veto the Forgeries and Provide Efficient Results in Education Sector*," Visual Computing for Industry Biomedicine and Art, Vol. 4, No. 1, Jun. 2021, DOI: 10.1186/s42492-021-00084-y.

[13] S. Laksono and W. Candra, "*Artificial Intelligence* dan *Kardiologi: A Mini Review*," Health & Medical Journal, Vol. 4, No. 2. p. 130, May 01, 2022. DOI: 10.33854/heme.v4i2.964.

[14] R. Lou, Z. Lv, S. Dang, T. Su, and X. Li, "*Application of Machine Learning in Ocean Data*," Multimedia Systems, Vol. 29, No. 3, p. 1815, Feb. 2021, DOI: 10.1007/s00530-020-00733-x.

[15] J. M. Ahn, J. Kim, and K. Kim, "*Ensemble Machine Learning of Gradient Boosting (XGBoost, LightGBM, CatBoost) and Attention-based CNN-LSTM for Harmful Algal Blooms Forecasting*," Toxins, Vol. 15, No. 10, p. 608, Oct. 2023, DOI: 10.3390/toxins15100608.

[16] A. Ferrario and R. Hämmerli, "*On Boosting: Theory and Applications*," SSRN Electronic Journal, Jan. 2019, DOI: 10.2139/ssrn.3402687.

[17] T. Chen and C. Guestrin, "*XGBoost*," p. 785, Aug. 2016, DOI: 10.1145/2939672.2939785.

[18] J. Ma, Z. Yu, Y. Qu, J. Xu, and Y. Cao, "*Application of the XGBoost Machine Learning Method in PM2.5 Prediction: A Case Study of Shanghai*," Aerosol and Air Quality Research, Vol. 20, No. 1, p. 128, Dec. 2019, DOI: 10.4209/aaqr.2019.08.0408.

[19] P. Hu et al., "*A Comparison of LASSO Regression and Tree-based Models for Delayed Cerebral Ischemia in Elderly Patients with Subarachnoid Hemorrhage*," Frontiers in Neurology, Vol. 13, Mar. 2022, DOI: 10.3389/fneur.2022.791547.

[20] A. Anghel, N. Papandreou, T. Parnell, A. D. Palma, and H. Pozidis, "*Benchmarking and Optimization of Gradient Boosted Decision Tree Algorithms*," arXiv (Cornell University), Sep. 2018, Accessed: Feb. 2025. [Online]. Available: https://arxiv.org/pdf/1809.04559.pdf

[21] Y. Kang, M. Özdoğan, X. Zhu, Z. Ye, C. Hain, and M. C. Anderson, "*Comparative Assessment of Environmental Variables and Machine Learning Algorithms for Maize Yield Prediction in the US Midwest*," Environmental Research Letters, Vol. 15, No. 6, p. 64005, Mar. 2020, DOI: 10.1088/1748-9326/ab7df9.

[22] R. Holla, "*Advanced Feature Engineering for Time Series Data*." Jun. 2024. Accessed: Jul. 14, 2025. [Online]. Available: https://medium.com/@rahulholla1/advanced-feature-engineering-for-time-series-data-5f00e3a8ad29

[23] J. Heaton, "*An Empirical Analysis of Feature Engineering for Predictive Modeling*," SoutheastCon, p. 1, Mar. 2016, DOI: 10.1109/secon.2016.7506650.

[24] R. Adawiyah, M. Muljono, and W. E. Nugroho, "*Algoritma Naive Bayes untuk memprediksi Bimbingan Konseling Siswa Sekolah Menengah Kejuruan*," Smart Comp Jurnalnya Orang Pintar Komputer, Vol. 12, No. 3, p. 723, Jul. 2023, DOI: 10.30591/smartcomp.v12i3.5365.

[25] D. A. Wibisono, D. Anggraeni, and A. F. Hadi, "Perbaikan Model *Seasonal* ARIMA dengan Metode *Ensemble Kalman Filter* pada Hasil Prediksi Curah Hujan" Majalah Ilmiah Matematika dan Statistika, Vol. 19, No. 1, p. 9, Mar. 2019, DOI: 10.19184/mims.v19i1.17262.

[26] V. Naghashi, M. Boukadoum, and A. B. Diallo, "*A Multiscale Model for Multivariate Time Series Forecasting*," Scientific Reports, Vol. 15, No. 1, Jan. 2025, DOI: 10.1038/s41598-024-82417-4.

[27] V. M. Manikandan and S. Neethirajan, "*Decoding Poultry Welfare from Sound—A Machine Learning Framework for Non-Invasive Acoustic Monitoring*," Sensors, Vol. 25, No. 9, p. 2912, May 2025, DOI: 10.3390/s25092912.

 ISSN:2302-8149
e-ISSN:2540-9719

[28] U. M. Rifanti, H. Pujiharsono, A. W. Setiawan, and J. Hendry, "*Implementasi Moving Average Filter* untuk Koreksi Kesalahan Sensor Pengukur Kedalaman Air," ELKOMIKA Jurnal Teknik Energi Elektrik Teknik Telekomunikasi & Teknik Elektronika, Vol. 8, No. 2, p. 432, May 2020, DOI: 10.26760/elkomika.v8i2.432.
[29] S. O. Odhiambo, N. O. Cornelious, and H. Waititu, "*Developing a Hybrid ARIMA-XGBOOST Model for Analysing Mobile Money Transaction Data in Kenya*," Asian Journal of Probability and Statistics, Vol. 26, No. 10, p. 108, Oct. 2024, DOI: 10.9734/ajpas/2024/v26i10662.
[30] R. K. Klimberg and S. J. Ratick, "*Development of a Practical and Effective Forecasting Performance Measure*," in Advances in business and management forecasting, Emerald Publishing Limited, 2017, p. 103. DOI: 10.1108/s1477-407020170000012007.
[31] F. Rodrigues and M. P. Machado, "*High-Frequency Cryptocurrency Price Forecasting using Machine Learning Models: A Comparative Study*," Mar. 2025, DOI: 10.20944/preprints202503.0261.v1.
[32] F. Annasiyah and M. Prastuti, "Peramalan Konsumsi Energi Listrik untuk Sektor Industri di PT PLN (Persero) Area Gresik menggunakan Metode *Time Series Regression* dan ARIMA," Jurnal Sains dan Seni ITS, Vol. 12, No. 1, May 2023, DOI: 10.12962/j23373520.v12i1.104264.
[33] C. S. Kalengkongan, Y. Langi, and N. Nainggolan, "Analisis Volatilitas Harga Bawang Putih di Kota Manado menggunakan Model GARCH," d Cartesian, Vol. 9, No. 1, p. 43, Jan. 2020, DOI: 10.35799/dc.9.1.2020.27398.
[34] Z. Hossain, A. Rahman, Md. M. Hossain, and J. H. Karami, "*Over-Differencing and Forecasting with Non-Stationary Time Series Data*," Dhaka University Journal of Science, Vol. 67, No. 1, p. 21, Jan. 2019, DOI: 10.3329/dujs.v67i1.54568.
[35] M. Z. Rusminto, S. A. Wibowo, and F. S. Wahyuni, "Peramalan Harga Saham menggunakan Metode ARIMA (*Autoregressive Integrated Moving Average*) *Time Series*", Jurnal Mahasiswa Teknik Informatika, Vol. 8, No. 2, p. 1263, Mar. 2024, DOI: 10.36040/jati.v8i2.9089.
[36] D. Pratiwi, S. M. U. Agustini, W. Windasari, and E. N. Kencana, "*Forecasting Farmer Exchange Rate in Bali Province using Seasonal Autoregressive Integrated Moving Average (SARIMA) Method*," Journal of Physics Conference Series, Vol. 1503, No. 1, p. 12002, Jul. 2020, DOI: 10.1088/1742-6596/1503/1/012002.
[37] Y. A. Sir and A. H. H. Soepranoto, "Pendekatan Resampling Data untuk menangani Masalah Ketidakseimbangan Kelas," Jurnal Komputer dan Informatika, Vol. 10, No. 1, p. 31, Mar. 2022, DOI: 10.35508/jicon.v10i1.6554.
[38] D. Hediyati and I. M. Suartana, "Penerapan *Principal Component Analysis* (PCA) untuk Reduksi Dimensi pada Proses *Clustering* Data Produksi Pertanian di Kabupaten Bojonegoro," Journal of Information Engineering and Educational Technology, Vol. 5, No. 2, p. 49, Dec. 2021, DOI: 10.26740/jieet.v5n2.p49-54.
[39] P. K. Handayani, "Penerapan *Principal Component Analysis* untuk Peningkatan Kinerja *Algoritma Decision Tree* pada Iris *Dataset*," Indonesian Journal of Technology Informatics and Science (IJTIS), Vol. 1, No. 2, p. 55, Jun. 2020, DOI: 10.24176/ijtis.v1i2.4939.
[40] W. Astuti and A. Adiwijaya, "*Principal Component Analysis sebagai Ekstraksi Fitur Data Microarray untuk Deteksi Kanker berbasis Linear Discriminant Analysis*," Jurnal Media Informatika Budidarma, Vol. 3, No. 2, p. 72, Apr. 2019, DOI: 10.30865/mib.v3i2.1161.
[41] C. Li, "*Preprocessing Methods and Pipelines of Data Mining: An Overview*," arXiv (Cornell University), Jan. 2019, DOI: 10.48550/arxiv.1906.08510.
[42] D. anon and R. Ślepaczuk, "*Hybrid Models for Financial Forecasting: Combining Econometric, Machine Learning, and Deep Learning Models*," Jan. 2025, DOI: 10.2139/ssrn.5268691.
[43] R. Paita, N. Nainggolan, and Y. Langi, "*Model Space Time Autoregressive* (STAR) Orde 1 dan Penerapannya pada Prediksi Harga Beras di Kota Manado, Tomohon dan Kabupaten Minahasa Utara," d CARTESIAN, Vol. 3, No. 1, p. 17, Feb. 2014, DOI: 10.35799/dc.3.1.2014.3798.
[44] M. I. Naufal and C. Susetyo, "Prediksi Perubahan Penutupan Lahan Pasca Beroperasinya Gerbang Tol (*Interchange*) Pandaan di Kecamatan Pandaan Kabupaten Pasuruan menggunakan Metode Regresi Logistik Biner," Jurnal Teknik ITS, Vol. 9, No. 1, Aug. 2020, DOI: 10.12962/j23373539.v9i1.48278.