Applied Soft Computing 145 (2023) 110566

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Forecasting long-term stock prices of global indices: A forward-validating Genetic Algorithm optimization approach for Support Vector Regression

Mohit Beniwal [a,*], Archana Singh [a], Nand Kumar [b]

[a] Delhi School of Management, Delhi Technological University, Delhi 110042, India
[b] Department of Humanities, Delhi Technological University, Delhi 110042, India

ARTICLE INFO

Article history:
Received 14 March 2023
Received in revised form 2 May 2023
Accepted 21 June 2023
Available online 26 June 2023

Keywords:
Long-term forecasting
Genetic Algorithm
Support Vector Regression
Stock market prediction
Forward validation
Multi-step prediction
Hyperparameters optimization
LSTM

ABSTRACT

Predicting long-term stock index prices is a challenging and debatable task. Most of the studies focus on predicting next-day stock prices. However, those are not useful to long-term investors and traders. In this paper, we attempt to predict up to a year's daily prices of global stock indices using daily close prices data. This study fills a gap in the existing literature by focusing on long-term stock index price forecasting, which is crucial for practical applications in the financial markets. Moreover, The empirical analysis highlights the superior performance of a rolling forward-validation approach over cross-validation in predicting long-term stock prices. A forward-validating Genetic Algorithm Optimization for Support Vector Regression (OGA-SVR) is used to efficiently forecast multi-step ahead long-term global stock indices. Further, the performance of the model is compared with that of Support Vector Regression (SVR), Grid Search based Support Vector Regression (GS-SVR), Genetic algorithm-based Support Vector regression (GA-SVM), and state-of-the-art Long Short-Term Memory (LSTM) algorithms. The models are empirically tested on five global stock indices time series daily data, namely Nifty, Dow Jones Industrial Average (DJIA), DAX performance index (DAX), Nikkei 225 (NI225), and Shanghai Stock Exchange composite index (SSE). Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) are used for the evaluation The result shows the OGA-SVR model outperforms other models in predicting the long-term prices of global indices. Further, the OGA-SVR model has the potential to forecast the long-term underlying future pattern of index prices which can be used to build trading and risk mitigation systems by investors and traders.

1. Introduction

The stock markets are considered a barometer of the economy of a country. Investing and trading in the stock market are risky as they are sensitive to many factors, such as general global and national economic conditions, geopolitical events, financial news, and trader sentiments. Still, the stock market attracts many investors and traders because of its ease of accessibility, the possibility of diversification, and the potential to give lucrative returns. The efficient market hypothesis [1] is well accepted in academia, which states that stock markets are always correctly priced, and it is impossible to outperform the stock market returns using publicly available information. In other words, it is futile to predict the stock market. Further, the random walk hypothesis [2] supports the efficient market hypothesis that the stock market may not be predictable. Some studies refute these claims and support predictability [3–6]. Benjamin Graham once said that a stock market is a voting machine in the short run, but in the long run, it is a weighing machine [7]. The stock index prices are also biased toward an upward direction in the long term.

Numerous approaches are available for predicting the stock market, including fundamental analysis, technical analysis, statistical analysis, and machine learning (ML). Fundamental analysis is a method to analyze data from past and current financial statements to recognize the firm whose value differs from its stock's price [8]. A fundamental analyst attempts to find undervalued firms to invest in, assuming that the stock markets are not perfectly efficient. In technical analysis, the prediction of the stock market relies on the interpretation of historical market prices, technical indicators, and charts [9]. Statistical analysis uses econometric methods such as Auto-Regressive Integrated Moving Average (ARIMA) to analyze time series stock data and gain insight into the stock market. The statistical method encounters several challenges when the future distribution of observations

* Corresponding author.
E-mail addresses: mohitbeniwal@dtu.ac.in (M. Beniwal), archanasingh@dtu.ac.in (A. Singh), nandkumar@dce.ac.in (N. Kumar).

https://doi.org/10.1016/j.asoc.2023.110566
1568-4946/© 2023 Elsevier B.V. All rights reserved.

is unknown [10–12]. With the advent of electronic trading and an explosion in stock data, researchers have explored Artificial Intelligence (AI) to predict the stock market. Machine learning (ML), a subset of AI, is appropriate for handling non-linear, noisy, large, and complex data. Common ML methods include Support Vector Machine/Regression (SVM/SVR), Random Forest (RF), and Artificial Neural Networks (ANNs). Deep learning (DL), a subset of ML, can learn complex patterns from large data sets. Examples of DL models are Convolution Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM). LSTM and SVM/SVR are the most commonly used ML algorithms for forecasting financial time series [13]. This study experiments with Support Vector Regression (SVR) and LSTM algorithms to predict the stock indices price.

Optimizing the hyperparameters to achieve the best possible model architecture is often necessary to implement machine learning algorithms successfully. However, refining hyperparameters through a trial-and-error approach can be time-consuming for a human ML expert. An alternative approach is to use Automated Machine Learning (AutoML), which can automate several phases of ML, such as data preprocessing, feature selection, and hyperparameter optimization. Hyperparameter optimization is a subfield of automated machine learning (AutoML) that automates the selection of hyperparameters for ML algorithms, leading to the creation of optimized models that are tailored to specific datasets and algorithms [14,15]. The common approaches for automated hyperparameter optimization include Grid-Search (GS), Random Search, and Genetic Algorithms (GA) [14]. Grid search performs an exhaustive search with a predetermined set of hyperparameters to find the optimum combination of hyperparameters for a given model [16]. A GA, on the other hand, is a heuristic optimization technique that emulates natural selection and evolution to find the optimum hyperparameters [17]. SVR has two important hyperparameters, C and epsilon, that control the trade-off between model complexity and error. Achieving high prediction accuracy using a single machine-learning method can be challenging, and combining multiple methods can improve accuracy [18]. In this study, grid-search and genetic algorithms are utilized to optimize these two hyperparameters of the SVR model.

Cross-validation is a commonly used technique for estimating the performance of a model on out-of-sample data, and it is widely used for both model selection and assessment [19]. One of the initial validations proposed by [20,21] is the leave-one-out cross-validation. Out of several extensions of leave-one-out cross-validation, k-fold cross-validation [22] is the most common [19]. K-fold cross-validation randomly splits data in k equally size subset. The model is trained on the 'k−1' subset and tested on the remaining subset. This method assumes that observations are independent and identically distributed. However, in time series analysis, this assumption does not hold. An alternative is forward validation. Schnaubelt [19] empirically studied common validation schemes and concluded that forward validation provides more accurate estimates. In forward validation, data is divided into consecutive intervals rather than being randomly split. The model is trained on earlier intervals and tested on later intervals, making it especially helpful for time series data. Therefore, this study's optimized genetic algorithm-based SVR model (OGA-SVR) uses a specific type of forward validation known as rolling window forward validation to estimate the model's performance on out-of-sample data. The rolling window forward validation takes into account any temporal patterns that may exist in the time series data.

ML involves organizing data into input features and output targets, but the task becomes more complex when it comes to multi-step prediction. Multi-step prediction involves mapping multiple output targets to input features, which can be particularly challenging when attempting to map long-term prices, such as daily prices over a year, to input features. Most studies predict next-day stock prices in stock price prediction, with some predicting intraday prices [23]. In their literature review, Rouf et al. [24] reported 24 studies, and those studies predicted intraday, daily, weekly, or up to 90 days, but none predicted yearly prices. However, accurately predicting stock prices over longer time horizons remains a challenging problem in ML. Many traders and investors are interested in predicting long-term prices, such as yearly predictions. Generally, for yearly prediction, yearly sampling is used, but this does not inform about the price pattern and fluctuations on a daily basis. However, to the best of our knowledge, no study has predicted yearly prices using daily stock data. This gap motivated us to predict the long-term prices of global indices using the SVR and LSTM algorithms. This study trains SVR and LSTM to identify patterns based on time dependency. As time is known for the future and provided as input to the models, our models can predict stock prices for any future time steps using historical data.

This study predicts the long-term prices of the five global indices with five models. The five global indexes are the Nifty from India, the Dow Jones Industrial Average (DJIA) from the US, the DAX performance Index (DAX) from Germany, Nikkei 225 (NI225) from Japan, and Shanghai Stock Exchange composite index (SSE) from China. These are stock indices from the world's top five economies in terms of Gross Domestic Product (GDP) [25]. The study optimizes the hyperparameters of the SVR model using a GA and rolling window forward validation. The rolling forward-validation approach enables the model to adapt to new data and avoid overfitting, while the GA helps to explore a large search space of possible solutions and find the optimal ones. The OGA-SVR effectively predicts one year ahead of stock indices prices. Further, the performance of OGA-SVR is compared with that of SVR, Grid Search-based SVR (GS-SVR), Genetic Algorithm-based SVR (GA-SVR), and LSTM. The models are evaluated using Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) on out-of-sample data.

The objectives of this study are to address the lack of emphasis on long-term and multistep prediction using daily prices in the literature. Specifically, the study aims to:

- Use daily historical data to predict stock prices one year ahead to help investors to take informed decisions in the long term.
- Demonstrate the superior performance of a rolling forward-validating genetic algorithm support vector regression (SVR) model over cross-validated SVR and genetic algorithm SVR models
- Optimize the OGA-SVR model to improve its predictions compared to other SVR and LSTM models
- Analyze and predict stock prices of multiple global indices to demonstrate the optimized model's robustness to perform well on different time series data.

Ultimately, this research endeavors to advance the development of effective long-term prediction models for stock prices and help make informed decisions by investors and traders over long-term time horizons.

The remainder of the paper is organized as follows: Section 2 presents the literature on stock prediction. Section 3 provides a detailed account of our model description and assumptions. Section 4 presents solution methods including experimental setup of models. Section 5 report results and discuss it. Finally, Section 6 concludes the paper and suggests future direction for future research.

2. Literature review

2.1. Long Short-Term Memory (LSTM) in stock prediction

Recently, the application of ML algorithms in predicting stock market prices has been a focal point in academic research within the field of time series analysis. ANNs have gained popularity in stock prediction because they can learn complex patterns and make generalizations. Teixeira Zavadzki de Pauli [26] compared the different architectures of ANNs to predict Brazilian stocks. They compared the performance of five Neural Network (NN) architectures to predict the close prices of six major stocks in the Brazilian stock exchange. The models were trained on historical stock prices to predict the next day's closing price. The models were evaluated based on the RMSE. The results showed that all architectures performed well except the radial basis function (RBF). Recurrent Neural Networks (RNNs), are a type of ANNs, specially developed to handle sequential data, such as time-series data. Yang et al. [27] proposed a hybrid model with improved Particle Swarm Optimization (PSO) algorithm and RNN to predict stock prices with better accuracy. The proposed algorithm, a combination of a NN and an adaptive adjustment of inertial weight, enhanced its global search ability in the early stage and local search ability in the later stage. The results indicated the practical effectiveness of the system. However, RNNs suffer from a vanishing gradient problem [28].

LSTM, a variant of RNN, is specifically designed to overcome the challenge of vanishing gradient that RNNs suffer. Moreover, the most frequently utilized ML algorithm for predicting financial time series is LSTM [13]. Mehtab et al. [29] proposed a hybrid modeling approach to predict stock prices using both ML and deep learning-based models. In this study, they used the NIFTY 50 index prices to construct eight regression models to predict the multi-step future open values of NIFTY 50 for the upcoming week. Four deep learning-based regression models were also built using LSTM networks with walk-forward validation and optimized hyperparameters. Their results showed that the LSTM-based univariate model that used one-week prior data as input was the most accurate model among all. Zhang et al. [30] applied a neural network and Back Propagation (BP) algorithm to forecast intraday stock prices by utilizing transaction data for five consecutive days. The BP algorithm neural network showed a prediction accuracy of 73.29%, whereas the deep learning fuzzy algorithm had an accuracy of 62.12%. They found that the best prediction range was 15 days, and the BP algorithm NN outperformed the deep learning fuzzy algorithm in terms of accuracy. Lu et al. [18] proposed a CNN-BiLSTM-AM model for predicting stock prices by combining CNN, bi-directional LSTM(BiLSTM), and attention mechanism (AM). Their model predicted the stock closing prices of the next day for the Shanghai Composite Index and compared the performance with seven other models as the benchmark. The results demonstrated that the CNN-BiLSTM-AM model was superior to the other models, with the smallest Mean Absolute Error (MAE), RMSE, and the largest R^2.

Some studies prefer predicting direction instead of price. Zhang et al. [31] predicted direction and developed a trading strategy. Further, they compared the performance of multiple models. They used Logistic Regression, SVM, Gradient-Boosting Decision Trees, RF, and LSTM. However, they experimented with the models on a single index, namely the Shanghai stock exchange (SSE) index. They proposed a methodology that considered both intraday and interday dynamics of the stock market during the 2015 stock market crisis in China. The empirical results demonstrated that the market can be predicted. Kumar and Haider [32] developed a hybrid mechanism that was a fusion of RNN-LSTM and combined it with metaheuristic optimization techniques to predict intra-day stock market trends. They introduced two hybrid approaches in their study. The proposed models provided a systematic method for automatically generating an optimized network, resulting in a more precise learning process with minimized error rates and improved accuracy. They evaluated the efficacy of the optimized RNN-LSTM network using data from six different stock exchanges, and the results showed that the metaheuristic approach increased the forecasting accuracy by approximately 4%–6%.

2.2. Support Vector Regression (SVR) in stock prediction

Another common machine-learning algorithm for stock prediction is SVM [33]. ANN and SVM are highly effective for predicting stock price movements and selecting appropriate model parameters [34]. While LSTM utilizes memory cells and gates to manage time-based connections between data points, SVM is a kernel-based algorithm that identifies the optimal hyperplane for class separation. SVR, a variation of SVM, is used for regression tasks and aims to find a hyperplane that fits the data closely with an acceptable degree of error, rather than dividing it into distinct classes like SVM. Zou et al. [35] proposed a Twin Support Vector Machines (TWSVM) prediction model. They employed thirteen indicators, extracted from historical data, as input features to predict the next day's stock direction. A comparison was made between the TWSVM predicting model and five other models, including decision tree, Naive-Bayes, RF, probabilistic neural network (PNN), and SVM. The results indicated that the TWSVM prediction model surpassed the other models in terms of predicting stock price and index daily movement. Doroudyan and Niaki [36] proposed a novel method based on an SVM for detecting upward and downward shifts in auto-correlated financial processes modeled by the Autoregressive Moving Average-Generalized Autoregressive Conditional Heteroskedasticity time series model. The authors identified specific features that can capture various characteristics of the patterns, which helps to detect the shifts in financial processes.

The SVM and SVR are frequently used with hyperparameters optimization techniques such as grid search (GS), GA, randomized search, or Bayesian optimization to find the best combination of hyperparameters for the given dataset and problem. These techniques help to efficiently search through the large hyperparameter space and find the best model configuration that can lead to better performance and reduce overfitting. Dash et al. [37] introduced a novel ML approach for time series stock forecasting using a fine-tuned version of SVR (FT-SVR). The authors used the grid search technique to find the fined tuned hyperparameters of SVR from the training set and validated them on separate data. The proposed method FT-SVR predicted the stock prices from eight large-sized datasets from various domains. The authors compared the FT-SVR with similar methods using RMSE and MAPE, which demonstrated that the proposed approach was more accurate in predicting stock performance for the utilized datasets and required less time compared to other methods. Mahmoodi et al. [38] utilized SVM with PSO to improve the classification. They compared the performance of SVM-PSO with two other meta-heuristic algorithms, namely the NN and the SVM-Cuckoo search(SVM-CS) algorithm. The results indicated that SVM-PSO outperformed SVM-CS and NN algorithms.

2.3. Genetic algorithm and hyperparameter optimization

The genetic algorithm (GA) possesses the capability to handle large-scale optimization problems and maintain diversity in the population, which helps in preventing premature convergence. Furthermore, genetic algorithms are computationally efficient and can rapidly converge to the optimal solution. Therefore,

Table 1
Recent stock price prediction studies.

Study	Dataset	Input features	Prediction methods	Evaluation metrics	Prediction duration
[41]	Crude Oil, German stock index (DAX)	Historical data	Hybrid DNN, LSTM, CNN	RSME, MAE	Daily
[42]	Korean Index and Individual Stocks	Historical data, Technical Indicator, Google trend	BPNN, LSTM, GRU, CNN	MAE, MSE, Accuracy, R^2	Daily
[43]	Korean Index	Historical data, Technical Indicators	Hybrid GA-XGBoost	Accuracy	Daily
[37]	State Bank of India	Historical data, Technical Indicators	SVR	RMSE, MAPE	Daily, Monthly
[44]	Portfolio	Macro-economic	Random Forest	Sharpe and Calmer ratio	Monthly
[18]	SSE Composite Index	Historical data	Hybrid CNN, BiLSTM	RMSE, MAE, R^2	Daily

GA is frequently utilized to search for the best combination of hyperparameters of SVM or SVR, such as the kernel type, regularization parameter, and kernel coefficient that minimize the error of the SVR model on a validation set. Li and Sun [39] presented a predictive model combining kernel parameters and optimization with the SVM model. The authors employed mesh search, genetic algorithm, and PSO to optimize SVM parameters. The results suggested that optimizing SVM parameters improved prediction accuracy, with the GA under the radial basis kernel (RBF) function achieving the best performance for the stock market forecast. The PSO was slightly less effective than the GS method. Comparison experiments demonstrated that the BP neural network was less accurate than the SVM in predicting the stock market. Similarly, for deep learning models, a GA can also be used to optimize parameters such as the number of hidden layers, the number of nodes per layer, the activation function, and the learning rate, among others. Gao et al. [40] proposed a deep learning approach that combined a GA to predict the overnight return direction of the stock market indices. The method utilized worldwide stock market indices as an information source and employed multiple convolution units to extract features from all regions. The model was optimized using a GA and applied to forecast the overnight return directions of nine stock indices from different locations. The results showed that the proposed model performed better than other competing methods in terms of accuracy, F-measure, and Sharpe ratio.

Solares et al. [45] presented a comprehensive decision support system that was proposed for the management of stock portfolios that addressed the forecasting of price, the selection of stock, and the optimization of the stock portfolios. The system utilized ANNs, differential evolution, and GA in different stages. The authors evaluated the system using historical data from the S&P 500 index and compared it to multiple benchmarks. The results demonstrated superior performance compared to the benchmarks. Thakkar and Chaudhari [46] proposed a method based on GA and information fusion, called inter–intra crossover and adaptive mutation (ICAN), for predicting stock prices and trends. The study proposed a method to optimize the parameters and feature selection of an LSTM model using a genetic algorithm with ICAN. Their method outperformed existing GA-based optimization approaches in terms of mean squared error, mean absolute error, MAPE, and R^2 score. Another proposed algorithm used a hybridized genetic algorithm-machine learning regressions for feature selection and achieved a parsimonious feature subset for interpretability and improved the average forecasting RMSE.

2.4. Research gaps

In stock price prediction, the majority of studies focus on predicting next-day stock prices, while some studies aim to predict intraday prices [23]. Chen et al. [47] predicted the prices of commercial bank shares in China using single-step and multi-step output for the next day, the next 5 days, and the next 22 days' price. The authors introduced a novel hybrid deep learning approach to enhance prediction accuracy. They proposed a novel hybrid method that involves using a K-means clustering algorithm to group banks that have the same price trends, followed by training an LSTM model to forecast stock prices. The LSTM network generates multi-step output across multiple time intervals, thereby boosting the accuracy of the 5-day and 22-day predictions. In their experiments, the hybrid model outperformed single models in terms of generalization ability and accuracy. Lv et al. [48] predicted the price of multiple stock indices. They proposed a new hybrid model called CEEMDAN-DAE-LSTM for stock index prediction. They conducted experiments using a dataset consisting of two emerging market indices and four developed market indices. The approach decomposed the stock indices using complete ensemble empirical mode decomposition with adaptive noise (CEEMDAN), applied deep autoencoder (DAE) to extract deep-level features, and then used LSTM networks to predict stock returns. Finally, the predicted value was obtained by combining the values of each component. The model showed superior performance in both prediction accuracy and stock index trends, particularly for stock indices with higher volatility, compared to other reference models.

Most of the study focuses on predicting next-day prices or directions. Some studies focus on a short-term prediction but there is a lack of emphasis on long-term prediction which is an important area for financial time series forecasting. Table 1 shows some recent studies with prediction duration. The full form of abbreviations is given in Appendix.

Hence, according to existing literature, there is a lack of research on applying ML techniques to predict long-term prices, such as yearly stock prices using daily data. To address this gap, the authors propose a method that utilizes SVR and GA, which can benefit investors and analysts seeking a longer-term view. To enhance the accuracy of our method, we employ rolling forward validating GA fitness function. Furthermore, few studies focus on experimenting with their models on multiple global indices. In this study, we experiment with the proposed method on multiple global indices. Additionally, the study compares the models with grid search-based SVR, genetic algorithm-based SVR, and LSTM.

3. Model description and assumptions

In this section, we present the details of the methodology to build a long-term stock price forecasting model. The comprehensive explanations of each step are provided in the following subsections.

3.1. Support Vector Regression (SVR)

The present form of Support Vector Machine (SVM) is largely attributed to the contributions of Vapnik et al. [49–54]. SVM is a type of supervised ML algorithm used for classification of data. If in a dataset, data points are given $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$,

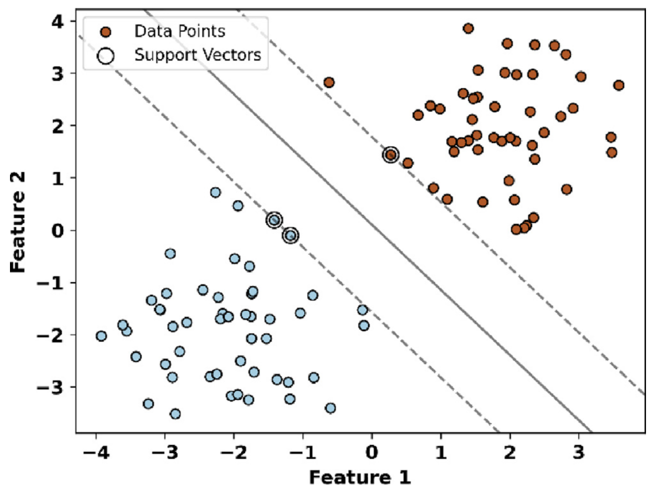


Fig. 1. Support vector machine.

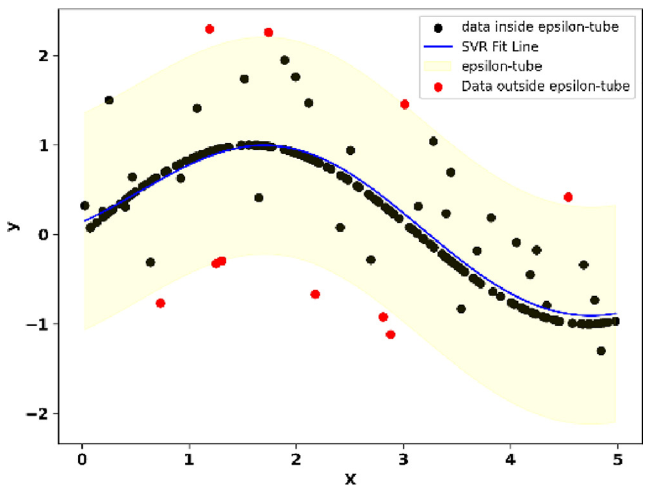


Fig. 2. Support vector regression.

where x_i is the input vector of features and y_i is the corresponding output, the SVM algorithm detects the weight vector w and the bias term b that define the hyperplane in Eq. (1):

$$y = w * x + b \tag{1}$$

The weight vector w determines the orientation of the hyperplane, while the bias term b sets its position. The hyperplane segregates the feature space into two regions, one for each class. SVM seeks to find the optimal hyperplane that best separates the two classes. The optimal hyperplane is the one that maximizes the margin, which is the distance between the hyperplane and the closest data points of each class. The margin is defined as in Eq. (2):

$$\gamma = \frac{2}{\|w\|} \tag{2}$$

where γ denotes margin and $\|w\|$ represents the Euclidean norm of the weight vector. To ensure that outliers do not influence the optimal hyperplane, SVM introduces the concept of "slack variables". The slack variables ξ_i and $\xi_i{}^*$ enable certain data points to exist on the incorrect side of the hyperplane while still penalizing them for being misclassified. The constraints on the slack variables are defined in Eq. (3):

$$\begin{aligned} & y_i - w * x_i - b \le \varepsilon + \xi_i \\ & w * x_i + b - y_i \le \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \ge 0 \end{aligned} \tag{3}$$

where ε is a small positive constant that defines the width of the margin, y_i is the output or class label of the data points. The optimization problem for SVM can be formulated as in Eq. (4):

$$\begin{aligned} & minimize \frac{1}{2}\|w\|^2 + C * \sum\left(\xi_i + \xi_i^*\right) \\ & subject\ to : y_i - w * x_i - b \le \varepsilon + \xi_i \\ & w * x_i + b - y_i \le \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \ge 0 \end{aligned} \tag{4}$$

where C is a positive constant that controls the trade-off between maximizing the margin and minimizing the slack variables. The optimization problem can be solved using various methods, such as the quadratic programming method or the gradient descent method. Fig. 1 shows the SVM.

Support Vector Regression (SVR) is a variant of the SVM algorithm that is widely employed for regression analysis. The SVR algorithm finds the function f(x) that approximates the relationship between the input features and the output values. The function f(x) is defined as in Eq. (5):

$$f(x) = w * x + b \tag{5}$$

where f(x) is the predicted output value, x is the input vector of features, w is the weight vector, and b is the bias term. The goal of SVR is to find the optimal values of w and b that minimize the error between the predicted output values and the actual output values. The optimization problem for SVR can be formulated as in Eq. (6):

$$\begin{aligned} & minimize \frac{1}{2}\|w\|^2 + C * \sum\left(\xi_i + \xi_i^*\right) \\ & subject\ to : y - f(x) \le \varepsilon + \xi_i \\ & f(x) - y \le \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \ge 0 \end{aligned} \tag{6}$$

where C is a positive constant that controls the trade-off between minimizing the error and minimizing the slack variables, ε is the width of the insensitive zone, and y is the actual output value. The optimization problem in SVM and SVR have some similarities, but they have different constraints. The constraints of SVM aim to maximize the margin between the classes, while the constraints of SVR focus on minimizing the error margin of the function f(x). Additionally, the loss function used to measure the error in SVR and SVR is different. Fig. 2 shows SVR.

3.2. *Long Short-Term Memory (LSTM)*

Long Short-Term Memory (LSTM), a type of RNN, overcomes the problem of the vanishing gradient that is common in traditional RNNs. Hochreiter and Schmidhuber [55] introduced LSTM to address this problem of RNNs. LSTMs are specifically designed to acquire and preserve sequential relationships in time-series data over an extended period of time. An LSTM unit comprises two states and three gates, with the gates regulating the information flow into and out of the cell state, while the states preserve the internal memory of LSTM.

- Input Gate: The input gate uses a sigmoid function to decide the weights of each value in the current input and adds the weights to the cell state. The sigmoid function produces an output range of values between 0 and 1.

- Forget Gate: The forget gate assesses which elements of the previous cell state should be retained or discarded, and the amount to forget is determined by a sigmoid function, which produces an output range of values between 0 and 1.
- Output Gate: The output gate determines which weights from the cell state should be output to compute the next hidden state. The gate uses a sigmoid function and a hyperbolic tangent (tan*h*) function to produce an output value in the range between −1 and 1, indicating the importance and strength of each cell state value.
- Cell State: The cell state serves as the internal memory of the LSTM, storing data that has been accumulated over time and updated by the input gate and forget gate.
- Hidden State: The hidden state is the output of the LSTM unit and is utilized for forecasting purposes. It is produced by the output gate and is reliant on the present input, prior hidden state, and the current cell state.

The gates and states of an LSTM unit function in tandem to enable the network to selectively preserve or discard data over time, resulting in an efficient tool for analyzing sequential data. The hidden state at time t, h_t, in an LSTM network is calculated using the input at time t, x_t, the previous hidden state at time t − 1, h_{t-1}, and the cell state at time t − 1, c_{t-1}. The cell state is updated based on x_t and h_{t-1}, and the updated cell state, c_t, is then passed through gates that decide which information to keep and which to discard. The forget gate, f_t, decide which information to discard from the previous cell state, c_{t-1}, by applying a sigmoid function to a linear transformation of x_t and h_{t-1}. The forget gate can be expressed as in Eq. (7):

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{7}$$

where W_f, U_f, and b_f are the weight matrix, recurrent weight matrix, and bias vector for the forget gate, respectively. The input gate, i_t determines which new information should be added to the cell state by applying sigmoid and tan*h* functions to a linear transformation of x_t and h_{t-1}. The sigmoid function controls which elements of the input should be updated, while the tan*h* function creates a vector of new candidate values to be added to the cell state. The input gate can be expressed as in Eq. (8):

$$i_t = \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) \tag{8}$$
$$\tilde{c}_t = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right)$$

where W_i, U_i, b_i, W_c, U_c, and b_c are the weight matrix, recurrent weight matrix, and bias vector for the input gate and the candidate values, respectively. The cell state, c_t, is updated by forgetting old information and adding new information, as in Eq. (9):

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{9}$$

where $\odot$ denotes element-wise multiplication, the input gate, i_t, determines which new information should be added to the cell state by applying sigmoid and tan*h* functions to a linear transformation of x_t and h_{t-1}. The sigmoid function controls which elements of the input should be updated, while the *tanh* function creates a vector of new candidate values to be added to the cell state. The output gate can be expressed as in Eq. (10):

$$o_t = \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) \tag{10}$$
$$h_t = o_t \odot \tanh\left(c_t\right)$$

where W_o, U_o, and b_o are the weight matrix, recurrent weight matrix, and bias vector for the output gate, respectively. By using these gates, LSTMs are able to selectively retain important information from the previous cell state, add new information based on the current input, and output relevant information as the hidden state. This enables LSTMs to capture long-term dependencies and overcome the vanishing gradient problem of traditional RNNs. Fig. 3 shows the LSTM unit.

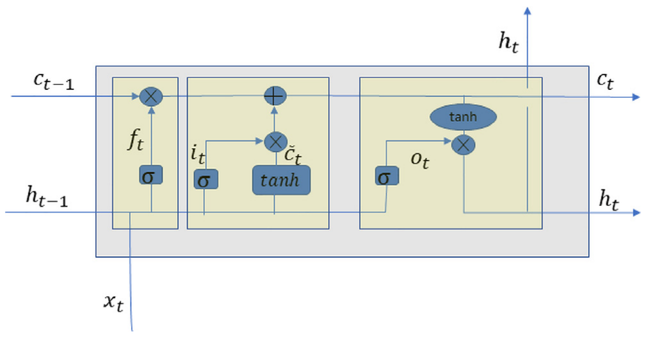


Fig. 3. LSTM unit.

3.3. Genetic Algorithm (GA)

The inspiration for Genetic Algorithms (GAs) came from Darwin's theory of evolution which involves simulating the survival of fitter creatures and their genes [17,56,57]. When the problem is computationally demanding and difficult to solve, GA can be used to obtain a solution that is close to optimal [58,59]. GAs are often used in ML and time series forecasting, where they can be used to identify optimal hyperparameters for ML models and to search for the best possible models for predicting future values in time series data. GA involves six stages: initialization, fitness calculation, selection, crossover, mutation, and termination condition check [60]. GA starts by randomly generating a population of chromosomes that represent potential solutions to a problem. These solutions are sorted based on a fitness score like accuracy, RMSE, etc. If the stopping criteria are met, the best solutions are returned, otherwise, a new generation is created through crossover and mutation. The new generation is evaluated again, and the process continues iteratively until the stopping criteria are met. Figs. 4(a) and 4(b) explain the flow of GA.

3.4. Rolling window forward validation

One of the crucial tools for assessing the performance of regression and classification techniques is cross-validation [61]. Cross-validation is widely utilized for evaluating how well a model performs on data that has not been seen during training, and it is commonly employed for both selecting and evaluating models. Cross-validation involves dividing a dataset into multiple subsets or folds for model training and testing. One fold is held out as a validation set for evaluation, while the remaining folds are used for training. This process is repeated for each fold, and the performance metrics are averaged to estimate the performance of generalization on unseen or out-of-sample data. The validation set is important to optimize hyperparameters and prevent overfitting of the model during training. K-fold cross-validation, a variation of cross-validation, is the most common [19]. k-fold cross-validation divides the dataset into k equal-sized subsets, where k is a user-defined positive integer. 5-fold cross-validation and 10-fold cross-validation are frequently used. After division, the model is trained on k−1 partitions and evaluated on the unseen partition. This process is repeated k times, with each subset serving as the validation set once for training a model. The evaluation metric such as accuracy or mean squared error (MSE) is recorded after each run. Lastly, the average is computed across all k runs for the final evaluation metric. This approach is a more reliable estimation of the performance of unseen data.

Cross-validation assumes that observations are independent and identically distributed which may not hold true in time

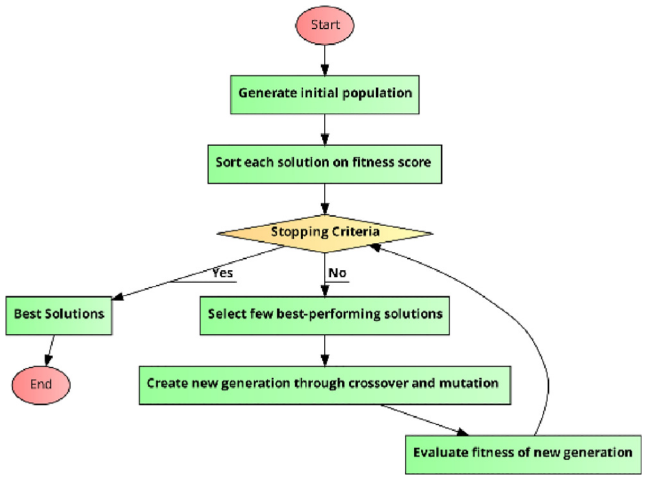

Start
Generate initial population
Sort each solution on fitness score
Stopping Criteria
Yes
No
Best Solutions
End
Select few best-performing solutions
Create new generation through crossover and mutation
Evaluate fitness of new generation

Fig. 4(a). Genetic algorithm flow.

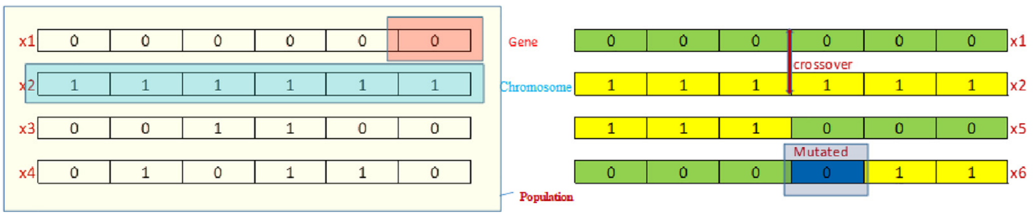


Fig. 4(b). Genetic algorithm operations.

series analysis. As an alternative, forward validation can be used. Forward validation is a technique that partitions the dataset into consecutive subsets, based on time instead of random assignment. This technique is particularly useful for time series data, where the temporal order of observations is important. By training the model on earlier subsets and validating it on later subsets, forward validation simulates the real-world scenario in which the model is deployed to predict future values based on historical data. One of the variants of forward validation is rolling window forward validation. In rolling window forward validation, the size of the subset remains constant, but they are shifted forward by a certain number of observations at each step. Rolling window forward validation can be especially helpful for stock time series data, where recent data is more relevant in predicting future. Figs. 5(a) and 5(b) visually explain cross-validation and rolling window forward validation.

3.5. Grid search

Grid search (GS) is a technique that involves searching through a pre-defined set of hyperparameters to identify the optimal combination of hyperparameters for a given model, by performing an exhaustive search [16]. GS divides parameters into grids of the same length within a certain range, with each point representing a set of parameters. Traversing all the points in the grid enables obtaining the optimal solution [62]. The model is then trained and evaluated for each combination of hyperparameters in the grid, and the optimal combination is selected based on the performance metrics. GS is commonly used with supervised ML algorithms, such as SVR. For example, when using SVR, one can fine tune hyperparameters such as the penalty parameter 'C' and the gamma parameter. A GS can test all possible combinations of values within specified ranges. For instance, the grid may include values of the penalty parameter (1, 10, 100) and gamma

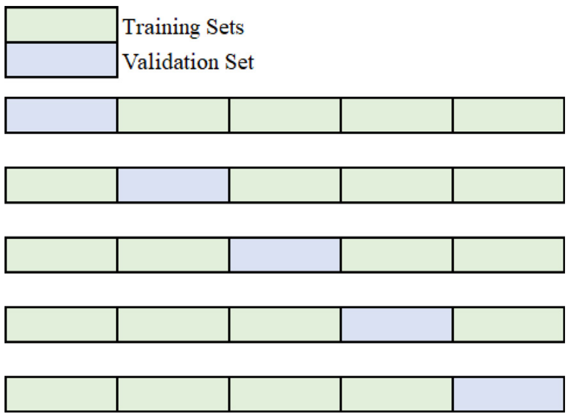


Fig. 5(a). 5-fold cross validation.

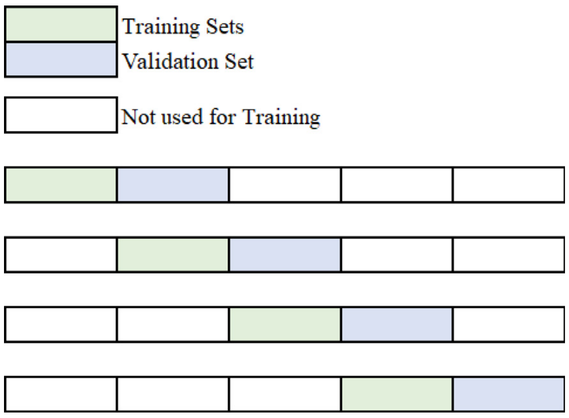
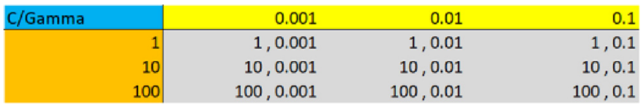


Fig. 5(b). Rolling window forward validation.



C/Gamma	0.001	0.01	0.1
1	1 , 0.001	1 , 0.01	1 , 0.1
10	10 , 0.001	10 , 0.01	10 , 0.1
100	100 , 0.001	100 , 0.01	100 , 0.1

Fig. 6. Grid search example.

parameter (0.001, 0.01, 0.1). Fig. 6 shows an example of the combination of the parameters in the grid search. GS can train and evaluate for each combination of these hyperparameters, and the combination with the best performance on cross validation dataset is selected. GS uses brute force. As the number of hyperparameters and their ranges increases, the number of models to be trained and evaluated increases exponentially. Therefore, this process can become computationally expensive, especially when dealing with many hyperparameters and a large dataset.

3.6. Prediction models and assumptions

This study experiments with five models, namely Support Vector Regression (SVR), Grid Search based Support Vector Regression (GS-SVR), Genetic Algorithm based Support Vector Regression (GA-SVR), Optimized Genetic Algorithm Support Vector Regression (OGA-SVR), and Long Short Term Memory (LSTM). The study optimized the predictions of OGA-SVR using a genetic algorithm and rolling window forward validation to tune the hyperparameters of SVR. The other models, SVR, GS-SVR, GA-SVR, and LSTM, are used as baseline models to compare the performance of the optimized model OGA-SVR. In this study, Radial Bias Kernel (RBF) and default value to gamma in Scikit-learn Python library is used for all models of SVR. The default value of gamma in the Scikit-learn Python library is 'scale' shown in the equation Eq. (11):

$$Scale = \frac{1}{n\_feature \times Var\,(X)} \tag{11}$$

The other two important hyperparameters of SVR are C and epsilon, which play a critical role in training the model. The cost parameter, denoted by C, is a regularization parameter that balances the trade-off between minimizing the training error and the complexity of the model. When C is small, the margin is wider, and the model allows more training errors. Conversely, a larger C value leads to a narrower margin and fewer training errors. In SVR, C controls the degree to which the margin is allowed to be violated in the training data. Epsilon is another threshold parameter in SVR that sets the minimum distance between the predicted and actual values before an error is counted. If the predicted value is within the epsilon distance of the actual value, it is considered accurate and has zero error. Any predicted value beyond the epsilon distance of the actual value is considered inaccurate and has a non-zero error. Choosing a smaller value of epsilon increases the sensitivity of the model to the errors, whereas a larger value of epsilon results in a less sensitive model. These two parameters of SVR are optimized in other models of SVR. Finally, the predictions by the models are evaluated using RMSE and MAPE. Fig. 7 shows the general flow chart of prediction algorithms.

The following are the assumptions of the prediction models:

- The models assume the future pattern of the stock prices is related to historical data. This is against the random market hypothesis but aligns with technical analysis.
- The experimental model OGA-SVR assumes that rolling forward validation is more appropriate than cross-validation to improve predictability for stock market forecasting.

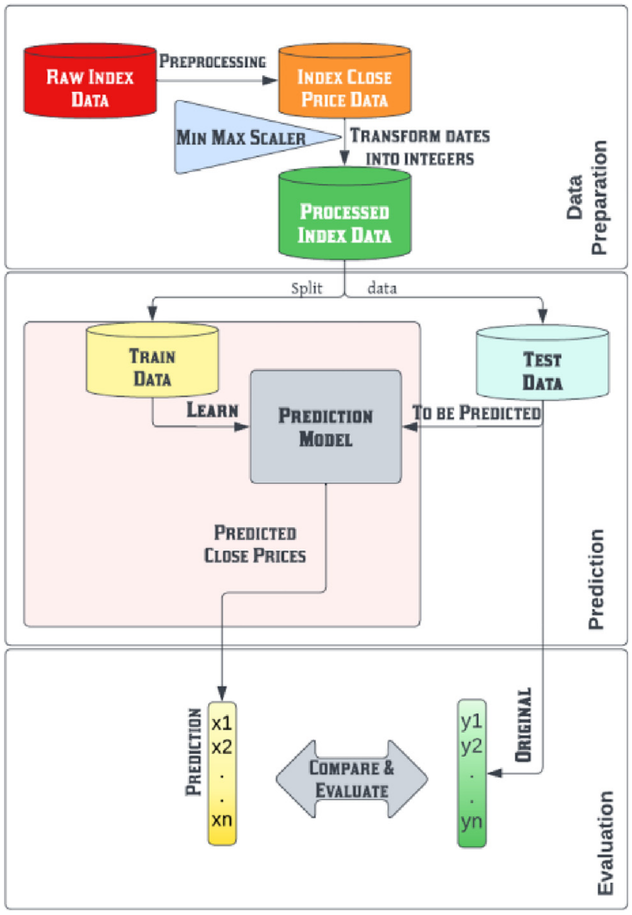


Fig. 7. Flowchart of prediction methods.

Table 2
SVR hyperparameters.

Hyperparameter	Values
Kernel	RBF
C	1
Epsilon	0.1
Gamma	Scale

4. Solution methods

4.1. SVR

SVR is the base model of this study. The model SVR is trained without hyperparameters optimization, and default values of hyperparameters are used. This is helpful in evaluating other model efficiencies. Table 2 lists the default value of SVR. Fig. 8 shows the flowchart of the prediction algorithm of SVR.

4.2. GS-SVR

The Grid Search-based Support Vector Regression (GS-SVR) algorithm uses a grid search approach to optimize the hyperparameters C and epsilon. Specifically, the Scikit-learn Python library's grid search function estimates the performance of different hyperparameters using cross-validation (CV). In this study, using grid search, 10-fold cross-validation was used to evaluate the performance of different hyperparameter combinations, with the Root Mean Squared Error (RMSE) used as the loss measurement. This approach aids in the training process by identifying

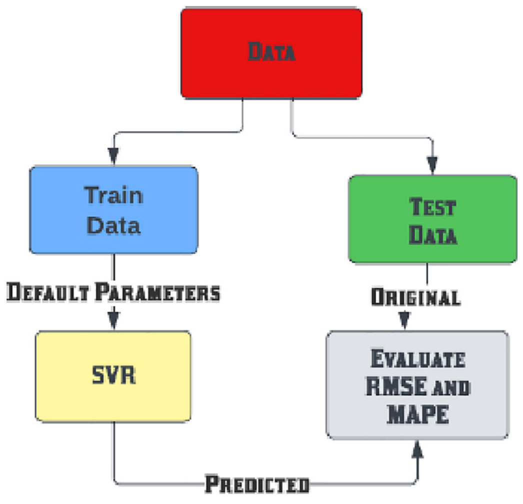


Fig. 8. Flowchart SVR prediction algorithm.

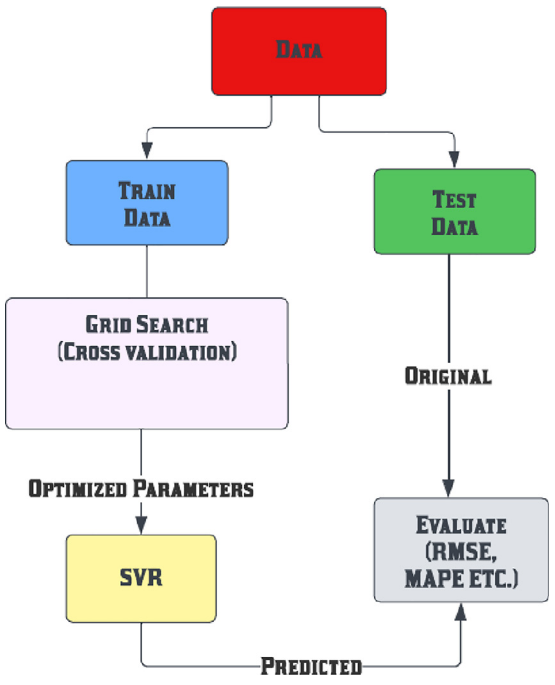


Fig. 9. Flowchart GS-SVR prediction algorithm.

Table 3
Grid search parameters.

Parameters	Values
param_grid	C: [0.001, 0.01, 0.1, 1, 10, 100] Epsilon: [0.1, 0.5, 0.1, 0.15, 0.2, 0.25]
scoring	neg_root_mean_squared_error
cv	10

the optimal combination of hyperparameters, which are then utilized in the Support Vector Regression (SVR) algorithm to make predictions about future prices. Table 3 shows the input to the grid search. Fig. 9 shows the flowchart of GS-SVR prediction.

4.3. GA-SVR

Genetic algorithm-based Support Vector Regression (GA-SVR) used a genetic algorithm instead of a grid search to optimize the hyperparameter C and epsilon. GS and GS are both techniques used for hyperparameter optimization but differ in their approaches. Grid search is a brute-force approach that evaluates the performance of all possible hyperparameter combinations within a pre-defined range. On the other hand, genetic algorithms are inspired by the natural selection process and survival of the fittest. Genetic algorithms use an iterative process to evaluate and optimize hyperparameters. In each iteration, the algorithm selects a set of hyperparameters, creates new sets by combining and mutating them, and evaluates their performance. The best-performing hyperparameters are then selected for the next iteration, and the process is repeated until exit criteria is met. In GA-SVR process, full training data is utilized to converge to best parameter. Fig. 10 shows the step of the genetic algorithm in this study.

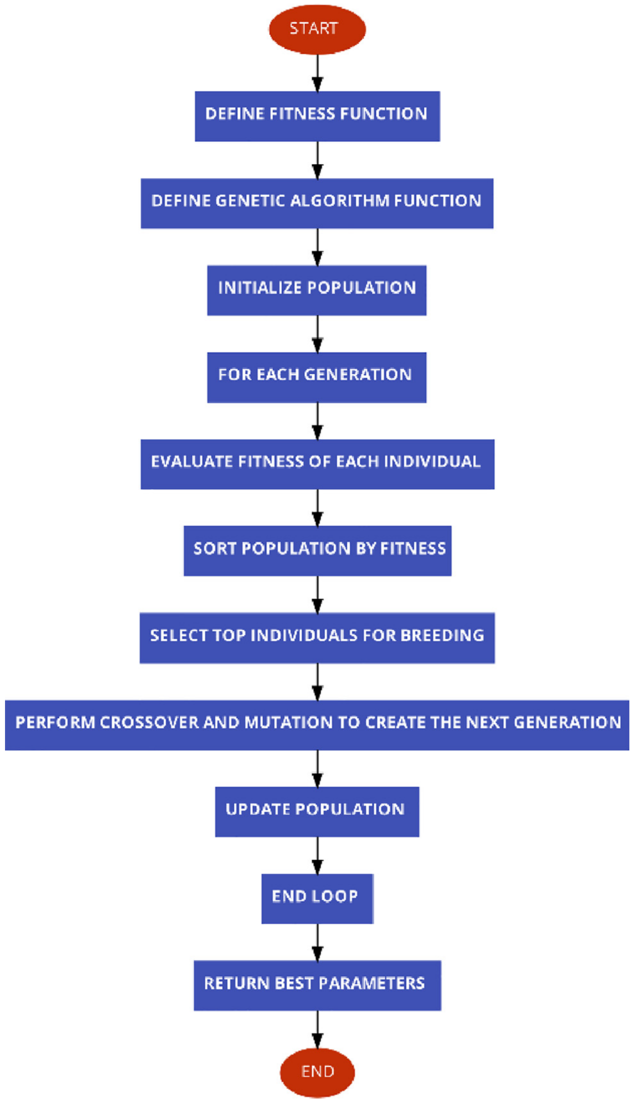

START
DEFINE FITNESS FUNCTION
DEFINE GENETIC ALGORITHM FUNCTION
INITIALIZE POPULATION
FOR EACH GENERATION
EVALUATE FITNESS OF EACH INDIVIDUAL
SORT POPULATION BY FITNESS
SELECT TOP INDIVIDUALS FOR BREEDING
PERFORM CROSSOVER AND MUTATION TO CREATE THE NEXT GENERATION
UPDATE POPULATION
END LOOP
RETURN BEST PARAMETERS
END

Fig. 10. Genetic algorithm steps.

The initial population comprises pairwise combinations of values of C and epsilon from Table 4. Each value from the C list is paired with every value from the epsilon list, resulting in 36 initial solutions. The top 50% of individuals from the population are selected as parents and allowed to undergo crossover with new randomly created children. Further, the mutation is permitted in 20% of the population, allowing the insertion of any random value in the individual solution, subject to the limit of maximum and

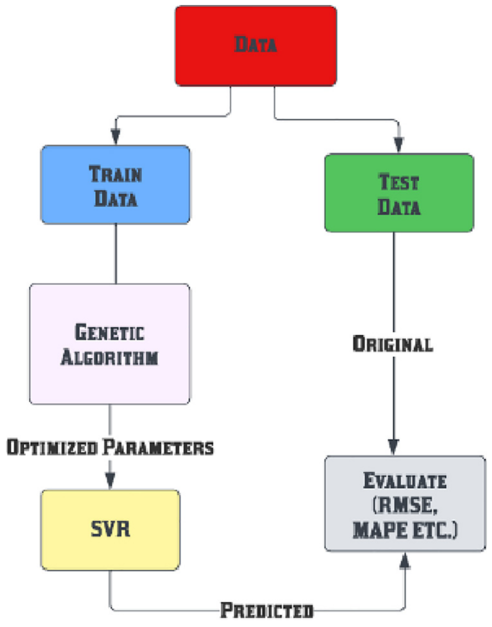


Fig. 11. Flowchart GA-SVR prediction algorithm.

Table 4
Chromosome initial population.

Chromosome	Initial values
C	0.001, 0.01, 0.1, 1, 10, 100
Epsilon	0.1, 0.5, 0.1, 0.15, 0.2, 0.25

minimum values of C and epsilon from Table 4. The exit criterion in this study is a maximum number of generations, which is set at 30. Similarly, [43] also used 30 generations while optimizing their machine-learning models. The fitness function of GA-SVR aims to minimize the root mean square error (RMSE) of the fit value of prices and training prices. Fig. 11 shows the flowchart of GA-SVR.

4.4. OGA-SVR

Optimized Genetic Algorithm-based Support Vector Regression (OGA-SVR) utilizes rolling window forward validation along with a genetic algorithm to optimize the hyperparameters C and epsilon of SVR. The genetic algorithm aims to find the best parameter on full training data, unlike grid search. In the Scikit-learn Python library, grid search has cross-validation inbuilt into it to estimate the performance on unseen data. In contrast to grid search, genetic algorithms do not inherently include cross-validation as a part of the optimization process on training data. This can lead to variations in the performance of the model on unseen data. Further, the assumption of independent and identically distributed observations, which is made in cross-validation, is not valid in time series analysis. Forward validation is proposed as an alternative to cross-validation. According to the empirical study of [19], forward-validation techniques were found to provide more accurate estimates of the out-of-sample error than other validation schemes. Hence, in this study, to optimize the performance on unseen data, rolling window forward validation is incorporated in the fitness function of the genetic algorithm in the model OGA-SVR.

Fig. 12 shows the flowchart of the OGA-SVR prediction algorithm. In this study, rolling window forward validation divides the data year-wise and trains the model on one year's data while validating it on the next year's data. In the next step, the model is trained on the data it was validated on and is further

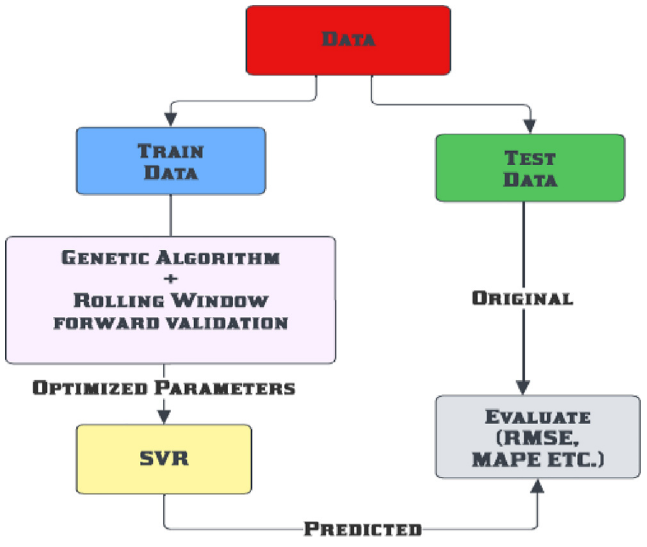

Data
Train Data
Test Data
Genetic Algorithm + Rolling Window forward validation
Original
Optimized Parameters
SVR
Evaluate (RMSE, MAPE etc.)
Predicted

Fig. 12. Flowchart OGA-SVR prediction algorithm.

validated in the next year of the currently trained window. This process continues until the training reaches the last year of data. The fitness function of OGA-SVR aims to minimize the average RMSE of all validated data, whereas the fitness function of the genetic algorithm aims to minimize the RMSE of the full training dataset. The remaining operations of OGA-SVR, including population initialization, crossover, and mutation, are similar to those of GA-SVR, as discussed in the preceding section.

4.5. LSTM

The Long Short-Term Memory (LSTM) algorithm is a powerful tool for processing sequential data, such as stock time series data. In this study, after trial and error, we stacked two LSTM layers and a dense output layer sequentially. The LSTM layers are recurrent neural networks that can handle sequential data, while the dense layer is a fully connected neural network layer that processes the output of the LSTM layers to produce a final output. Using two LSTM layers makes this architecture deeper than a single LSTM layer architecture which can handle complex patterns of stock prices. Additionally, we added dropout layers to reduce overfitting and make the model more robust. Dropout is a regularization technique commonly used in neural networks to reduce overfitting. The dropout process involves the selective omission of a portion of the neurons in a layer during training by setting them to zero. This approach helps the neural network learn more resilient features, preventing it from becoming overly reliant on any feature. The number of epochs is 100, the batch size is 32, and the dropout percentage is 20% for all indices data. Fig. 13 shows the architecture of the LSTM model.

5. Results and discussion

This study experiments with five global indices: the Nifty from India, the Dow Jones Industrial Average (DJIA) from the US, the DAX performance Index (DAX) from Germany, Nikkei 225 (NI225) from Japan, and Shanghai Stock Exchange composite index (SSE) from China. These stock indices represent the world's top five economies in terms of GDP [25]. The study obtains 10 years of data for all indices from Yahoo Finance spanning from January 1st, 2013, to December 31st, 2022. The training set uses data from January 1st, 2013, to December 31st, 2021, and the testing set reserves data from January 1st, 2022, to December 31st, 2022. The

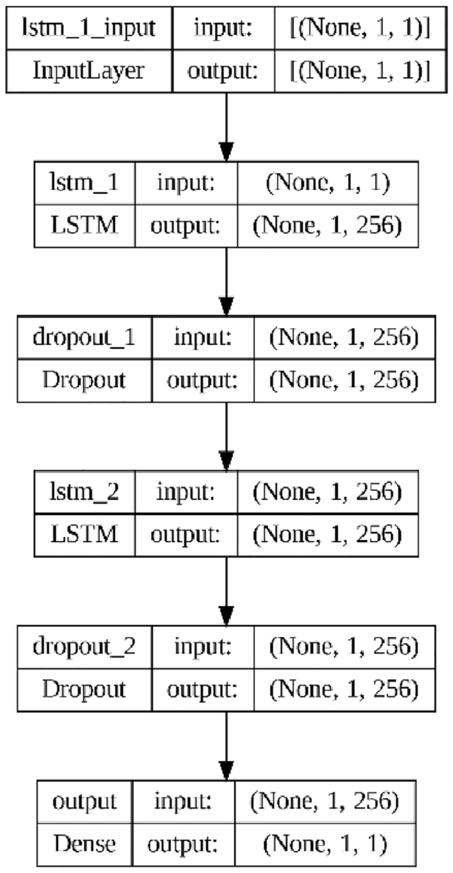

lstm_1_input | input: | [(None, 1, 1)]
InputLayer | output: | [(None, 1, 1)]
lstm_1 | input: | (None, 1, 1)
LSTM | output: | (None, 1, 256)
dropout_1 | input: | (None, 1, 256)
Dropout | output: | (None, 1, 256)
lstm_2 | input: | (None, 1, 256)
LSTM | output: | (None, 1, 256)
dropout_2 | input: | (None, 1, 256)
Dropout | output: | (None, 1, 256)
output | input: | (None, 1, 256)
Dense | output: | (None, 1, 1)

Fig. 13. LSTM architecture for stock price prediction.

data is then divided into training and testing sets, with a ratio of 90:10 for all models. In GS-SVR, the training data is then split for 10-fold cross-validation. In contrast, in OGA-SVR, the data is divided year-wise to utilize consecutive years of data in training and validation in rolling window forward validation. The data is preprocessed for training by removing fields such as Open, High, and Low prices and the Adjusted Close and Volume fields. The data is then reduced to only include the date index and close price. The date index is encoded into integers ranging from 1 to n. The encoded date values and close prices are further transformed using the min–max scaler equation shown in Eq. (12).

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{12}$$

where X is the feature matrix, X_{min} and X_{max} represent the minimum and maximum values of X, respectively. Since input features rely on previous prices, the models can predict prices for any time in the future using only future dates. After training, the models predict future prices using future dates from the testing data. The data is then inverse-transformed and compared to the testing data. The study evaluates the results using RMSE and MAPE in the training and testing data. RMSE is used to compare the model on the same dataset, while MAPE is used to compare models on a different dataset.

5.1. NIFTY

Table 5 shows the results for NIFTY during training and testing. There is no difference in performance between SVR and GS-SVR, as their train and test RMSE and MAPE values are the same. This is because the best-estimated parameters by grid search for GS-SVR are the same as the default parameters of SVR. In contrast, GA-SVR performs better in both the training and testing data than SVR and GS-SVR.

Due to its strong capability to handle sequential data, LSTM has the second-best performance on the test data and the best on the training data. The optimized model OGA-SVR performs best on the test data but worst on the training data. Fig. 14 shows the models' prediction chart, where the gray line divides the training and testing data.

Table 5
Nifty.

Model	Train RMSE	Test RMSE	Train MAPE	Test MAPE
SVR	765.38	2787.05	7.24	15.29
GS-SVR	765.38	2787.05	7.24	15.29
GA-SVR	729.72	1721.7	6.64	8.87
OGA-SVR	1016.26	1296.24	9.18	6.36
LSTM	663.57	1522.84	5.61	7.72

Table 6
DJIA.

Model	Train RMSE	Test RMSE	Train MAPE	Test MAPE
SVR	1315.96	7064.22	5.04	20.28
GS-SVR	1315.96	7064.22	5.04	20.28
GA-SVR	1244.76	6653.96	4.44	18.95
OGA-SVR	1591.99	1961.45	5.98	4.87
LSTM	1625.81	2827.79	5.01	7.39

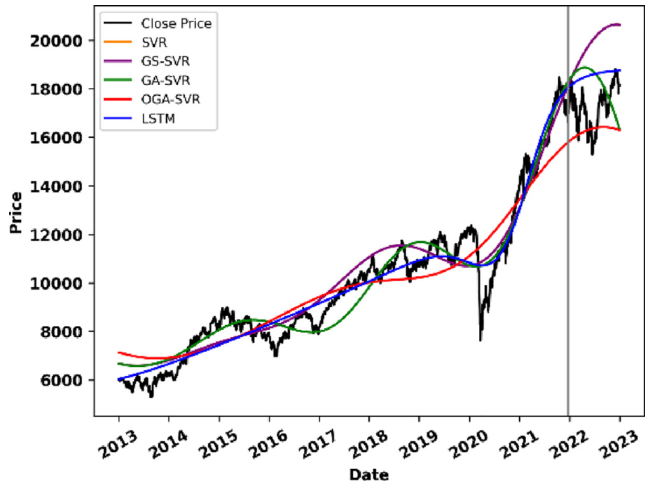


Fig. 14. Performance of models on NIFTY.

5.2. Dow Jones Industrial Average (DJIA)

Table 6 presents the performance of all models on DJIA. Similar to the Nifty results, SVR and GS-SVR exhibit similar performance. However, both SVR and GS-SVR demonstrate poor performance compared to the other models. GA-SVR exhibits some improvement in training and testing data compared to SVR and GS-SVR. LSTM exhibits the lowest RMSE and MAPE on training data but second-best on testing data. The optimized model exhibits the worst performance on training data but the best on testing data. Fig. 15 visualizes the performance of all models.

5.3. DAX performance index (DAX)

Table 7 presents the performance of all models on DAX. SVR has the worst performance on testing data. GS-SVR performs

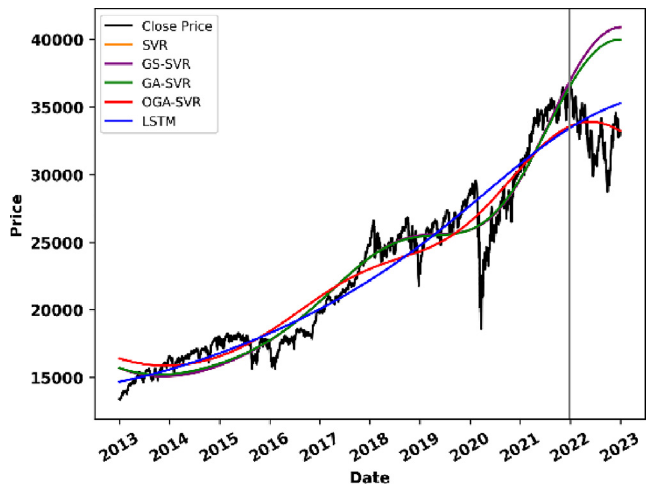


Fig. 15. Performance of models on DJIA.

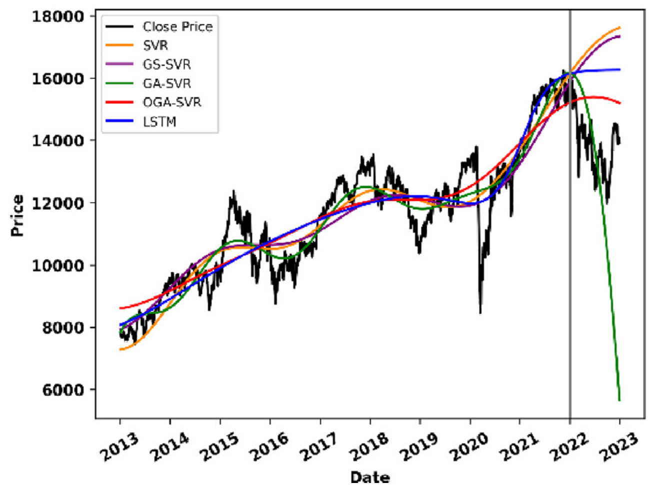


Fig. 16. Performance of models on DAX.

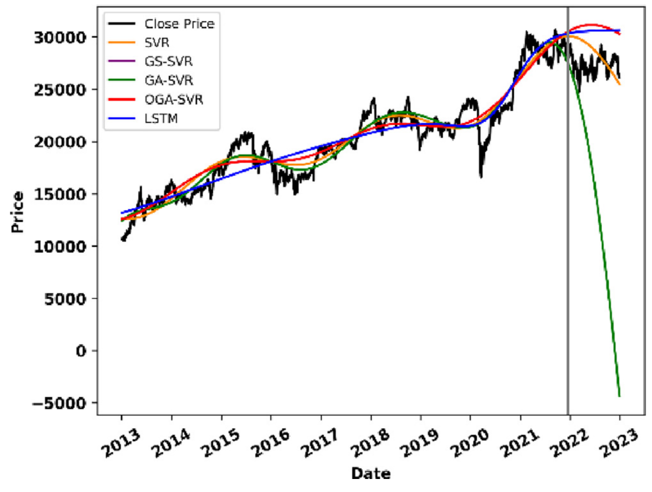


Fig. 17. Performance of models on Nikkei 225.

better than SVR, but both show poor performance compared to other models. The GA-SVR model performs best on training data compared to all other models but does not generalize well on testing data, indicating overfitting. Fig. 16 illustrates the fit and prediction prices of all models. Interestingly, unlike the previous LSTM performance on NIFTY and DJIA, the performance of LSTM is lower than that of GA-SVR on DAX. The optimized model OGA-SVR shows the lowest performance on training but the best on testing data.

Table 7
DAX.

Model	Train RMSE	Test RMSE	Train MAPE	Test MAPE
SVR	721.98	3416.08	5.14	23.62
GS-SVR	774.81	3207.71	5.4	21.99
GA-SVR	656.02	3094.12	4.54	15.49
OGA-SVR	843.06	1761.95	6.02	11.7
LSTM	779.22	2554.55	5.47	17.63

Table 8
Nikkei 225.

Model	Train RMSE	Test RMSE	Train MAPE	Test MAPE
SVR	1262.73	2042.3	5.38	6.4
GS-SVR	1379.07	3690.73	5.77	13.16
GA-SVR	1168.82	15487.49	4.91	45.11
OGA-SVR	1379.07	3690.73	5.77	13.16
LSTM	1507.74	3329.79	6.4	11.86

Table 9
SSE.

Model	Train RMSE	Test RMSE	Train MAPE	Test MAPE
SVR	311.85	419.06	7.33	12.43
GS-SVR	341.66	421.39	7.66	12.1
GA-SVR	269.61	931.85	6.67	28.1
OGA-SVR	313.06	502.3	7.53	15.08
LSTM	193.46	501.21	4.57	14.41

5.4. Nikkei 225 (NI225)

Table 8 presents the performance of all models on Nikkei 225. In contrast to previous results, the best performance on SVR models is the best on testing data. Fig. 17 shows the performance of all models. The RMSE is minimum for GA-SVR on training data but highest on test data. Hence, it is overfitting in the testing data. The GS-SVR and OGA-SVR models have similar results as both models estimate the same hyperparameters. The LSTM model is second best on testing data.

5.5. Shanghai Stock Exchange composite index (SSE)

Table 9 presents the performance of all models on SSE. The GA-SVR performs worst on test data. Fig. 18 shows the fit and predicted prices of all models on SSE. The best performance exhibited by GS-SVR. The performance of SVR is close to GS-SVR. The performance of the optimized model is less than SVR, GS-SVR, and LSTM on testing data.

5.6. Consolidated result

To compare the performance of all models on all datasets, we consolidated the result in Table 10. The GA-SVR model exhibits good performance on testing data but worst on testing data. This result implies that the genetic algorithm finds the hyperparameters which overfit training data. The reason for overfitting can be attributed to the absence of validation on unseen data. However, the is also an absence of validation on unseen data in the SVR model, but its performance on testing data is better than GA-SVR. The MAPE of training data of GA-SVR is lesser than the MAPE of SVR. This indicates that finding the best parameter using training datasets does not guarantee good performance on testing data. There is an increase in RMSE value in GS-SVR compared to the RMSE values of SVR and GA-SVR. However, the performance of

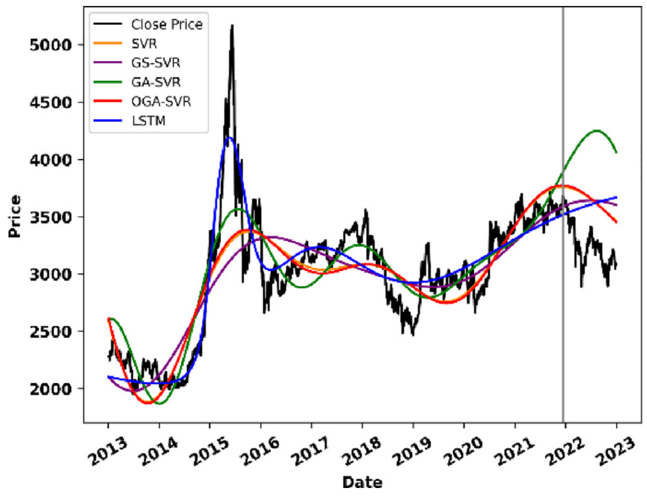


Fig. 18. Performance of models on SSE.

Table 10
Consolidated.

Model	Train MAPE	Test MAPE
SVR	6.03	15.60
GS-SVR	6.22	16.56
GA-SVR	5.44	23.30
OGA-SVR	6.90	10.23
LSTM	5.41	11.80

GS-SVR is better than GA-SVR but less than SVR on the training dataset. GS-SVR uses 10-fold cross-validation, which should increase the model's generalization ability on the testing dataset. However, the results do not comply. The reason why 10-fold cross-validation does not improve the result of GS-SVR compared to SVR can be attributed to the violation of the assumption that observations are independent and identically distributed. In stock time series data, this assumption of cross-validation does not hold true. Hence, 10-fold cross-validation does not improve the performance of GS-SVR over SVR.

The training MAPE of the LSTM model is better compared to all other models. Hence, the LSTM network is able to fit the complex pattern of the stock market better than other models. Further, it also performs the second best on unseen test data. This result clearly reflects the outstanding ability of LSTM to handle sequential data. Based on MAPE on the training datasets, the optimized model OGA-SVR does not fit the prices as well as compared to other models. However, the performance of testing datasets is the best for OGA-SVR. This suggests that the stock prices are noisy, and fitting the model well on the training datasets can degrade the performance of the testing datasets. The OGA-SVR models used the rolling window forward validation. This helps the models compromise the performance on the training dataset to predict well on the testing dataset. Hence, in the stock market time series, higher RMSE or MAPE value on training datasets suggests that the model is bad. Further, the results also suggest that training a model on stock time series using rolling window forward validation is a better alternative to cross-validation to predict on testing datasets.

5.7. Managerial insights

The results clearly demonstrate the excellent forecasting ability of the proposed OGA-SVR approach. Therefore, this study recommends the use of the OGA-SVR model for long-term forecasting of stock market prices due to its superior performance on testing data. The study also highlights the importance of using rolling window forward validation to avoid overfitting on the training data and to improve the generalization ability of the model on testing data. Cross-validating grid search or genetic algorithm alone cannot generalize well on unseen testing data. Therefore, managers and investors should be cautious about relying on predictions solely based on grid search or genetic algorithm with SVR. This study also demonstrates that LSTM is also an appropriate algorithm for long-term stock prediction. Future research could explore the use of other models and validation techniques for stock market time series data to improve the accuracy of the predictions. The insights from this study can assist managers in making better investment decisions and understanding the limitations of the models used for stock market forecasting. The proposed approach (OGA-SVR) has never been used in previous studies to forecast long-term stock prices up to a year, and the present paper could be considered a landmark for long-term prediction of time series stock prices using forward validating genetic algorithm with SVR.

6. Conclusion and future recommendations

Predicting the stock market is controversial with some researchers supporting unpredictability [1,2] and some supporting predictability [3–6]. In recent times, the surge of electronic trading and the abundance of stock data has prompted researchers to investigate the potential of artificial intelligence in forecasting the stock market. LSTM and SVM/SVR are the most commonly used ML algorithms for forecasting financial time series [13]. In stock price prediction, the majority of studies focus on forecasting the stock prices for the following day [23]. The majority of research employs a daily sampling frequency with some statistical and technical calculations based on previous prices, so generally, prediction frequency is also daily. Nevertheless, accurately forecasting stock prices for the long term remains a difficult issue in the field of machine learning. In this study, we predicted long-term stock prices from daily data of global indices using a model of optimized genetic algorithm and support vector regression (OGA-SVR). Further, we compared the performance of the optimized model OGA-SVR with baseline models SVR, GS-SVR, GA-SVR, and the deep learning model of LSTM. Additionally, we empirically compared the impact of rolling forward validation and cross-validation in predicting long term stock prices.

While using daily data for prediction, we avoided using statistical calculations and technical indicators based on previous prices of historical data. This study only used dates, and close prices are input and output features, respectively. As dates were known input variables for the long-term duration, the models predicted daily close prices for up to a year. After empirical evaluation using RMSE and MAPE, we found that the optimized model OGA-SVR outperformed all baseline models on testing data. The optimized model exhibited better generalization ability by avoiding overfitting in training data. The model used rolling window forward validation, which helped it to perform well on unseen future data. Our results suggest that the generic algorithm was overfitting in the training process and hence performed poorly on testing data. We also found that cross-validation did not improve the result of SVR, as observations in time series data are not independent. Therefore, rolling window forward validation is more suitable for estimating the future performance in stock time series data. After OGA-SVR, LSTM had the best performance among other models. The performance of LSTM was good on both training and testing data, which affirms its outstanding ability to handle sequential data and suitability to stock time series data.

The methods and models utilized in this study can be further explored to develop trading, risk management, and anomaly detection systems. These systems can assist investors, traders, and analysts in making better-informed decisions. Additionally, researchers can add variables to improve the models' performance. Although the LSTM model showed good performance on training and testing data with two dropout layers, further optimization of hyperparameters such as the number of neurons, batch size, and the number of epochs can improve performance. Other ML algorithms, such as RF and XGBoost, can also be explored to improve performance. Further, other metaheuristic algorithms such as PSO, Simulated Annealing, Ant Colony Optimization, etc., could also be explored to evaluate their impact on long-term forecasting. Additionally, the daily data approach used to predict long-term or multi-step outputs can be applied to small sampling frequencies such as intraday, and profitability can be evaluated.

One of the limitations of this study is that it only tested the models on global indices, which have a tendency to go up and are comparatively less volatile than individual stock prices. It would be interesting to explore whether the approach produces similar results in individual stock prediction. Furthermore, the use of only a single input variable limits the models' performance. The performance can be improved by feeding prices predicting from sampling frequencies such as monthly, quarterly, and yearly.

CRediT authorship contribution statement

Mohit Beniwal: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft. **Archana Singh:** Supervision, Writing - review & editing. **Nand Kumar:** Supervision, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Appendix

Abbreviation	Explanation
DNN	Deep Neural Network
LSTM	Long Short-Term Memory
CNN	Convolutional Neural Network
BPNN	Backpropagation Neural Network
GRU	Gated Recurrent Unit
GA	Genetic Algorithm
XGBoost	Extreme Gradient Boosting
SVR	Support Vector Regression
BiLSTM	Bidirectional Long Short-Term Memory
ML	Machine Learning
ARIMA	Auto-Regressive Integrated Moving Average
SVM	Support Vector Machine
RF	Random Forest
LSTM	Long Short-Term Memory
AutoML	Automated Machine Learning
GS	Grid-Search
GA	Genetic Algorithms
AI	Artificial Intelligence
ARCH	Auto-Regressive Conditional Heteroscedasticity
VAR	Vector Auto-Regression
GS-SVR	Grid Search based Support Vector Regression
GA-SVR	Genetic Algorithm based Support Vector Regression
OGA-SVR	Optimized Genetic Algorithm Support Vector Regression
RSME	Root Mean Square Error
MAE	Mean Absolute Error
MSE	Mean Squared Error
ARIMA	AutoRegressive Integrated Moving Average
ARCH	Autoregressive Conditional Heteroskedasticity
VAR	Vector Autoregression
R^2	Coefficient of Determination
MAPE	Mean Absolute Percentage Error

References

[1] E.F. Fama, Efficient capital markets: A review of theory and empirical work, J Financ. 25 (1970) 383–417, http://www.jstor.org URL: http://www.jstor.org/stable/2325486 Accessed:30/04/200806:47.
[2] B.G. Malkiel, A Random Walk Down Wall Street, Norton & Co, New York, 1973.
[3] A.W. Lo, A.C. MacKinlay, Stock market prices do not follow random walks: Evidence from a simple specification test, Rev. Financ. Stud. 1 (1998) 41–61.
[4] J.M. Poterba, L.H. Summers, Mean reversion in stock prices: Evidence and implications, J. Financ. Econ. 22 (1988) 27–59.
[5] N. Jegadeesh, S. Titman, Returns to buying winners and selling losers: Implications for stock market efficiency, J. Financ. 48 (1993) 65–91, http://dx.doi.org/10.1111/j.1540-6261.1993.tb04702.x.
[6] H. J. Cochrane, New facts in finance, 1999.
[7] B. Graham, L. Dodd. David, The Intelligent Investor, Harper & Row, New York, 1965.
[8] J.S. Abarbanell, B.J. Bushee, Abnormal returns to a fundamental analysis strategy, Account. Rev. 73 (1998) 19–45.
[9] M. Ballings, D. van den Poel, N. Hespeels, R. Gryp, Evaluating multiple classifiers for stock price direction prediction, Expert Syst. Appl. 42 (2015) 7046–7056, http://dx.doi.org/10.1016/j.eswa.2015.05.013.
[10] O. Abdolazimi, F. Bahrami, D. Shishebori, M.A. Ardakani, A multi-objective closed-loop supply chain network design problem under parameter uncertainty: comparison of exact methods, Environ. Dev. Sustain. 24 (2022) 10768–10802, http://dx.doi.org/10.1007/s10668-021-01883-2.
[11] O. Abdolazimi, M. Salehi Esfandarani, M. Salehi, D. Shishebori, Robust design of a multi-objective closed-loop supply chain by integrating on-time delivery, cost, and environmental aspects, case study of a Tire Factory, J. Clean. Prod. 264 (2020) http://dx.doi.org/10.1016/j.jclepro.2020.121566.
[12] O. Abdolazimi, M.S. Esfandarani, D. Shishebori, Design of a supply chain network for determining the optimal number of items at the inventory groups based on ABC analysis: a comparison of exact and meta-heuristic methods, Neural Comput. Appl. 33 (2021) 6641–6656, http://dx.doi.org/10.1007/s00521-020-05428-y.
[13] B.M. Henrique, V.A. Sobreiro, H. Kimura, Stock price prediction using support vector regression on daily and up to the minute prices, J. Financ. Data Sci. 4 (2018) 183–201, http://dx.doi.org/10.1016/j.jfds.2018.04.003.
[14] K. Drachal, M. Pawłowski, A review of the applications of genetic algorithms to forecasting prices of commodities, Economies 9 (2021) http://dx.doi.org/10.3390/economies9010006.
[15] J. Taljard, The Use of Genetic Algorithms for Automated Machine Learning in Trend Prediction in Time Series Data: A Review, (n.d.).
[16] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2012) 281–305.
[17] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1992) 66–73, http://dx.doi.org/10.2307/24939139.
[18] W. Lu, J. Li, J. Wang, L. Qin, A CNN-bilstm-AM method for stock price prediction, Neural Comput. Appl. 33 (2021) 4741–4753, http://dx.doi.org/10.1007/s00521-020-05532-z.
[19] M. Schnaubelt, A Comparison of Machine Learning Model Validation Schemes for Non-Stationary Time Series Data, FAU Discussion Papers in Economics. 11, 2019, http://hdl.h{and}le.net/10419/209136.
[20] M. Stone, Cross-validatory choice and assessment of statistical predictions, 1974.
[21] D.M. Allen, The relationship between variable selection and data agumentation and a method for prediction, Technometrics 16 (1974) 125–127, http://dx.doi.org/10.1080/00401706.1974.10489157.

[22] S. Geisser, The predictive sample reuse method with applications, J. Amer. Statist. Assoc. 70 (1975) 320–328, http://dx.doi.org/10.1080/01621459.1975.10479865.

[23] N. Nazareth, Y.Y.R. Reddy, Financial applications of machine learning: A literature review, Expert Syst. Appl. 219 (2023) http://dx.doi.org/10.1016/j.eswa.2023.119640.

[24] N. Rouf, M.B. Malik, T. Arif, S. Sharma, S. Singh, S. Aich, H.C. Kim, Stock market prediction using machine learning techniques: A decade survey on methodologies, recent developments, and future directions, Electronics (Basel) 10 (2021) http://dx.doi.org/10.3390/electronics10212717.

[25] Countries by GDP, 2022, https://www.populationu.com/gen/countries-by-gdp.

[26] S. Teixeira Zavadzki de Pauli, M. Kleina, W.H. Bonat, Comparing artificial neural network architectures for Brazilian stock market prediction, Ann. Data Sci. 7 (2020) 613–628, http://dx.doi.org/10.1007/s40745-020-00305-w.

[27] F. Yang, J. Chen, Y. Liu, Improved and optimized recurrent neural network based on PSO and its application in stock price prediction, Soft Comput. (2021) http://dx.doi.org/10.1007/s00500-021-06113-5.

[28] J.F. Kolen, S.C. Kremer, A Field Guide to Dynamical Recurrent Networks, John Wiley & Sons, 2001.

[29] S. Mehtab, J. Sen, A. Dutta, Stock price prediction using machine learning and LSTM-based deep learning models, in: Machine Learning and Metaheuristics Algorithms, and Applications: Second Symposium, SoMMA 2020, Chennai, India, 2021.

[30] D. Zhang, S. Lou, The application research of neural network and BP algorithm in stock price pattern classification and prediction, Future Gener. Comput. Syst. 115 (2021) 872–879, http://dx.doi.org/10.1016/j.future.2020.10.009.

[31] Q. Zhang, P. Zhang, F. Zhou, Intraday and interday features in the high-frequency data: Pre- and post-crisis evidence in China's stock market, Expert Syst. Appl. 209 (2022) http://dx.doi.org/10.1016/j.eswa.2022.118321.

[32] K. Kumar, M.T.U. Haider, Enhanced prediction of intra-day stock market using metaheuristic optimization on RNN–LSTM network, New Gener. Comput. 39 (2021) 231–272, http://dx.doi.org/10.1007/s00354-020-00104-0.

[33] A. Kurani, P. Doshi, A. Vakharia, M. Shah, A comprehensive comparative study of artificial neural network (ANN) and support vector machines (SVM) on stock forecasting, Ann. Data Sci. 10 (2023) 183–208, http://dx.doi.org/10.1007/s40745-021-00344-x.

[34] Y. Kara, M. Acar Boyacioglu, Ö.K. Baykan, Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul stock exchange, Expert Syst. Appl. 38 (2011) 5311–5319, http://dx.doi.org/10.1016/j.eswa.2010.10.027.

[35] B. Zou, H. Wang, H. Li, L. Li, Y. Zhao, Predicting stock index movement using twin support vector machine as an integral part of enterprise system, Syst. Res. Behav. Sci. 39 (2022) 428–439.

[36] M.H. Doroudyan, S.T.A. Niaki, Pattern recognition in financial surveillance with the ARMA-GARCH time series model using support vector machine, Expert Syst. Appl. 182 (2021) http://dx.doi.org/10.1016/j.eswa.2021.115334.

[37] R.K. Dash, T.N. Nguyen, K. Cengiz, A. Sharma, Fine-tuned support vector regression model for stock predictions, Neural Comput. Appl. (2021) http://dx.doi.org/10.1007/s00521-021-05842-w.

[38] A. Mahmoodi, L. Hashemi, M. Jasemi, S. Mehraban, J. Laliberté, R.C. Millar, A Developed Stock Price Forecasting Model using Support Vector Machine Combined with Metaheuristic Algorithms, OPSEARCH, 2022, http://dx.doi.org/10.1007/s12597-022-00608-x.

[39] X. Li, Y. Sun, Stock intelligent investment strategy based on support vector machine parameter optimization algorithm, Neural Comput. Appl. 32 (2020) 1765–1775, http://dx.doi.org/10.1007/s00521-019-04566-2.

[40] R. Gao, X. Zhang, H. Zhang, Q. Zhao, Y. Wang, Forecasting the overnight return direction of stock market index combining global market indices: A multiple-branch deep learning approach, Expert Syst. Appl. 194 (2022) http://dx.doi.org/10.1016/j.eswa.2022.116506.

[41] A. Kanwal, M.F. Lau, S.P.H. Ng, K.Y. Sim, S. Chandrasekaran, BiCuDNNLSTM-1dCNN — A hybrid deep learning-based predictive model for stock price prediction, Expert Syst. Appl. 202 (2022) http://dx.doi.org/10.1016/j.eswa.2022.117123.

[42] K. Chaudhari, A. Thakkar, Neural network systems with an integrated coefficient of variation-based feature selection for stock price and trend prediction, Expert Syst. Appl. (2023) 119527, http://dx.doi.org/10.1016/j.eswa.2023.119527.

[43] K.K. Yun, S.W. Yoon, D. Won, Prediction of stock price direction using a hybrid GA-XGBoost algorithm with a three-stage feature engineering process, Expert Syst. Appl. 186 (2021) http://dx.doi.org/10.1016/j.eswa.2021.115716.

[44] T. Kaczmarek, K. Perez, Building portfolios based on machine learning predictions, Econ. Res.-Ekon. Istraz. 35 (2022) 19–37, http://dx.doi.org/10.1080/1331677X.2021.1875865.

[45] E. Solares, V. De-León-Gómez, F.G. Salas, R. Díaz, A comprehensive decision support system for stock investment decisions, Expert Syst. Appl. 210 (2022) http://dx.doi.org/10.1016/j.eswa.2022.118485.

[46] A. Thakkar, K. Chaudhari, Information fusion-based genetic algorithm with long short-term memory for stock price and trend prediction, Appl. Soft Comput. 128 (2022) http://dx.doi.org/10.1016/j.asoc.2022.109428.

[47] Y. Chen, J. Wu, Z. Wu, China's commercial bank stock price prediction using a novel K-means-LSTM hybrid approach, Expert Syst. Appl. 202 (2022) 117370, http://dx.doi.org/10.1016/j.eswa.2022.117370.

[48] P. Lv, Y. Shu, J. Xu, Q. Wu, Modal decomposition-based hybrid model for stock index prediction, Expert Syst. Appl. 202 (2022) http://dx.doi.org/10.1016/j.eswa.2022.117252.

[49] C. Cortes, V. Vapnik, L. Saitta, Support-Vector Networks Editor, Kluwer Academic Publishers, 1995.

[50] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 1992.

[51] B. Schslkopf, C. Burges, V. Vapnik, Incorporating invariances in support vector learning machines, in: Artificial Neural Networks—ICANN 96, Springer Berlin Heidelberg, Bochum, Germany, 1996.

[52] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 1992, pp. 144–152.

[53] I. Guyon, B. Boser, V. Vapnik, Automatic capacity tuning of very large VC-dimension classifiers, in: Advances in Neural Information Processing Systems, 1993.

[54] V. Vapnik, S. Golowich, A. Smola, Support vector method for function approximation, regression estimation, and signal processing, neural information processing systems, 1997).

[55] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997).

[56] K. De, Learning with genetic algorithms: An overview, 1988.

[57] S. Mirjalili, Genetic Algorithm, in: Studies in Computational Intelligence, Springer Verlag, 2019, pp. 43–55, http://dx.doi.org/10.1007/978-3-319-93025-1_4.

[58] O. Kramer, O. Kramer, Genetic Algorithms, Springer, 2017.

[59] H. Chung, K. shik Shin, Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction, Neural Comput. Appl. 32 (2020) 7897–7914, http://dx.doi.org/10.1007/s00521-019-04236-3.

[60] S.K. Pal, P.P. Wang, Genetic Algorithms for Pattern Recognition, CRC Press, 1996.

[61] C. Bergmeir, J.M. Benítez, On the use of cross-validation for time series predictor evaluation, Inf. Sci. (N Y). 191 (2012) 192–213, http://dx.doi.org/10.1016/j.ins.2011.12.028.

[62] Y. Sun, S. Ding, Z. Zhang, W. Jia, An improved grid search algorithm to optimize SVR for prediction, Soft Comput. 25 (2021) 5633–5644, http://dx.doi.org/10.1007/s00500-020-05560-w.