XgBoost Hyper-Parameter Tuning Using Particle Swarm Optimization for Stock Price Forecasting

Dwi Pebrianti[1,2], Haris Kurniawan[1], Luhur Bayuaji[1], Rusdah Rusdah[1]
[1]Faculty of Information Technology Universitas Budi Luhur, Jl. Ciledug Raya, Jakarta Selatan, Indonesia
[2]Department of Mechanical and Aerospace Engineering International Islamic University Malaysia, Selangor, Malaysia

ARTICLE INFO

Article history:

Received November 29, 2023
Revised December 28, 2023
Published January 09, 2024

Keywords:

Forecasting;
Stock market;
Time-series data;
XGBoost;
Particle Swarm Optimization

ABSTRACT

Investment in the capital market has become a lifestyle for millennials in Indonesia as seen from the increasing number of SID (Single Investor Identification) from 2.4 million in 2019 to 10.3 million in December 2022. The increase is due to various reasons, starting from the Covid-19 pandemic, which limited the space for social interaction and the easy way to invest in the capital market through various e-commerce platforms. These investors generally use fundamental and technical analysis to maximize profits and minimize the risk of loss in stock investment. These methods may lead to problem where subjectivity and different interpretation may appear in the process. Additionally, these methods are time consuming due to the need in the deep research on the financial statements, economic conditions and company reports. Machine learning by utilizing historical stock price data which is time-series data is one of the methods that can be used for the stock price forecasting. This paper proposed XGBoost optimized by Particle Swarm Optimization (PSO) for stock price forecasting. XGBoost is known for its ability to make predictions accurately and efficiently. PSO is used to optimize the hyper-parameter values of XGBoost. The results of optimizing the hyper-parameter of the XGBoost algorithm using the Particle Swarm Optimization (PSO) method achieved the best performance when compared with standard XGBoost, Long Short-Term Memory (LSTM), Support Vector Regression (SVR) and Random Forest. The results in RSME, MAE and MAPE shows the lowest values in the proposed method, which are, 0.0011, 0.0008, and 0.0772%, respectively. Meanwhile, the R^2 reaches the highest value. It is seen that the PSO-optimized XGBoost is able to predict the stock price with a low error rate, and can be a promising model to be implemented for the stock price forecasting. This result shows the contribution of the proposed method.

Corresponding Author:

Dwi Pebrianti, Faculty of Information Technology Universitas Budi Luhur, Jl. Ciledug Raya, Jakarta Selatan, Indonesia
Email: dwi.pebrianti@budiluhur.ac.id, dwipebrianti@iium.edu.my

1. INTRODUCTION

Three key tenets in financial theory include the weak form efficiency, semi-strong form efficiency and strong form efficiency. In weak form efficiency, prices of financial assets already reflect all historical information, including past prices and trading volumes. Semi-strong form efficiency mentions that in addition to historical information, all publicly available information is already reflected in asset prices. While strong form efficiency mentions that all information, whether public or private, is fully reflected in asset prices. Existence of the anti-thesis, the Efficient Market Hypothesis (EMH) by Fama stated that in the condition of an efficient market then stock price prediction is merely impossible because the historical price or any publicly historical data related to stock market is freely and publicly available [1]. Additionally, the market would react almost immediately to any new information, therefore the stock price is already fully reflected by all the publicly available information. The accurate prediction of stock prices has long been regarded as a challenging endeavor in financial research, analogous to a complex puzzle within the realm of market dynamics. The inherent complexities arising from multifaceted factors, including market sentiment, economic indicators, and unforeseen events, have rendered stock price prediction an intricate and tricky. But that did not stop researchers, especially data scientists, from putting every effort to beat the market using various machine learning methods. Methods as simple as linear regression to complex regression model such as Support Vector Regression (SVR), Neural Network based method specifically sequence model such as Long Short-Term Memory (LSTM), and advanced decision tree-based model such as XGBoost are known to be used for this task.

Meanwhile, the last three years have been a roller coaster ride for the Indonesian stock market. Impacted by the pandemic Covid19, Indonesian composite index plunged to 3,938 which was the lowest point in 10 years at 24th of March 2020, just three weeks after the first Covid19 case was announced in Indonesia. While 5 days earlier, on 19th of March 2020, the trading volumes dropped to the lowest point.

However, the Covid19 outbreak that continues to hit Indonesia has not dampened the interest of investors in Indonesia to continue investing in shares. In the following days, the condition of the Indonesian capital market began to recover and gradually improved. The number of investors in the capital market also continues to increase, as indicated by the increase in the number of Single Investor Identification (SID) as can be seen in Fig. 1. From 2.4 million SID in 2019 to 3.8 million SID in 2020, then increasing to 7.4 million SID in 2021, and 10.3 million SID in 2022.

The amount of money turnover in Indonesian capital market has also experienced a significant increase as can be seen in Table 1. After previously decreasing from 2,230 trillion Rupiah in 2019 to 2,228 trillion Rupiah in 2020, it rose again to 3,303 trillion Rupiah in 2021 and 3,617 trillion Rupiah in 2022. Likewise with the increase in the value of Indonesian stock market capitalization as can be seen in Fig. 2,

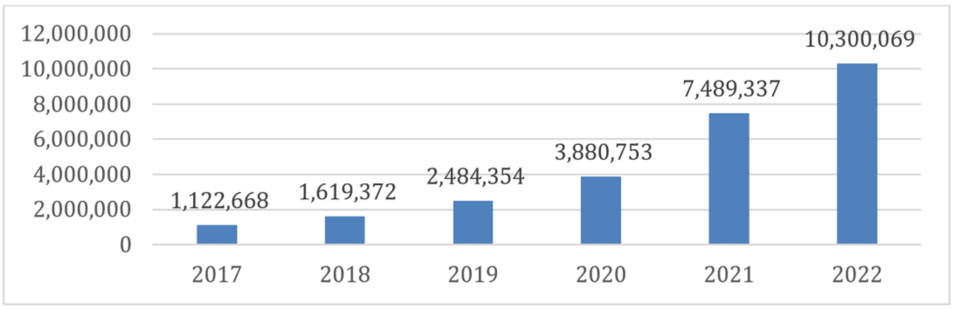


Fig. 1. The Total Single Investor Identification (SID) Number from 2017 to 2022

Tabel 1. Statistic of Indonesian Stock Market from 2019 to 2022

Jakarta Composite Index	**2019**	**2020**	**2021**	**2022**
High	6,525	6,325	6,723	7,318
Low	5,827	3,938	5,761	6,568
Close	6,299	5,979	6,581	6,851
Stock Market Summary	**2019**	**2020**	**2021**	**2022**
Trading Day	245	242	247	246
Listed Companies, *eop*	668	713	766	825
Listed Shares, *m.shares*	5,736,081	6,054,120	7,227,250	9,704,602
Market Capitalization, *b.IDR*	7,265,016	6,970,009	8,255,624	9,499,139
Stock Trading, total	**2019**	**2020**	**2021**	**2022**
Volume, *m.Shares*	3,562,369	2,752,471	5,096,688	5,885,863
Value, *b.IDR*	2,230,919	2,228,798	3,303,362	3,617,897
Frequency, *times*	114,857,097	163,937,983	319,852,452	321,322,600

Which in fact, it managed to set a record after it had fallen in the early days of the Covid-19 pandemic from 7,265 trillion Rupiah in 2019, down in 2020 to 6,970 trillion Rupiah, finally increasing significantly at the end of 2022 to 9,499 trillion Rupiah.

Despite a significant dip in the Indonesian stock market in March 2020 due to the impact of the Covid-19 pandemic, there has been a remarkable recovery. The number of investors and Single Investor Identifications (SID) has steadily increased, and both the money turnover and market capitalization have seen substantial growth, indicating a resilient and thriving market in the aftermath of the pandemic. Additionally, from the increase in SID and transaction volume in the last two years, it can be concluded that stock investment has become the new favorite among millennials. Stock investors usually rely on two tools in determining stock investment strategies, which are fundamental analysis and technical analysis. Fundamental analysis will consider data from financial reports and macroeconomic conditions that will be used for long-term investment decisions. On the other hand, technical analysis uses a lot of historical data such as stock price movements with the goal of predicting future stock price directions. Given the challenges posed by stock price volatility, which sometimes exhibits intricate and hard-to-explain patterns, predicting stock prices becomes crucial. Understanding these patterns is vital for mitigating investment risks, including the potential capital loss resulting from selling shares at a price lower than the purchase price. The recent trends in the Indonesian stock market, as highlighted, underscore the significance of accurate stock price predictions in navigating and capitalizing on market fluctuations.

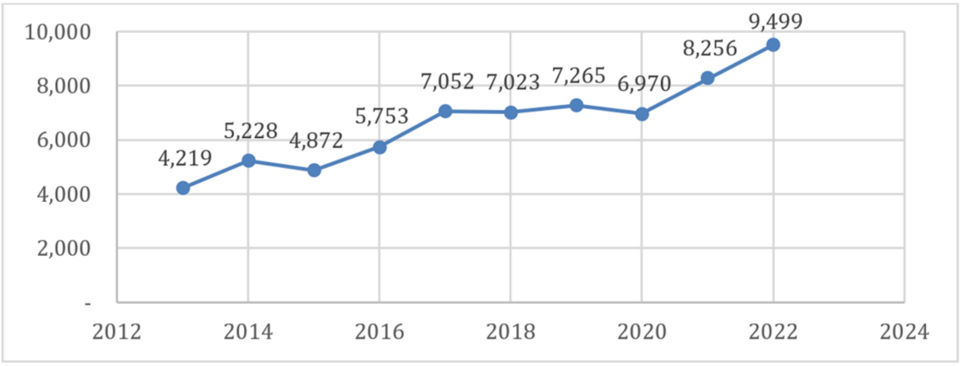


Fig. 2. Capitalization of Indonesian Stock Market from 2013 to 2022 (in Trillion Rupiah)

Several works related to stock price forecasting have been conducted all over the world. Several methods have been produced and implemented in predicting the stock price. The methods can be divided into four categories which are regression, neural network based, tree-based and deep learning.

The literature, as outlined in references [2]–[8], showcases the efficacy of both linear and complex regression models in stock price prediction. Linear regression, as demonstrated by Bhuriya and Vinay, yields high accuracy, reaching confidence values of 98% and 95%, respectively [2], [3]. Conversely, Support Vector Regression (SVR), discussed in references [5]–[8], achieves impressive results with the smallest Mean Average Error (MAE) reported by Bezerra at 0.000205 and 70% accuracy on Yahoo Finance stock data [5]. Additionally, a swarm intelligence-based optimization algorithm, which is firefly was used to tune the hyper-parameters of SVR [9]. The choice between these models may hinge on task-specific requirements, with linear regression offering simplicity and high accuracy, while complex models like SVR bring advanced techniques for enhanced predictive performance.

The literature review investigates enhancements to Support Vector Machines (SVM) for stock price prediction, utilizing optimization algorithms like Imperialist Competition (ICA), Genetic Algorithm (GA), and Teaching & Learning Based Optimization (TLBO). Combining SVM with ICA and GA resulted in improved prediction performance compared to feed-forward static neural networks [10]. PCA-SVM, incorporating Principal Component Analysis for data cleaning, demonstrated higher accuracy than PCA-ANN [11]. The integration of TLBO with SVM reduced the burden of setting hyper-parameters and achieved significant improvements in Mean Absolute Error (MAE) for various forecast periods [12]. A novel multi-kernel SVM with automatic parameter tuning based on fruit fly optimization (FFO) outperformed other optimization methods like SVM-PSO and SVM-GA [13]. Additionally, metaheuristics-based SVM models, particularly the combination of Ant Bee Colony (ABC) with ANFIS and SVM, were effective in addressing weaknesses in traditional approaches to stock price prediction [14]. These approaches consistently demonstrate improved performance compared to traditional methods, as evidenced by reduced Mean Absolute Error (MAE) and enhanced accuracy across different forecast periods. The emphasis on addressing weaknesses in existing prediction models, including the need for automated parameter tuning, efficient data cleaning, and the incorporation of metaheuristics, highlights the necessity to continually develop and refine prediction models to meet the challenges posed by stock market dynamics and evolving research findings.

Stock price prediction based on neural network model has been discussed in [7], [15]–[21]. Convolution Neural Network was combined with LSTM in [22]–[24]. A novel method which implements Fuzzy time series (FTS), Genetic Algorithm (GA) and Fuzzy C means (FCM) was combined with multifactor Back Propagation Neural Network (BPNN) [25]. Long Short-Term Memory (LSTM), hybrid model between Genetic Algorithm (GA) and Artificial Neural Network (ANN), Dynamic neural network and Multi Layer Perceptron (MLP) NN, modification of BPNN with fuzzy were discussed in the study. The results show that LSTM can improve the accuracy of the stock price prediction compared to traditional methods. This is because neural network is suited for the time series data, which is the nature of stock price data. NN can capture the temporal dependencies and hidden patterns in historical price movements. Additionally, neural networks can adapt and learn from large datasets which makes NN effective in modeling the intricate relationships that influence stock market dynamics. However, NN can be sensitive to the quality and quantity of data, and overfitting may occur if the model is too complex relative to the available data.

Chalvatzis and Hristu-Varsakelis present an innovative automated trading architecture that shifts focus from accuracy to profitability in the prediction model, revealing the importance of aligning model performance with trading strategy. Their study on S&P 500, Dow Jones, NASDAQ, and Russell 2000 indices from 2010 to 2019 demonstrates cumulative returns surpassing benchmark strategies, emphasizing the need for a nuanced approach in developing predictive models tailored to trading objectives [26]. The research explores the challenges of predicting stock prices amidst market volatility by framing the problem as a directional prediction exercise, leveraging classification algorithms, random forests, and gradient-boosted decision trees. The incorporation of technical indicators and feature engineering contributes to enhanced accuracy, particularly for medium to long-term predictions, reinforcing the importance of refining models to tackle intricate market dynamics [27]. A proposed hybrid GA-XGBoost prediction system further supports the need for advanced feature engineering, data preparation, and optimal feature selection to improve prediction performance, emphasizing the ongoing necessity for model development in addressing complex financial data characteristics [28]. Additionally, the focus on gradient boosting algorithms, particularly XGBoost, in identifying long-term trends underscores the importance of fine calibration and ongoing research in feature engineering for designing effective investment strategies [29]. The popularity of tree-based models like XGBoost in stock price prediction is acknowledged for their interpretability, ability to capture complex relationships, and handling of missing data. However, the study highlights the importance of addressing potential limitations such as overfitting and interpretability challenges in ensemble methods, emphasizing the need for careful consideration in developing robust predictive models for financial data [28], [29].

Deep learning was introduced to the stock price prediction model in [30]–[33]. Long Short-Term Memory (LSTM) networks, a sophisticated sequence learning technique, are applied to predict out-of-sample directional movements in the constituent stocks of the S&P 500 from 1992 to 2015. The LSTM networks demonstrate superior performance compared to memory-free classification methods, including random forest, deep neural net, and logistic regression classifier, achieving daily returns of 0.46 percent and a Sharpe ratio of 5.8 before transaction costs [34]. An improved Stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms is proposed in [35]. A hybrid model integrating deep learning with investor sentiment analysis for stock price prediction [36]. The study incorporates technical features and macroeconomic indicators into the framework, combining tree-based algorithms (Random Forest, Extremely Randomized Trees, Extreme Gradient Boosting, Light Gradient Boosting Machine) and deep learning algorithms (Recurrent Neural Networks, Bidirectional RNN, RNN with Long Short-Term Memory, Gated Recurrent Unit) as base classifiers in the first layer. The second layer employs logistic regression and its regularized version as meta-classifiers through cross-validation to prevent overfitting. The improved Stacking method outperforms existing ensemble learning algorithms and deep learning models in terms of accuracy, F-score, and AUC value. An improve deep learning for stock prediction [37], short term stock market price using comprehensive deep learning [38] and stock prediction using deep learning [39], [40], were actively conducted. The strengths of deep learning models lie in their ability to automatically learn hierarchical representations from data, adapt to varying market conditions, and handle complex relationships that may be challenging for traditional models. These characteristics make deep learning particularly suitable for tasks like stock price prediction where patterns can be subtle and evolve over time. However, challenges exist in this domain. Financial markets are influenced by numerous unpredictable factors, and historical price movements may not always be indicative of future behavior. Overfitting is also a concern, especially when dealing with limited datasets. It's crucial to carefully preprocess data, select appropriate architectures, and implement robust validation strategies to ensure the reliability of deep learning models. Combining deep learning with other traditional techniques or ensemble methods can be a prudent approach to enhance predictive accuracy.

Financial markets are dynamic and influenced by various factors, continuous research in stock price prediction is essential. Introducing new methodologies and optimization techniques contributes to the evolution of the field and provides researchers and practitioners with additional tools for making informed investment decisions. Based on the literature, it is known that there is still plenty of potential research that can be conducted in the field of stock price forecasting. This study aims to combine XGBoost with PSO to create a hybrid approach that leverages the strengths of both techniques. XGBoost is adept at capturing complex patterns, while PSO excels in searching for optimal hyperparameters. This synergy can potentially lead to a more powerful and efficient model for stock price prediction. This study has two following contributions. First is a new method for hyper-parameter tuning of XGBoost. The second is the implementation of the proposed method will give a low error rate for the stock price forecasting.

This paper will be divided into 4 sections. Section 1 is the introduction of the paper that contains the motivation behind the study, problem statement, literature review and the proposed method. Section 2 will be the methodology which covers the object of the study, description on the proposed method which is the improved XGBoost algorithm by using Particle Swarm Optimization method. Section 3 will be the result and discussion. Lastly, Section 4 will be the conclusion and the future works of the study.

2. METHODS

This section will discuss about the XGBoost algorithm, Particle Swarm Optimization (PSO) and the complete algorithm of PSO optimized XGBoost for the stock price forecasting. The research methodology is shown in Fig. 3. The research will be divided into 5 phases. The first phase is the data collection, second is data preparation which includes the data selection and data construction. Next, the third phase will be the development of several machine algorithms for the stock price forecasting. The details of the proposed method, which is the combination of PSO for XGBoost hyperparameters tuning will be explained in detail. The fourth phase is validation. Lastly is the analysis that will be written in a report.

2.1. Data Collection and Data Preparation

The sampling method in this research employs the purposive sampling technique, which is a method of sample selection based on specific considerations. The method was chosen due to the volatility and market conditions. Stock markets can be highly volatile, and different stocks may respond differently to market conditions. Purposive sampling allows researchers to select stocks that are more likely to reflect the desired characteristics or trends, considering the dynamic nature of financial markets. The second reason is the resource constraints. In this study, a large data set was used. To analyze a large number of stocks, it requires substantial computational power and data processing capabilities. Purposive sampling enables a more manageable analysis focused on a select subset of stocks.

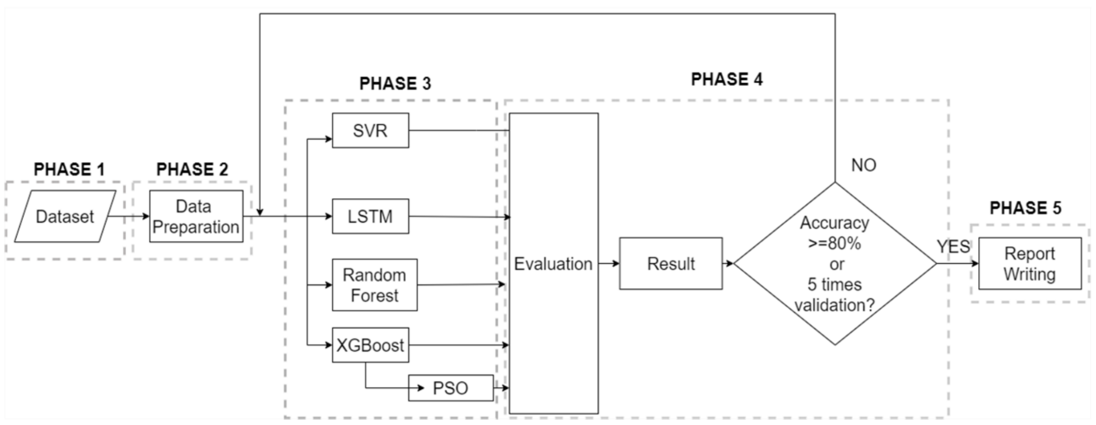


Fig. 3. Research Flowchart

In this study, the population consists of all stock price data available on the Indonesia Stock Exchange (BEI), and the sample comprises data from the Composite Stock Price Index (IHSG) obtained from the Yahoo Finance website for the period of 2012 to 2022, totaling 2,733 rows of data. IHSG was chosen as the sample because it is deemed to provide an indication of the performance of all stock prices on the Indonesia Stock

Exchange. This data can be accessed on the Yahoo Finance website's database, available at https://finance.yahoo.com/quote/^JKSE/history?p=^JKSE. The data can also be downloaded using the Python library, *yfinance*.

The data used in this study is a time-series data. One characteristic of time series data is its relative variability over time. Additionally, this type of data typically involves a single variable (although it can be extended to include multiple variables). Analysis of time series data is based on past values and their influence on the variable. Time series data is highly influenced by the data sequence, which is why analyses using time series data heavily rely on lag or differentiation. Other important characteristics of time series data include stationary, trend, seasonal and cycle. In stationary, the data's variance should be relatively constant throughout the time series. If the variance changes over time, the data is non-stationary. Trends can indicate consistent increases or decreases over time, showing long-term change patterns. Longer-term patterns of change, often associated with medium or long-term economic fluctuations.

The IHSG data consists of 7 features. If the data is directly downloaded from the Yahoo Finance website, it will be in comma-separated value (CSV) format with the total number of data rows is 2,733. Another way to access the data is by using the Python *yfinance* library, resulting in 2,681 rows of data. The difference of 32 rows is due to null values in the *Open*, *High*, *Low*, *Close*, *Adj Close*, and *Volume* columns. The *yfinance* library has an internal mechanism to automatically exclude (skip) null data.

2.1.1. Data Selection

Of all the features available in the dataset, only 2 features will be used for further analysis: the *Date* and *Close* features. The *Date* feature is chosen because advanced features, such as day-of-the-week, will be created from it. The *Close* feature is selected because the closing stock price is considered to represent the last transaction price on a trading day.

2.1.2. Data Cleaning

In datasets, it is common for one or more features to have missing values, denoted as Not a Number (*NaN*). Some machine learning models, when exposed to data with missing values during the learning process, may produce less accurate models. Missing data will be imputed, a method to handle missing data by estimating values based on available information. If all features in a record are empty, the record will be removed.

2.1.3. Data Construction

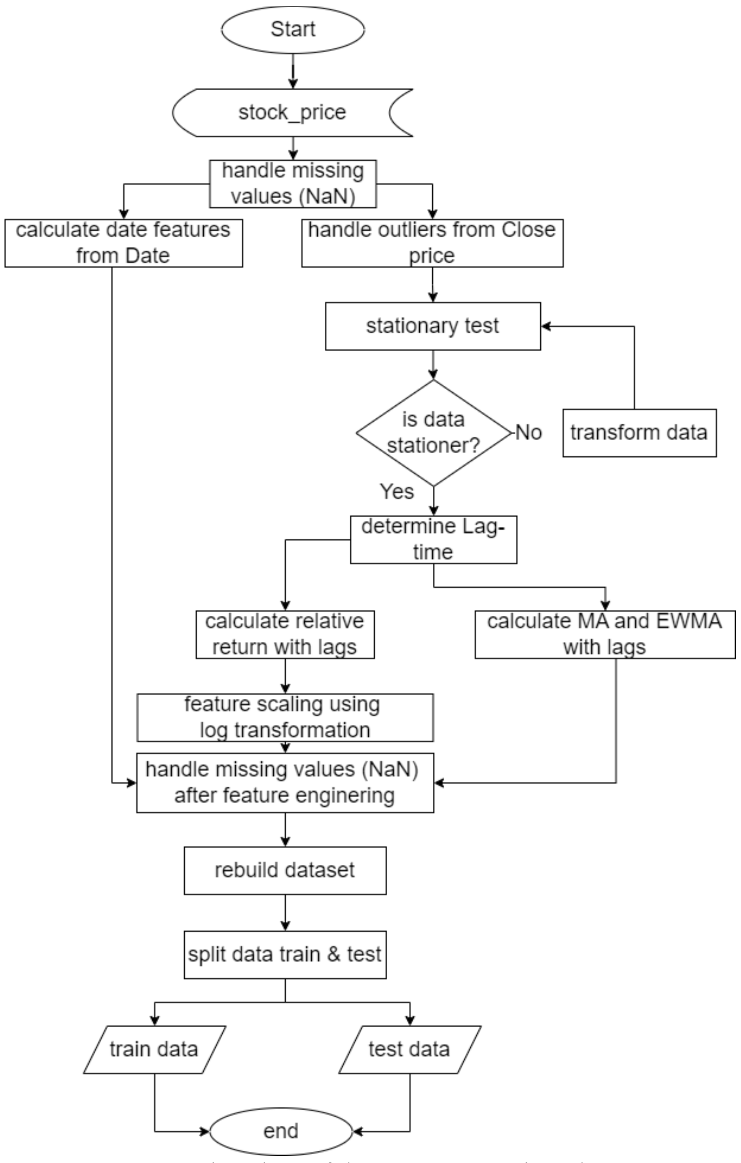
The column details and data types for historical stock data (IHSG) are explained in Table 2. Of these 7 features, the author focuses on the *Date* and *Close* columns, which represent the transaction date and the closing stock price on that date.

Since machine learning is a process that learns from data to generate a model, the compatibility between training data and the algorithm used in machine learning becomes crucial. The flowchart of the data preparation process is shown in Fig. 4. The detailed explanation is as follows.

Tabel 2. Description of IHSG Data Columns

No.	Column	Data Type	Description
1	Date	Date	Date of transaction data on the stock exchange
2	Open	Numeric (float)	The opening stock price on that trading date
3	High	Numeric (float)	The highest stock price reached on that trading date
4	Low	Numeric (float)	The lowest stock price reached on that trading date
5	Close	Numeric (float)	The closing stock price on that trading date
6	Adj Close	Numeric (float)	The adjusted closing stock price based on stock splits, dividend distribution or other factors influencing the stock price.
7	Volume	Numeric (integer)	The number of hares trade on that trading date

A fundamental assumption when analyzing time series data is that it should be in a stationary condition [41], meaning the variance of the data is relatively constant over time. Therefore, a stationarity test will be conducted on the *Close* feature. If the *Close* feature is not stationary, it will be transformed into relative returns. If the data is stationary, relative returns will become the dependent variable. Relative returns are scaled ratios; a value above 1 indicates a capital gain, while a value below 1 indicates a capital loss. For instance, if the relative return is 1.035, it means there is a capital gain of 0.035 (3.5%). Conversely, if the relative return is below 1, it indicates a capital loss. Transforming the *Close* feature into relative returns is a common practice in financial time series analysis for several reasons. The first is the normalization data. Relative returns represent the percentage change in the price from one period to the next, providing a normalized view of price movements. The second is stationary requirements. Many financial time series models assume that the data is stationary, meaning that statistical properties do not change over time. Transforming the *Close* feature into relative returns helps meet this stationarity requirement by capturing the percentage change and reducing trends or seasonality. Last reason is enhanced model performance. Achieving stationarity through the transformation enhances the reliability and accuracy of subsequent analysis and predictions.


Start
stock_price
handle missing values (NaN)
calculate date features from Date
handle outliers from Close price
stationary test
is data stationer?
No
transform data
Yes
determine Lag-time
calculate relative return with lags
calculate MA and EWMA with lags
feature scaling using log transformation
handle missing values (NaN) after feature enginering
rebuild dataset
split data train & test
train data
test data
end
Fig. 4. Flowchart of the Data Preparation Phase

The *Date* feature will be used to create new features such as *day_of_weeks*, *week_of_year*, etc. The relative return feature will be used to create new features using lag relative returns, Moving Average (MA), and Exponential Weighted Moving Average (EWMA) with daily lags of 1, 2, 3, 4, 5, 6, and 7. These new features will become independent variables in this study.

2.2. XGBoost Algorithm

XGBoost is an advanced implementation of the gradient boosting algorithm. XGBoost uses the ensemble principle, which combines several weak learner sets (trees) into a strong model to produce strong predictions. XGBoost has many advantages including, it can perform parallel processing which can speed up computation, has high flexibility of objective setting, built-in cross-validation, and overcomes split during negative loss. With these advantages, XGBoost is very suitable for processing classification data. XGBoost will create a tree to classify train data so that a specific target can be obtained. XGBoost has several parameters that can be set so that it can adjust to the dataset obtained. Parameter tuning in XGBoost can be done using *gridSearchCV* and tuning manually [42].

Parameter tuning is an activity to determine the right parameters to get the XGBoost classifier algorithm that has high accuracy. Parameter tuning is done by changing the parameters given to the XGBoost classifier algorithm. From the given parameters, the prediction accuracy is evaluated. The highest accuracy is when certain given parameters are taken to be the ideal parameters for the XGBoost classifier algorithm.
Parameters in the XGBoost classifier algorithm include:

1. *learning_rate :* The level of learning affects the XGBoost classifier algorithm in making a classifier in the form of a tree.
2. *n*_estimators*:* the number of trees created by the classifier.
3. max*_depth:* the maximum depth of the tree.
4. *min_child_weight:* the minimum amount of weight (hessian) required on child nodes.
5. *Gamma:* the minimum loss required to partition the nodes in the tree.
6. *Subsample:* the number of samples selected to construct the tree. subsampling will be done every boosting iteration.

The selection of the hyper-parameters of the XGBoost is a crucial task since this will give impact on the performance, stability, and efficiency of the model. The appropriate selection of the hyper-parameter will avoid the overfitting and underfitting, increase the computational efficiency, as well as increase the robustness of the model to the data change.

2.3. Particle Swarm Optimization (PSO) Based XGBoost Algorithm

One of the methods for the hyper-parameter tuning of XGBoost is based on swarm intelligence algorithm. Swarm intelligence is a collective behavior exhibited by groups of simple entities (agents or particles) interacting with each other and their environment. One of the methods is Particle Swarm Optimization (PSO).

Particle Swarm Optimization (PSO) is employed for hyperparameter tuning due to its ability to efficiently explore the hyperparameter space in search of optimal values. PSO leverages a swarm of particles, each representing a potential solution in the hyperparameter space. The particles iteratively adjust their positions based on their own experience and the collective knowledge of the swarm, allowing for a dynamic and collaborative search process. This approach is particularly effective for hyperparameter tuning as it can navigate complex, multi-dimensional spaces to find configurations that enhance the performance of machine learning models.

The six parameters of XGBoost as explained in section 2.3 will be optimized by using PSO. The objective in XGBoost is to minimize the value of the objective function in (1).

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{1}$$

where l is the loss, the difference between the estimated value, $\hat{y}_i$, and the actual value, y_i, Ω is used to determine the penalty on the decision tree to avoid overfitting, as expressed in (2).

$$\Omega(f) = \gamma T + \frac{1}{2}\|\omega\|^2 \tag{2}$$

where γ is a hyperparameter that controls the complexity of the model, T is the number of leaf nodes, λ is the penalty coefficient for the leaf weight ω, which is typically set as a constant.

In this stage, the objective function of the original XGBoost will be expanded using Taylor expansion. Thus, (3) will be obtained and used for the optimization process of XGBoost parameters using the PSO algorithm.

$$\tilde{L}^{(t)} = \sum_{i=1}^{n} [g_i f_i(x_i) + \frac{1}{2} h_i {f_i}^2(x_i)] + \Omega(f_k) \tag{3}$$

where g_i is the first derivative, and h_i is the second derivative of the loss function as expressed in (1).

The implementation diagram of PSO on XGBoost is shown in Fig. 5. In the initial step, the PSO parameters that need to be set are the number of *iterations* (n), *cognitive parameter* (c_1), *social parameter* (c_2), and *weight* (*w*). The XGBoost hyperparameters to be optimized include the *learning rate*, *Maximum tree depth*, *Subsample ratio*, *Column subsample ratio*, *Minimum child weight*, *Maximum delta step*, and *Gamma*.

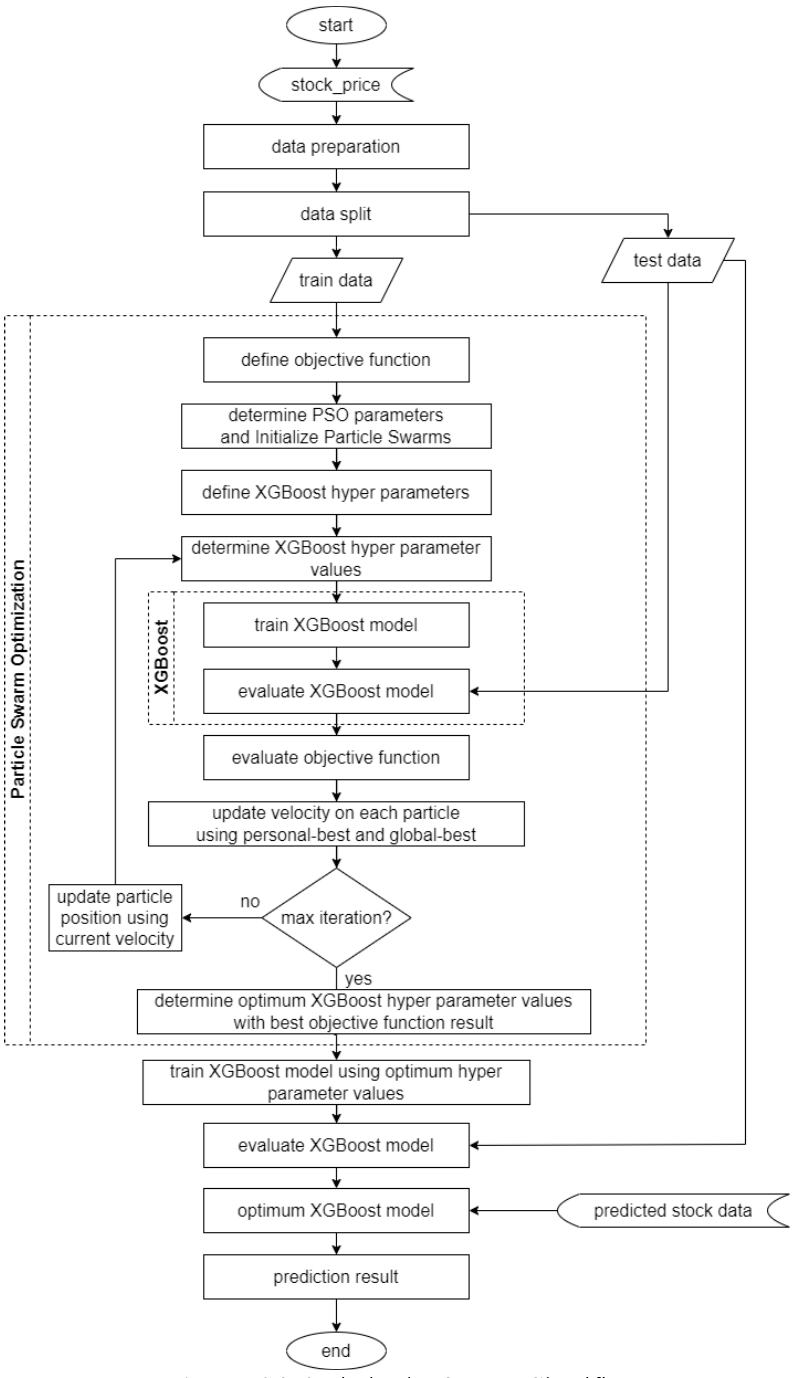

start
stock_price
data preparation
data split
train data
test data
Particle Swarm Optimization
define objective function
determine PSO parameters and Initialize Particle Swarms
define XGBoost hyper parameters
determine XGBoost hyper parameter values
XGBoost
train XGBoost model
evaluate XGBoost model
evaluate objective function
update velocity on each particle using personal-best and global-best
max iteration?
no
update particle position using current velocity
yes
determine optimum XGBoost hyper parameter values with best objective function result
train XGBoost model using optimum hyper parameter values
evaluate XGBoost model
optimum XGBoost model
predicted stock data
prediction result
end

Fig. 5. PSO Optimized XGBoost Classifier

2.4. Analysis Method

The predictive model created using XGBoost optimized with PSO needs to be evaluated and then compared with predictive models built using the XGBoost algorithm without optimization and models created using another algorithm. This comparison aims to draw conclusions regarding the performance of the created predictive model (XGBoost with PSO).

Evaluation metrics to be used include Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R-squared. Stock prices are inherently dynamic and influenced by various factors. RMSE helps quantify the accuracy of predictions in absolute terms, while MAPE provides a percentage-based measure, allowing for an assessment of the relative accuracy of forecasts, which is valuable for investors. R-squared, capturing the proportion of explained variance, ensures that the model's predictions align with the underlying trends in the stock prices, reflecting the model's overall effectiveness in capturing market dynamics.

The alternative algorithms for comparison include Support Vector Regression (SVR), Long Short-Term Memory (LSTM), Random Forest, Gaussian Process and ARIMA.

3. RESULTS AND DISCUSSION

This section will present the result of data acquisition, data pre-processing, and data representation section, followed by the implementation of the proposed method, PSO-optimized XGBoost, and the comparison with the original XGBoost, SVR, LSTM, Random Forest, Gaussian process and ARIMA.

3.1. Data Acquisition and Data Preparation

The proposed method as explained in section 2.1 is implemented to the raw data taken from yahoo finance website. The visualization of stock price data from January 2012 until December 2022 is shown in Fig. 6. As seen in the graph, the data fluctuated a lot. However, after conducting a detailed observation, it is known that some of the data has no value (*null*). In XGBoost, handling *null* or missing data is crucial for obtaining accurate and reliable results. XGBoost is generally robust to missing values, but it is essential to address the *null* data appropriately to prevent potential issue. The approach used to handle *null* data in this study is to ignore or delete the particular data. The approach was chosen for several reasons. First is ignoring missing data simplifies the modeling process and avoids the complexities associated with imputation methods. Second is, after observing the raw data, it is known that there is no systematic pattern or bias in the missingness. Therefore, ignoring those instances is expected for maximum utilization of available data without introducing potential bias.

The next step is to conduct testing for data stationarity. The method employed is the Augmented Dickey-Fuller test with a p-value threshold of 0.05. After performing the stationarity test, a p-value of 0.44 was obtained. Therefore, it is concluded that the *Close* data is not stationary. This circumstance makes it challenging to predict the data accurately. This finding is also supported by visual observations from the line plot in Fig. 6 and further substantiated by the histogram of the *Close* data, revealing two 'peaks'. This process is illustrated in Fig. 7**.**

Since the *Close* data is not stationary, it is necessary to undergo a data transformation process to obtain stationary data. The transformation process involves converting *Close* into *relative return* and subsequently applying a logarithmic transformation to the *relative return*. This is done, among other reasons, to reduce data variance, adjust data scale, improve data distribution, and mitigate the impact of outliers, ensuring optimal learning from the data.

Next, a stationarity test is conducted on the log-transformed relative return (LogRR), yielding a p-value of 0.0. It is thus concluded that the LogRR data is stationary and ready for use in the subsequent stages. This feature will then be employed as the dependent variable. The result is shown in Fig. 8.

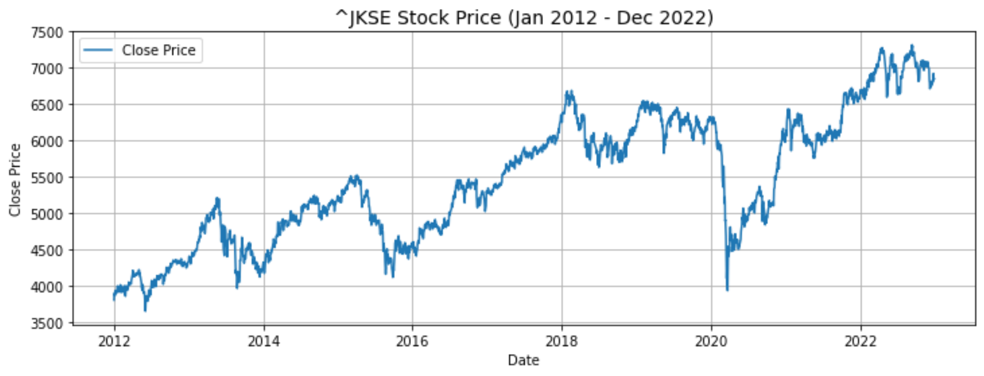


Fig. 6. Line Plot Visualization of IHSG Data (*Close*)

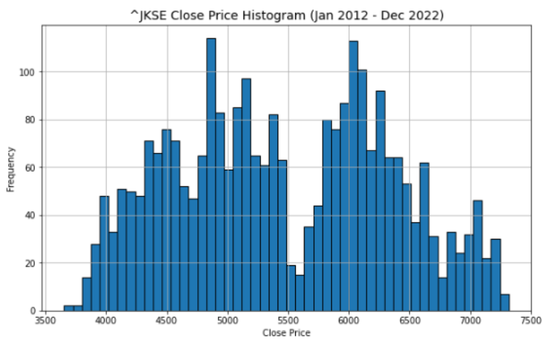


Fig. 7. Histogram Visualization of IHSG Data (*Close*)

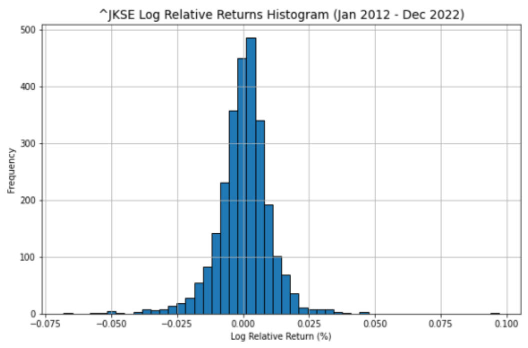


Fig. 8. Histogram Visualization for Log Relative Return

3.2. Performance Comparison

In this section, the performance of the proposed method will be analyzed by observing the model evaluation metrics in terms of Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R-squared. These metrics provide a comprehensive assessment on the predictive capabilities of the proposed PSO-optimized XGBoost algorithm.

3.2.1. Performance Comparison with Standard XGBoost

The performance of the proposed system, PSO-optimized XGBoost is compared to the standard XGBoost. The objective function used in the optimization process is the value of RMSE of the XGBoost model. PSO parameters namely the number of agents, *n* and the maximum iteration, *mi* will be adjusted to get the best performance. Three different values of number of agents, n and three different values of maximum iteration are evaluated. The hyper-parameter of standard XGBoost and the ones obtained from the optimization process by using PSO with different setting of number of agents and maximum iteration is summarized in Table 3.

By looking at the result in Table 3, number of agents and maximum iteration will not have any effect to the number of estimators, *n_estimators, gamma* and *col_sample_bytree*. The highest value of *n_estimators* will improve the model's performance. However, it can also lead to overfitting if the values are too high. *gamma* parameters give 0 for all combinations number of agents, *n* and maximum iteration, *mi*. *gamma* controls whether a given node will split based on the expected reduction in loss after the split or not. Generally, *gamma* is a regularization term that penalizes the creation of new nodes in the three. In our case study, it is clearly seen that, for all the combination of *n* and *mi*, the model does not need the creation of new nodes in the three, which is the same as the standard XGBoost algorithm.

The obvious difference is seen on the parameter *max_depth*, and *alpha*. When the number of agents increases, the *max_depth* parameter also increasing. *max_depth* controls the maximum depth of each tree in the ensemble. It limits the number of nodes in a tree. A higher *max_depth* allows the tree to capture more complex relationships within the training data but increases the risk of overfitting. On the other hand, low *max_depth* number can lead to simple trees and less prone to overfitting. However, it might not capture complex patterns in the data. By looking at the result from different value of number of agents, it is seen that almost all the combination gave value of 50 for the *max_depth*, except the standard XGBoost, combination ***C1***, and ***C4***.

Another hyper-parameter to be concerned with is *alpha*. alpha is related to L1 regularization, encouraging sparsity to the feature weights. When a model is sparse, it indicates that only a subset of features is contributing to the predictions, while the others have effectively been deemed less important or irrelevant. From the result it is seen that three combinations which are ***C1***, ***C3***, and ***C4***. The effect of alpha on the performance of the system will be analyzed in detail after this.

Table 4 shows the performance comparison between standard XGBoost and PSO-optimized XGBoost. The prediction graph is shown in Fig. 9. By considering the performance metrics in terms of RMSE, MAE, MAPE and R^2, overall, the PSO-optimized XGBoost with the PSO parameters combination of ***C2*** is on par with combination of ***C5***. The results show that both combinations gave 0.0007 of Root Mean Squared Error (RMSE), 0.0005 of Mean Absolute Error (MAE), about 0.05 % of Mean Absolute Percentage Error (MAPE) and 0.99 of R^2. From the result, it can be concluded that these two combinations have a very accurate performance in prediction task and very low error. It is proven with the result shown in Fig. 9.

Table 4 also shows the effect of number of agents and maximum iteration to the performance of the models. Combination ***C1***, ***C2*** and ***C3*** were conducted to see the effect of number of agents when the maximum iteration is maintained the same. From the result, it is seen that there is no significant effect on the number of agents to the RMSE, MAE, and R^2. Meanwhile, combination ***C4***, ***C5*** and ***C6*** were conducted to see the effect of maximum iteration to the performance of the proposed method. It is seen that when the maximum iteration is increased, the RMSE, MAE, MAPE and R2 also improve. However, during the validation process, the maximum iteration of 100 took a longer time and the performance is not much different compared to maximum iteration of 50.

In terms of the effect of hyper-parameter *alpha* to the model performance, it is clearly seen from Table 4 that for combination that has high alpha value, the MAPE value is also high. This indicates that the prediction result has higher error compared to the expected one. Hence, the model is less accurate. This is true since increasing the value of *alpha* encourages a stronger penalty on the absolute values of the coefficients, making it more likely that some coefficients will be exactly zero. This, in turn, leads to a sparser model, which some features tend to be ignored in the model.

Tabel 3. Hyper-parameters values for Standard XGBoost

Hyper-parameter	**Standard XGBoost**	**PSO Optimized XGBoost**					
		C1	***C2***	***C3***	***C4***	***C5***	***C6***
		n = 20 *mi = 25*	*n = 50* *mi = 25*	*n = 100* *mi = 25*	*n = 20* *mi = 10*	*n = 20* *mi = 50*	*n = 20* *mi = 100*
max_depth	6	5	50	49.26	44.07	50	12.89
learning_rate	0.3	0.15	0.08	0.13	0.3	0.10	0.11
gamma	0	0	0	0	0	0	0
colsample_bytree	1	1	1	1	0	1	1
subsample	1	0.54	0.30	1	1	0.37	0.85
alpha	0	5.48	0.26	8.85	10	0	0
lambda	1	0	0	0	0.28	0.79	1.17
min_child_weight	1	1	5.24	1	10	1.32	9.9
n_estimators	20	100	100	100	100	100	99.08

*** *n* is number of PSO agents and *mi* is the maximum iteration, *Ci* is combination *#i***

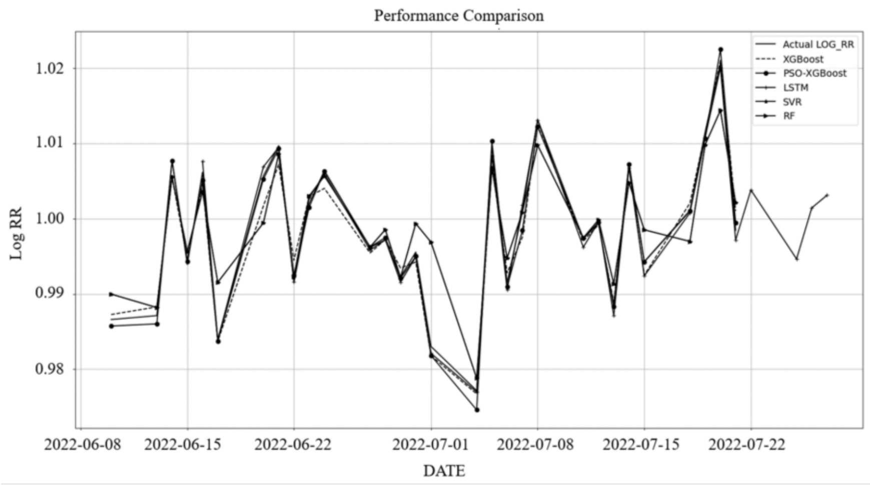


Fig. 9. Prediction Line of Standard XGBoost, PSO-Optimized XGBoost and Benchmark Algorithm

Tabel 4. Performance Evaluation of Standard XGBoost and PSO-Optimized XGBoost

Metric	**Standard XGBoost**	**PSO Optimized XGBoost**					
		C1	***C2***	***C3***	***C4***	***C5***	***C6***
		n = 20 *mi = 25*	*n = 50* *mi = 25*	*n = 100* *mi = 25*	*n = 20* *mi = 10*	*n = 20* *mi = 50*	*n = 20* *mi = 100*
RMSE	0.0017	0.0008	**0.0007**	0.0008	0.0044	**0.0007**	0.0008
MAE	0.0013	0.0006	**0.0005**	0.0006	0.0033	**0.0005**	**0.0005**
MAPE (%)	0.1294	0.0586	0.0496	0.0572	0.3321	**0.0478**	0.0507
R^2	0.9571	0.9900	**0.9934**	0.9895	0.7089	0.9933	0.9907

*** bold number is the best performance**

3.2.2. Comparison with Benchmark Algorithms

To further assess the effectiveness of the proposed approach, we compared the PSO-optimized XGBoost with benchmark algorithms, including Support Vector Regression (SVR) [43], Long Short-Term Memory (LSTM) [44], Random Forest , Gaussian Process [45], and ARIMA [46]. In this section, combination of ***C2*** where the number of agents in PSO is 50, and the maximum of iteration is 25, will be used for the comparison. The reason is because it gives the best performance metrics compared to the other combinations. Table 5 shows the hyper-parameter setting for the three benchmark models. The parameter setting for Random Forest and Gaussian Process are the default setting in scikit-sklearn library. ARIMA parameter is chosen by following the authors in [46].

Table 6 summarizes the comparative results. As seen in the table, LSTM has the best performance compared to the other method. However, the performance of the proposed method is not far from LSTM. Gaussian process and ARIMA have the lower performance compared to the proposed method. It can be concluded that the proposed method also has a promising performance and will be suitable for the application of stock price forecasting.

Tabel 5. Parameter Setting for Benchmark Models

SVR	**LSTM**	**Random Forest**	**Gaussian Process**	**ARIMA**
Regularization parameter, $c = 10$	Number of hidden layer = *64*	n_estimators = *100*	Optimizer : Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS-B)	Model : ARIMA (4,1,3)
Tube radius, $\varepsilon = 0.01$	Activation function = *'relu'*	Criterion = *'mse'*	Kernel : Radial Basis Function (RBF)	Method: css-mle
gamma = 0.3	Optimizer = *'adam'*	Min_samples_split = *2*	Normalization : ACTIVATED	AIC: 71180.553
	Loss function = *'mse'*	Min_samples_leaf = *1*		BIC: 71243.107
	Epoch = *50*	Bootstrap = *'True'*		HQIC: 71202.003
	Batch size = *32*			
	Validation split = *0.1*			
	Early stopping = *10 epoch*			

Tabel 6. Performance Evaluation of PSO-Optimized XGBoost and Benchmark Models

Metric	**PSO Optimized XGBoost**	**SVR**	**LSTM**	**Random Forest**	**Gaussian Process**	**ARIMA**
RMSE	0.0007	0.0025	**0.0004**	0.0016	1.9765	NaN
MAE	0.0005	0.0018	**0.0003**	0.0009	3.5019	29.6975
MAPE (%)	0.0496	0.1806	**0.0276**	0.0993	0.1191	0.4672
R^2	0.9934	0.9111	**0.9979**	0.9618	NaN	NaN

Bold number is the best performance.

In machine learning, another performance that should be concerned with is the consistency of the model. Empirical evaluation is conducted to see the performance of PSO-optimized XGBoost and LSTM for 5 times of running process. This analysis has the purpose of observing the robustness and consistency of the algorithm across different runs. LSTM will start to set the initial weight and bias randomly. While, in PSO, the position of agents at the beginning of the process will also be random. Through the optimization process, both algorithms will try to reach their convergence value. The consistent convergence value is preferable. Table 7 shows the result from 5 time running for both algorithms. Statistical metric average and standard deviation are used to examine the performance of both algorithms. It is clearly seen that PSO-optimized XGBoost has better and more consistent results compared to LSTM.
In terms of RMSE, MAE and MAPE, all the values of PSO-optimized XGBoost are lower than LSTM. This suggests that the proposed method is better than LSTM.

Tabel 7. Empirical Evaluation of PSO-Optimized XGBoost and LSTM

Metric		PSO Optimized XGBoost	LSTM
RMSE	Averange	**0.0011**	0.0039
	Standard deviation	**0.0003**	0.0029
MAE	Averange	**0.0008**	0.0031
	Standard deviation	**0.0002**	0.0023
MAPE (%)	Averange	**0.0772**	0.3076
	Standard deviation	**0.0183**	0.2312
R^2	Averange	**0.9799**	0.6359
	Standard deviation	**0.01**	0.3058

Bold number is the best performance

Meanwhile, the standard deviation obtained shows that PSO-optimized XGBoost has lower standard deviation, which means that the proposed algorithm is more consistent compared to LSTM.

The improvement in predictive accuracy, as evidenced by lower RMSE and MAPE values, underscores the impact of PSO optimization on refining the XGBoost model. This enhancement is particularly crucial in stock price forecasting.

4. CONCLUSION

This study proposed a new approach for the optimization of XGBoost algorithm for stock price forecasting. The proposed method gives a promising result after comparing it with standard XGBoost, Random Forest, Long-Short Term Memory (LSTM), Support Vector Regression (SVR), Gaussian Process and ARIMA. The proposed method achieved 0.0011 of Root Mean Squared Error (RMSE), 0.0008 of Mean Average Error (MAE), 0.0772% of Mean Absolute Percentage Error (MAPE) and 0.9799 of R^2.

In conclusion, the proposed PSO-optimized XGBoost algorithm is demonstrating its superior performance compared to the standard XGBoost and benchmark algorithms. The results underscore the potential of integrating PSO for optimizing hyperparameters in machine learning models, offering a promising avenue for further research and practical applications. This is proven by the fact that the proposed method, PSO-optimizaed XGBoost can improve the performance of standard XGBoost by 60%, 61%, 63% and 4% in terms of RMSE, MAE, MAPE and R^2, respectively.

While the results are promising, it is essential to acknowledge certain limitations. Inconsistency of the prediction result may appear when using the proposed algorithm. In the future, the proposed algorithm will be used for stock price forecasting for different data sets from other countries, for example Tokyo Stock Exchange, New York Stock Exchange, Shanghai Stock Exchange, etc. Implementing the proposed method with diverse data sets is expected to result in the development of a more robust and reliable prediction model.

Acknowledgments

This research is fully sponsored by the Indonesian Ministry of Higher Education, Research and Technology under the Fundamental Research Grant Scheme, Master Thesis Research, 179/E5/PG.02.00/PL/2023, 1404/LL3/AL.04/2023.

REFERENCES

[1] E. F. Fama, "Random Walks in Stock Market Prices," *Financial Analysts Journal*, vol. 21, no. 5, pp. 55–59, 1965, [Online]. Available: http://www.jstor.org/stable/4469865.

[2] D. Bhuriya, G. Kaushal, A. Sharma, and U. Singh, "Stock Market Prediction Using A Linear Regression," in *International Conference on Electronics, Communication and Aerospace Technology*, pp. 510–513, 2017, https://doi.org/10.1109/ICECA.2017.8212716.

[3] N. Vinay and R. Mahaveerakannan, "Analyze the Lack of Accuracy in Stock Price Prediction using Novel K-Nearest Neighbors Regression Compared with Logistic Regression to Improve Accuracy," in *Proceedings of 8th IEEE International Conference on Science, Technology, Engineering and Mathematics, ICONSTEM,* pp. 1-5, 2023, https://doi.org/10.1109/ICONSTEM56934.2023.10142304.

[4] K. M. Hindrayani, T. M. Fahrudin, R. Prismahardi Aji, and E. M. Safitri, "Indonesian Stock Price Prediction including Covid19 Era Using Decision Tree Regression," in *3rd International Seminar on Research of Information Technology and Intelligent Systems, ISRITI,* pp. 344–347, 2020, https://doi.org/10.1109/ISRITI51436.2020.9315484.

[5] P. C. S. Bezerra and P. H. M. Albuquerque, "Volatility forecasting via SVR–GARCH with mixture of Gaussian kernels," *Computational Management Science*, vol. 14, no. 2, pp. 179–196, 2017, https://doi.org/10.1007/s10287-016-0267-0.

[6] Z. Liu, Z. Dang, and J. Yu, "Stock price prediction model based on RBF-SVM algorithm," in *Proceedings - International Conference on Computer Engineering and Intelligent Control, ICCEIC*, pp. 124–127, 2020, https://doi.org/10.1109/ICCEIC51584.2020.00032.
[7] G. Bathla, "Stock price prediction using LSTM and SVR," in *PDGC 6th International Conference on Parallel, Distributed and Grid Computing*, pp. 211–214, 2020, https://doi.org/10.1109/PDGC50313.2020.9315800.
[8] E. Edward and S. Suharjito, "Tesla Stock Close Price Prediction Using KNNR, DTR, SVR, and RFR," *Journal of Human, Earth, and Future*, vol. 3, no. 4, 2022, [Online]. Available: https://www.hefjournal.org/index.php/HEF/article/view/214/90.
[9] J. Zhang, Y.-F. Teng, and W. Chen, "Support vector regression with modified firefly algorithm for stock price forecasting," *Applied Intelligence*, vol. 49, no. 5, pp. 1658–1674, 2019, https://doi.org/10.1007/s10489-018-1351-7.
[10] E. Ahmadi, M. Jasemi, L. Monplaisir, M. A. Nabavi, A. Mahmoodi, and P. A. Jam, "New efficient hybrid candlestick technical analysis model for stock market timing on the basis of the Support Vector Machine and Heuristic Algorithms of Imperialist Competition and Genetic," *Expert Syst Appl*, vol. 94, pp. 21–31, 2018, https://doi.org/10.1016/j.eswa.2017.10.023.
[11] Y. Chen and Y. Hao, "Integrating principle component analysis and weighted support vector machine for stock trading signals prediction," *Neurocomputing*, vol. 321, pp. 381–402, 2018, https://doi.org/10.1016/j.neucom.2018.08.077.
[12] S. P. Das and S. Padhy, "A novel hybrid model using teaching–learning-based optimization and a support vector machine for commodity futures index forecasting," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 1, pp. 97–111, 2018, https://doi.org/10.1007/s13042-015-0359-0.
[13] M. M. Gowthul Alam and S. Baulkani, "Local and global characteristics-based kernel hybridization to increase optimal support vector machine performance for stock market prediction," *Knowl Inf Syst*, vol. 60, no. 2, pp. 971–1000, 2019, https://doi.org/10.1007/s10115-018-1263-1.
[14] M. Sedighi, H. Jahangirnia, M. Gharakhani, and S. F. Fard, "A novel hybrid model for stock price forecasting based on metaheuristics and support vector machine," *Data (Basel)*, vol. 4, no. 2, pp. 1–28, 2019, https://doi.org/10.3390/data4020075.
[15] D. Kumar, P. K. Sarangi, and R. Verma, "A systematic review of stock market prediction using machine learning and statistical techniques," in *Materials Today: Proceedings*, pp. 3187–3191, 2020, https://doi.org/10.1016/j.matpr.2020.11.399.
[16] A. W. Li and G. S. Bastos, "Stock market forecasting using deep learning and technical analysis: A systematic review," *IEEE Access*, vol. 8, pp. 185232–185242, 2020. https://doi.org/10.1109/ACCESS.2020.3030226.
[17] M. Qiu and Y. Song, "Predicting the direction of stock market index movement using an optimized artificial neural network model," *PLoS One*, vol. 11, no. 5, 2016, https://doi.org/10.1371/journal.pone.0155133.
[18] X. Pang, Y. Zhou, P. Wang, W. Lin, and V. Chang, "An innovative neural network approach for stock market prediction," *Journal of Supercomputing*, vol. 76, pp. 2098–2118, 2018, https://doi.org/10.1007/s11227-017-2228-y.
[19] D. Noel, "Stock Price Prediction using Dynamic Neural Networks," *Computational Engineering*, 2023, [Online]. Available: https://www.researchgate.net/publication/371786532.
[20] R. Pimprikar, S. Ramachandran, and K. Senthilkumar, "Use Of Machine Learning Algorithms And Twitter Sentiment Analysis For Stock Market Prediction," *International Journal of Pure and Applied Mathematics*, vol. 115, no. 6, pp. 521–526, 2017, [Online]. Available: http://www.ijpam.eu.
[21] N. Maknickiene, I. Lapinskaite, and A. Maknickas, "Application of ensemble of recurrent neural networks for forecasting of stock market sentiments," *Quarterly Journal of Economics and Economic Policy*, vol. 13, no. 1, pp. 7–27, 2018, https://doi.org/10.24136/eq.
[22] S. W. Lee and H. Y. Kim, "Stock market forecasting with super-high dimensional time-series data using ConvLSTM, trend sampling, and specialized data augmentation," *Expert Syst Appl*, vol. 161, p. 113704, 2020, https://doi.org/https://doi.org/10.1016/j.eswa.2020.113704.
[23] W. Lu, J. Li, Y. Li, A. Sun, and J. Wang, "A CNN-LSTM-based model to forecast stock prices," *Complexity*, pp. 1-10, 2020, https://doi.org/10.1155/2020/6622927.
[24] H. Chung and K. S. Shin, "Genetic algorithm-optimized long short-term memory network for stock market prediction," *Sustainability (Switzerland)*, vol. 10, no. 3765, pp. 1–18, 2018, https://doi.org/10.3390/su10103765.
[25] W. Zhang, S. Zhang, S. Zhang, D. Yu, and N. Huang, "A novel method based on FTS with both GA-FCM and multifactor BPNN for stock forecasting," *Soft comput*, vol. 23, no. 16, pp. 6979–6994, 2019, https://doi.org/10.1007/s00500-018-3335-2.
[26] C. Chalvatzis and D. Hristu-Varsakelis, "High-performance stock index trading via neural networks and trees," *Applied Soft Computing Journal*, vol. 96, 2020. https://doi.org/10.1016/j.asoc.2020.106567.
[27] S. Basak, S. Kar, S. Saha, L. Khaidem, and S. R. Dey, "Predicting the direction of stock market prices using tree-based classifiers," *The North American Journal of Economics and Finance*, vol. 47, pp. 552–567, 2019, https://doi.org/10.1016/j.najef.2018.06.013.

[28] K. K. Yun, S. W. Yoon, and D. Won, "Prediction of stock price direction using a hybrid GA-XGBoost algorithm with a three-stage feature engineering process," *Expert Syst Appl*, vol. 186, 2021, https://doi.org/10.1016/j.eswa.2021.115716.

[29] E. Zolotareva, "Aiding Long-Term Investment Decisions with XGBoost Machine Learning Model," in *International Conference on Artificial Intelligence and Soft Computing, Lecture Notes in Computer Science*, pp. 414–427, 2021, https://doi.org/10.1007/978-3-030-87897-9_37.

[30] M. Biswas, A. Shome, M. A. Islam, A. J. Nova, and S. Ahmed, "Predicting stock market price: A logical strategy using deep learning," in *ISCAIE IEEE 11th Symposium on Computer Applications and Industrial Electronics*, pp. 218–223, 2021, https://doi.org/10.1109/ISCAIE51753.2021.9431817.

[31] J. Huang, J. Chai, and S. Cho, "Deep learning in finance and banking: A literature review and classification," *Frontiers of Business Research in China*, vol. 14, no. 1, 2020. https://doi.org/10.1186/s11782-020-00082-6.

[32] M. Nabipour, P. Nayyeri, H. Jabani, A. Mosavi, E. Salwana, and S. Shahab, "Deep learning for stock market prediction," *Entropy*, vol. 22, no. 8, 2020, https://doi.org/10.3390/E22080840.

[33] E. Chong, C. Han, and F. C. Park, "Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies," *Expert Syst Appl*, vol. 83, pp. 187–205, 2017, https://doi.org/10.1016/j.eswa.2017.04.030.

[34] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *Eur J Oper Res*, vol. 270, no. 2, pp. 654–669, 2018, https://doi.org/10.1016/j.ejor.2017.11.054.

[35] M. Jiang, J. Liu, L. Zhang, and C. Liu, "An improved Stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms," *Physica A: Statistical Mechanics and its Applications*, vol. 541, p. 122272, 2020, https://doi.org/https://doi.org/10.1016/j.physa.2019.122272.

[41] D. N. Gujarati, *Basic_Econometrics*. McGraw-Hill, 2009, https://thuvienso.hoasen.edu.vn/handle/123456789/8914.

[42] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. 2016, https://doi.org/10.1145/2939672.2939785.

[43] A. Virigineni, M. Tanuj, A. Mani, and S. R, "Stock Forecasting using HMM and SVR," in *International Conference on Communication, Computing and Internet of Things (IC3IoT)*, pp. 1–7, 2022, https://doi.org/10.1109/IC3IOT53935.2022.9767969.

[44] D. Sharma and D. Kaur, "Stock Price Prediction Using Long Short Term Memory Algorithm," in *4th IEEE Global Conference for Advancement in Technology (GCAT)*, pp. 1–5, 2023, https://doi.org/10.1109/GCAT59970.2023.103533821.

[45] J. H. Moedjahedy, R. Rotikan, W. F. Roshandi, and J. Y. Mambu, "Stock Price Forecasting on Telecommunication Sector Companies in Indonesia Stock Exchange Using Machine Learning Algorithms," in *2nd International Conference on Cybernetics and Intelligent System (ICORIS)*, pp. 1–4, 2020, https://doi.org/10.1109/ICORIS50180.2020.9320758.

[46] G. W. R. I. Wijesinghe and R. M. K. T. Rathnayaka, "Stock Market Price Forecasting using ARIMA vs ANN; A Case study from CSE," in *2nd International Conference on Advancements in Computing (ICAC)*, pp. 269–274, 2020, https://doi.org/10.1109/ICAC51239.2020.9357288.

BIOGRAPHY OF AUTHORS

Dwi Pebrianti is Senior Lecturer in Universitas Budi Luhur, Jakarta, Indonesia in the field of Computer Science and Information Technology. Previously, she was a senior lecturer in Universiti Malaysia Pahang, Malaysia in the field of Electrical and Electronics Engineering since 2013 until 2022. She received Philosophy Doctor in Artificial System Science from Chiba University, JAPAN in 2011, Master of Engineering in Precision Engineering from The University of Tokyo, JAPAN in 2006 and Bachelor of Engineering in Electronics from Universitas Indonesia, INDONESIA in 2001. Her main interests are including Nonlinear control & robotics, Unmanned Aerial Vehicle, underactuated mobile robot, Vision based robot navigation, Motion & dynamics control, Swarm robot control, Optimization technique, Machine Learning and Artificial Intelligence. She was the Principal Investigator of 11 research grants from the Japan, Malaysian and Indonesian governments. Additionally, she is the Co-Researcher for 20 research grants. She can be contacted at dwi.pebrianti@budiluhur.ac.id, dwipebrianti@iium.edu.my, https://orcid.org/0000-0001-9938-5219.

Haris Kurniawan is a Master's Student in Universitas Budi Luhur, Jakarta, Indonesia in the field of Computer Science and Information Technology. He received his 3rd degree diploma from Sekolah Tinggi Akuntansi Negara in 2004, later on in 2011 he received Bachelor of Economics with a major in Finance from Universitas Mercu Buana, Jakarta, Indonesia. Eventhough he came from an economic academic background but computer science has always been his passion. He is currently working as Software Developer in Directorate General Taxes of The Republic of Indonesia. His current research is on the predicting and forecasting of time series data using Machine Learning. He can be contacted at haris.kurniawan@gmail.com.

Luhur Bayuaji received the B. Eng degree in Electrical Engineering majoring Computer Engineering in 2001 from Universitas Indonesia, Indonesia and M. Eng. and Ph.D both from Chiba univerity, Japan in 2006 and 2010. His research interest includes Computer Networks, Internet of Things (IoT), Cybersecurity, Image Processing, Remote Sensing and Geographical Information System (GIS). He is currently an Assistant Professor in Faculty of Computing, Universiti Malaysia Pahang, since 2013. Additionally, he is a MASTER TRAINER for CISCO NETWORKING ACADEMY, who is Responsible for teaching and give training Cisco Networking Academy (NetAcad) instructor candidate and student for the following courses: Cisco Certified Network Associate (CCNA), CyberOps Associate, DevNet Associate, Network Security, Cisco Certified Network Professional – Enterprise, Python Programming, and Network Essential and Cybersecurity Essential. He can be emailed at luhurbayuaji@gmail.com, luhur.bayuaji@budiluhur.ac.id, https://orcid.org/0000-0001-8128-3302.

Rusdah is a Head of Computer Science Department in Faculty of Information Technology Universitas Budi Luhur, Jakarta, Indonesia. She is a Senior Lecturer in the field of Computer Science and Information Systems as well. She received Doctor in Computer Science in 2018 from Universitas Gadjah Mada, Yogyakarta, Indonesia, Master of Computer Science in 2006 and Bachelor of Information System in 2001 from Universitas Budi Luhur. Her main interests are Data Mining, Machine Learning, Decision Support System and Information Systems. She is the co-researcher for two research grants from Indonesian government. She can be contacted at rusdah@budiluhur.ac.id, https://orcid.org/0000-0002-1475-3789.