Received: 13 January 2021 | Revised: 20 April 2021 | Accepted: 4 May 2021

DOI: 10.1002/int.22495

RESEARCH ARTICLE

WILEY

Hybrid evolutionary intelligent system and hybrid time series econometric model for stock price forecasting

Gourav Kumar[1] | **Uday Pratap Singh**[2] | **Sanjeev Jain**[3]

[1]Department of Computer Science & Engineering, Shri Mata Vaishno Devi University, Katra, Jammu and Kashmir, India

[2]Department of Mathematics, Shri Mata Vaishno Devi University, Katra, Jammu and Kashmir, India

[3]Department of Computer Science & Engineering, Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, Madhya Pradesh, India

Correspondence
Uday Pratap Singh, School of Mathematics, Shri Mata Vaishno Devi University, Katra, Jammu and Kashmir 182320, India.
Email: usinghiitg@gmail.com

Abstract

In this paper, a hybrid evolutionary intelligent system is proposed for dimensionality reduction and tuning the learnable parameters of artificial neural network (ANN) that can forecast the future (1-day-ahead) close price of the stock market using various technical indicators. Although the ANN possesses the ability to model highly uncertain and complex nonlinear data but the key challenge in ANN is tuning its parameters and minimizing the feature set that can be used in the input layer. The backpropagation approach used for training the ANN has a limitation to get trapped in local minima and overfitting the data. Motivated by this, we proposed a hybrid intelligent system for optimizing the initial parameters and for reducing the dimensions of the feature set. The proposed model is a combination of feature extraction technique, namely principal component analysis (PCA), particle swarm optimization (PSO), and Levenberg-Marquardt (LM) algorithm for training the feed-forward neural networks (FFNN). This paper also compares the forecasting efficiency of the proposed model with PSO-FFNN, regular FFNN, two standard benchmark approaches viz. GA and DE and another hybrid model obtained by the combination of PCA and a time series econometric model viz. auto-regressive distributed lag model. The presented technique has been tested to

predict the close price of three stock indices viz. Nifty 50, Sensex, and S&P 500. Simulation results indicate that the proposed model shows superior forecasting accuracy as compared with other methods.

KEYWORDS
auto-regressive distributed lag model, evolutionary neural network, particle swarm optimization, principal component analysis, stock price time series

1 | INTRODUCTION

Financial markets play a key role in the global economy because the economic developments of the countries are dependent on several financial activities.[1] The growth of the stock market plays an essential role in generating revenue and sustainable economic development of a country. Accordingly, stock market forecasting is one of the most emerging fields of research and challenging tasks for researchers and financial analysts. In the domain of financial market analysis, the movement of the stock index is governed by various factors like government policies, organization's growth, investor's expectations, economic conditions of the country, global economic conditions, political affairs, psychology of investors, and so forth. The positive aspect of the stock market is that it returns higher profit as compared with the rest of financial market and negative aspect of the market is that it contains huge risk, but intelligent decision helps in reducing most of the risks. So, forecasting stock market data is an important task for reducing the risk of the investor.

There are four kinds of procedures that are used for analyzing and forecasting the stock market, viz. fundamental analysis, technical analysis, traditional time series forecasting techniques, and soft computing techniques. The fundamental analysis involves analyzing various metrics of the companies that take into account the overall growth of a company, such as turnover, profit and loss, periodic reports, balance sheets, and so forth. Technical analysis is an approach for forecasting the future behavior of the stock market by analyzing the past behavior of stock price time series. Traditional statistical forecasting techniques include time series forecasting models such as autoregressive moving average (ARMA),[2] exponential smoothing (ES),[3] autoregressive integrated moving average (ARIMA),[4] autoregressive conditional heteroskedasticity (ARCH), and generalized autoregressive conditional heteroskedasticity (GARCH)[5] model the stock price time series as a linear combination of past stock values to forecast the future price.[6] Stock price time series data possess complex nonlinear behavior, highly noisy, dynamic, and chaotic in nature.[7] Hence, traditional time series forecasting techniques are unable to model the complex nonlinear and nonstationary behavior of stock markets. In the past decade, many soft computing approaches have been proposed for stock market price forecasting. Application of machine learning-based approaches in solving problems in different research and industry field such as the Internet of Things[8] spacecraft,[9] optical networks,[10] aerospace system,[11] image segmentation,[12,13] vision-based gait recognition,[14] and so forth have proven that such approaches perform better than traditional methods.

Soft computing is an emerging computation approach that mimics the thinking capability of the human brain to learn and generalize in a scenario of uncertainty and imprecision to solve complex real-world problems.[15] Artificial neural network (ANN), fuzzy logic (FL), genetic

algorithm (GA), and nature inspired algorithms (NIA) such as particle swarm optimization (PSO), ant colony optimization (ACO), artificial bee colony (ABC), bacterial foraging optimization (BFO), and so forth, are collectively used in soft computing. ANNs can be effectively applied to forecast the value of stock price time series as it does not incorporate complex model structure and strong model assumption in comparison with traditional linear and nonlinear time series forecasting models.[16] ANNs are self-organizing, data-oriented, and self-adaptive approaches with a capability to capture the nonlinearity of time series data without taking into account the prior assumption about data.[17] Due to the capability of ANNs to deal with nonlinear data and uncertainty, it is widely used for forecasting stock price time series. Feedforward neural networks (FFNN) and recurrent neural networks (RNN) are the two most frequently used ANNs approaches for stock market forecasting.[18] The biggest challenge in ANNs is defining their architecture in terms of the number of neurons in the input and hidden layers as well as tuning the connection weight and bias of the network so as to obtain an accurate result. Iteratively adjusting the weight and bias in ANNs is known as training or learning of neural networks. Training in ANNs is governed by minimizing the loss function like mean squared error between actual and predicted value averaged over all training samples. The second-order method, such as Levenberg–Marquardt (LM) method, can be successfully used to obtain a more accurate result. The LM method is a powerful optimization technique used in ANNs as it provides the procedure to speed-up the learning process and convergence of the networks. It uses backpropagation (BP) algorithm in which gradients are computed and propagated iteratively from the last layer of the network to the first layer until the error between actual and predicted value reached the minimum level or other stopping criteria such as the number of epochs are satisfied. Despite being the good performance of the LM method in some areas, it has two drawbacks: first, it gets trapped in a local minimum if the loss function is multimodal or/and nondifferentiable.[19] Second, it leads to the problem of vanishing gradient if networks have more than three layers.[20] A number of complex real-world optimization problems are described by complex solution space with multiple local optima. A nonconvex and multimodal objective function in optimization problem possess the number of valleys and ridges in which optimization techniques can stuck in local minima.[21] The vanishing gradient problem is a challenging issue in a multilayer FFNN trained by the BP algorithm when sigmoid activation functions are utilized in the network with several hidden layers. In sigmoid activation function, a huge change in the input leads to a small change in the output, and hence the derivative of the loss function gets close to zero. Since the BP algorithm uses the gradient to update the weights in the network in every iteration and vanishing gradient cause no change or very small change in weights during training.[22] These shortcomings of LMBP have motivated the researcher to use nature inspired and evolutionary methods to determine the optimal set of connection weights for neural networks.

Due to the fact that LM has the weakness to converge locally, it can be shown that optimization of network parameters such as weight and bias are strongly dependent on an initial random value. If the initial parameters are located in local search space, there is a chance that networks get trapped in local solution. The local convergence problem can be overcome by applying global search techniques for training neural networks. Considering the drawback of local convergence of LM, in this paper, we present the combination of PSO, which is a stochastic optimization technique and FFNN trained with Levenberg–Marquardt backpropagation (LMBP) algorithm for forecasting stock market indices. At the initial stage, PSO is utilized to find the initial value of weight and bias of the network to reduce the search space. After search space gets minimized, then the obtained weights and bias are assigned as initial parameters for the FFNN algorithm.

Selecting and reducing the dimension of feature space for stock market forecasting is a challenging task. Dimensionality can be reduced either by feature selection or feature extraction techniques.[23] Feature selection techniques minimize the feature space by excluding the irrelevant features and choosing the appropriate one. Various feature selection approaches have been used to enhance the interpretability of the model.[24,25] But the drawback of the feature selection approach is that we cannot gain any information from the eliminated features. On the other side, feature extraction methods involve mapping of higher dimensional feature space to a lower dimension, which provides more information with respect to the problem to being solved. Principal component analysis (PCA) is a feature extraction method that reduces dimensionality while retaining the most valuable part of all the features. In certain cases, input data has a wide range of values which reduces the efficiency of the FFNNs. Hence, to scale the data into a small range, the data transformation approach such as the min-max normalization technique is applied.

This paper proposes a hybrid evolutionary intelligent mechanism that can forecast the future (1-day-ahead) close price of the stock market by utilizing various technical indicators. The proposed model is a combination of PCA, PSO, and LM algorithm for training the FFNN. This paper also compares the forecasting capability of the proposed model with other hybrid models obtained by the combination of PCA and a time series econometric model such as autoregressive distributed lag model (ARDL), PCA and FFNN trained by GA,[26,27] PCA, and FFNN trained by differential evaluation (DE),[28,29] PSO-FFNN, and FFNN. The predictive ability of the proposed model has been judged by employing it to forecast three stock market indices viz. Nifty 50, Sensex, and S&P 500.

The rest of the paper is organized as follows: Section 2 presents the review of related work. Section 3 introduces the framework for creating the proposed models. Section 4 provides the description about experimental data. Section 5 describes the evaluation metrics and experimental setup. Section 6 demonstrates the experimental results and analysis. Finally, the conclusion is presented in Section 7.

2 | RELATED WORK

Stock market forecasting is a difficult task as there are many factors that have an impact on the stock market. Many researchers have used several statistical methods and soft computing techniques to forecast the stock market. Among the soft computing techniques, hybrid models are the most widely used and accepted methods.[18,30] In this section, we focus on related research work that has proposed an ANN-based hybrid model for forecasting of the stock market index of various emerging markets. Table 1 presents most of the related research in the domain of stock market forecasting using soft computing approaches.

3 | METHODOLOGY

This study presents a four-phase framework to create a hybrid evolutionary intelligent system to forecast stock price time series. The first phase is data pre-processing, where we employ a data transformation technique on technical indicators to map the data in a specified range, and then we apply PCA for dimensionality reduction to extract key features that are to be used as reduced feature set in the model. In the next phase, we apply PSO as a global search technique to obtain the initial weights and bias of FFNNs. In the third phase, we trained the FFNN using

TABLE 1 Relevant studies on stock market forecasting

Year	Authors	Method and application	Feature used	Financial market	Country
2019	Yang et al.[31]	Extreme learning method (ELM) a special case of single-layer FFNNs with Differential evolution (DE) for stock selection.	Fundamental factors	A-share Index	China
2019	Gocken et al.[32]	Harmony search (HS) optimization technique to tune the parameters of the neural network, Jordan recurrent neural network, Extreme learning machine (ELM) & Recurrent ELM.	Technical indicators	Borsa Istanbul stock exchange	Turkey
2018	Nadkarni et al.[33]	PCA with NeuroEvolution of Augmenting Topologies (NEAT) to generate the trading signal.	Technical indicators	S&P 500	USA
2018	Senapati et al.[34]	Adaline neural network with PSO to optimize initial parameters for predicting stock price.	Stock price time series	BSE	India
2018	Weng et al.[35]	PCA with Neural network regression ensemble, support vector regression ensemble, boosted regression tree & random forest regression for feature extraction and forecasting.	Technical indicators & Online data sources	Citi Group stock	USA
2018	Singh et al.[36]	Artificial neural network (ANN) with Modified Quaternion Firefly Algorithm to optimize initial weight and bias for currency exchange rate prediction.	Exchange rate time series	INR/EUR, INR/CAD and INR/PKR	India
2018	Ulke et al.[37]	Comparison of ANN with time series model auto-regressive distributed lag model (ARDL) for inflation forecasting.	Macro-economic data	US Inflation rate	USA
2017	Zhong et al.[38]	ANN with PCA for dimensionality reduction.	Financial & Economic factors	SPDR S&P 500 ETF	USA

TABLE 1 (Continued)

Year	Authors	Method and application	Feature used	Financial market	Country
2017	Pradeepkumar et al.[39]	Quantile regression neural network (QRNN) trained with PSO to forecast volatility from financial time series.	Financial time series	S&P 500 NSE India	USA, India
2016	Chiang et al.[40]	ANN with PSO to evolve initial weights and wavelet transformation for denoising the input signals.	Technical indicators	S&P 500 BSE Sensex	USA, India
2016	Qiu et al.[41]	ANN with GA to optimize initial weight for forecasting the stock market.	Technical indicators	NIKKEI225	Japan
2016	Inthachot et al.[42]	ANN with GA for feature selection to predict the stock index	Technical indicators	SET 50	Thailand
2016	Gocken et al.[43]	ANN with Harmony search (HS) and GA for feature selection.	technical Indicators	BIST100	Turkey
2015	Reza et al.[44]	ANN with BAT algorithm, namely Bat-neural network multiagent system (BNNMAS) to forecast the stock price.	Fundamental data and technical indicators	DAX	Germany
2015	Naranjo et al.[45]	Fuzzy logic with GA for trading system and capital management.	Technical Indicators	NASDAQ100 &EURO STOXX	USA, Eurozone
2014	Bisoi et al.[46]	Infinite impulse response (IIR) filter based dynamic NN (DNN), Unscented Kalman filter with DE to optimize parameters for forecasting stock indices.	Past stock prices and technical indicators	BSE, IBM, RIL & Oracle	India
2012	Asadi et al.[47]	Stepwise regression analysis (SRA), GA and FFNN for variable selection and stock market prediction.	Technical indicators	TAIEX, TEPIX, DJIA & NASDAQ Index	Taiwan, Iran, USA

LMBP algorithm[48] to tune the weights and bias obtained in the previous stage. In the last phase, we measure the performance of the proposed model using four performance metrics and then generate the forecasted stock prices. Figure 1 shows the general architecture of the proposed work. Each phase is explained in detail in the next section. This study also presents a hybrid econometric model-based approach for stock price time series forecasting.

3.1 | Data pre-processing

Pre-processing of data is an essential task in soft computing so as to enhance the performance of the proposed model because the accuracy and reliability of the model depend on the quality

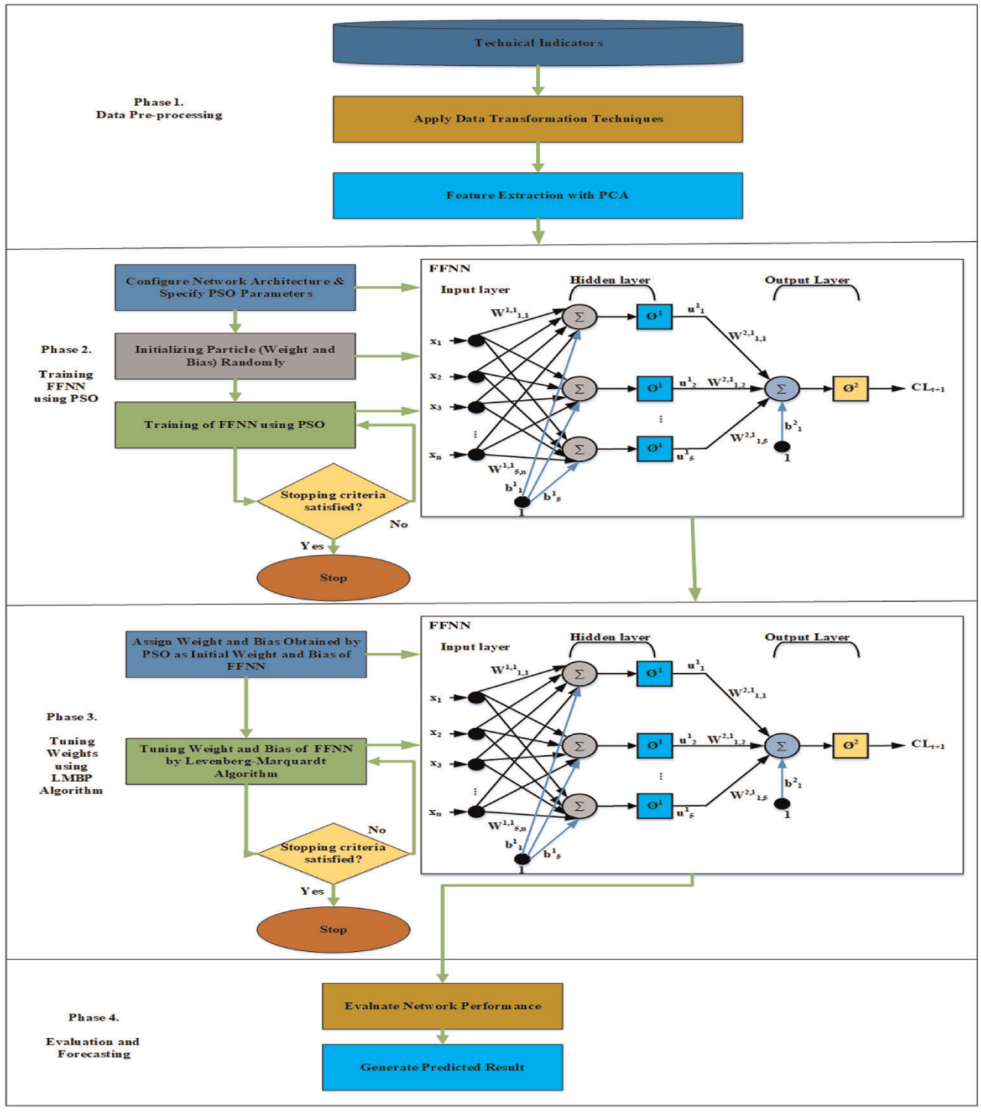

FIGURE 1 Flowchart of proposed work [Color figure can be viewed at wileyonlinelibrary.com]

of data.[49] Pre-processing techniques can be utilized for removing noise, detecting outliers, filling missing values, and scaling the data. In this paper we apply, data transformation and feature extraction technique as a pre-processing step.

3.1.1 | Data transformation

To minimize the variance among the values of selected features, data must be transformed to a certain range. Mapping of data from the original scale to another scale is an important task in many heuristics approaches, particularly when dealing with data-driven forecasting problems. A data transformation method must have the capability to retransform the scaled data into its original scale to obtain actual forecasted value. Data normalization is one of the most common transformation approach applied in stock price time series forecasting problems to prevent the learning of irrelevant patterns in data and speed up the learning process.[44,47]

There are various normalization techniques, viz. *Z*-score normalization, sigmoid normalization and Min-Max normalization.[32] We employ the Min-Max normalization technique. Suppose that $X_{\min}$ and $X_{\max}$ are the minimum and maximum values of an attribute X. Then Min-Max normalization technique transforms a value v of X to v^* in a new range $[X^*_{\min}, X^*_{\max}]$ by computing the following transformation function:

$$v^* = \left(\frac{v - X_{\min}}{X_{\max} - X_{\min}}\right)(X^*_{\max} - X^*_{\min}) + X^*_{\min}. \tag{1}$$

3.1.2 | Feature extraction by PCA

In soft computing, high dimensionality in the data set may result in the problem of overfitting, high computational complexity, and reduce the performance of the model. Hence, by reducing the dimensions of feature space, one can overcome these problems. Dimensionality reduction aims at representing the feature space with lower dimensions while preserving the hidden information present in the original data set. PCA is an unsupervised feature extraction procedure for extracting the uncorrelated features called principal components (PCs) from high dimensional feature space.[38]

PCA employs the orthogonal transformation to obtain a low dimensional representation of high dimensional data. The basic idea of PCA is the projection of space coordinates into new directions without changing the structure of original data, and the resultant PCs are a linear combination of actual attributes that include maximum possible original information.[50] This mapping is such that each PC is orthogonal to every other component and the first PC possesses maximum possible variance, and each succeeding component capture variance in decreasing order. The resultant transformation has the same number of dimensions as original data, but to utilize PCA for dimensionality reduction, only a few PCs are selected that capture maximum variance.

In PCA, the linear transformation of *M*-dimensional feature space represented as $X = [X_1, X_2, ..., X_M]$ to set of PCs, $Z^* = [Z_1^*, Z_2^*, ...,Z_M^*]$ is obtained. To meet this objective, the spectral-decomposition of variance–covariance matrix, Σ, is needed such that:

$$\Sigma = \text{Ω}\lambda\text{Ω}^T, \tag{2}$$

where, Σ is $M \times M$ variance–covariance matrix of normalized data, $Z = [Z_1, Z_2, ..., Z_M]$ obtained from X by Equation (2) and $\lambda = \{\lambda_{l}{}_{l=1}^{M}\}$ be the eigenvalues of the variance–covariance matrix Σ such that $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_M$ and $\boldsymbol{\omega_l^T} = (\omega_{l1}, \omega_{l2}..., \omega_{lM})$ denotes the eigenvectors with respect to eigenvalues, λ_l, $l = 1,2, ..., M$.

$$Z = X - \bar{X}, \tag{3}$$

where $\bar{X} = [\overline{X_1}, \overline{X_2}, ..., \overline{X_M}]$ is mean vector of X and Σ is obtained by Equation (3)

$$\sum = Z^T Z. \tag{4}$$

The PCs of standardized data can be written as

$$\begin{cases} Z_1^* = \boldsymbol{\omega_1}.\ Z^T = \omega_{11}Z_1 + \omega_{12}Z_2 + \cdots + \omega_{1M}Z_M \\ Z_2^* = \boldsymbol{\omega_2}.\ Z^T = \omega_{21}Z_1 + \omega_{22}Z_2 + \cdots + \omega_{2M}Z_M \\ \qquad\qquad \cdots\ \cdots\cdots \\ Z_M^* = \boldsymbol{\omega_M}.\ Z^T = \omega_{M1}Z_1 + \omega_{M2}Z_2 + \cdots + \omega_{MM}Z_M \end{cases} \tag{5}$$

Here, $\boldsymbol{\omega_i}$ can be estimated by maximizing $var(Z_i) = \boldsymbol{\omega_i^T}\Sigma\boldsymbol{\omega_i} = \lambda_i$ with the constraint that $\boldsymbol{\omega_i^T}\boldsymbol{\omega_i} = 1$ and $\boldsymbol{\omega_i^T}\boldsymbol{\omega_j} = 0$ such that different eigenvectors are orthogonal to each other. Thus, the variance of the first PC Z_1^* is largest eigenvalue, that is, λ_1 and so on. Also, the PCs are uncorrelated with each other.

To select the first P ($P < M$) PCs that capture maximum information, the proportion of variance explained by P components is used, which is defined as the sum of eigenvalues of first P components divided by the sum of eigenvalues of all the components that is, $\sum_{i=1}^{P}\lambda_i / \sum_{i=1}^{M}\lambda_i$. If the proportion of variance captured by the first P components is large enough (>85%), the dimensionality of data can be reduced from M to P without losing much information. The limitation about the usage of PCA is the manual selection of threshold value i.e. proportion of variance explained to select P components. However, there is no such threshold in autoencoder.

3.2 | ANNs

ANN is a soft computing model that imitates the learning capability of the human brain to deal with complex real-life problems on the basis of past experience. ANNs are massively parallel adaptive distributed networks made up of simple nonlinear processing units known as neurons that are purposed to abstract and model the basic functionality of the human brain to acquire some of its computational power.[51] ANN may be seen as a weighted directed graph where the nodes represent the artificial neurons and directed weighted edges constitute the connections among the neurons, as shown in Figure 2. The neurons in ANN have an efficiency of manipulating and memorizing experimental knowledge, and these neurons are arranged systematically into layer structure to create ANN. An ANN mimics the human brain with respect to two aspects: first, intelligence attained by the neural network through the learning process by utilizing historical data and second, strength (synaptic weight) of inter-neuron connection stores the knowledge gained while learning process. ANN is a connected system composed of a

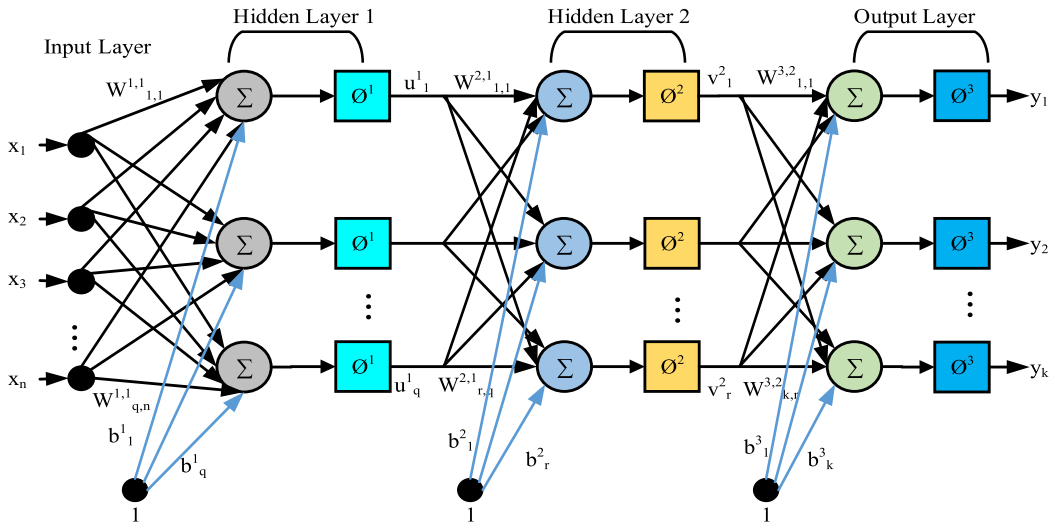

FIGURE 2 Three-layered architecture of ANN. ANN, artificial neural network [Color figure can be viewed at wileyonlinelibrary.com]

layered structure viz. input layer, hidden layer, and output layer. Inputs to the system are provided through the input layer. These inputs are propagated through one or more intermediate layers of hidden neurons through inter-neuron connection links. Each link has an associated weight which is multiplied with input, and the result is provided to an adder which aggregates the weighted input and adds a scalar bias to obtain net input. The net input is then given to the activation function, which generates the output of the network. In ANN, inter-neuron connection weights are optimized to enhance the performance of the network through the learning process. BP algorithm is mostly used as an error-correction technique to train ANN in which error between actual response and network response is computed and propagated in backward direction to modify weights.[52] Most of the training methods, such as LM, Newton method, gradient descent, conjugate gradient, and Quasi-Newton method and are gradient-based. Among the learning algorithms, a second-order method such as LMBP method can be successfully used to obtain a more accurate result. The LM method is a powerful optimization technique used in ANNs as it provides the procedure to speed-up the training process and convergence of the neural network. Figure 2 shows the three-layered architecture of ANN with n nodes in the input layer, q neurons in the first hidden layer, r neurons in the second hidden layer and k neurons in the output layer. Mathematically, ANN is represented by

$$y_t = \varnothing^3\left(\sum_{t=1}^{k} w_{t,m}^{3,2} v_m^2 + b_t^3\right), \tag{6}$$

$$v_m^2 = \varnothing^2\left(\sum_{i=1}^{q} w_{m,i}^{2,1} u_i^1 + b_m^2\right), \tag{7}$$

$$u_i^1 = \varnothing^1\left(\sum_{j=1}^{n} w_{i,j}^{1,1} x_j + b_i^1\right), \tag{8}$$

where y_t, v_m^2, u_i^1 are outputs, $\varnothing^3$, $\varnothing^2$, $\varnothing^1$ are activation functions, b_t^3, b_m^2, b_i^1 are the bias values of the output layer, second hidden layer and first hidden layer, respectively for $t = 1,2, ..., k$, $m = 1,2, ..., r$, and $i = 1,2, ..., q$. Inputs to the ANN, $x_j (j = 1,2, ..., n)$, are provided to the input layer, $w_{t,m}^{3,2}$ are weights between the second hidden layer and output layer, $w_{m,i}^{2,1}$ are weights between the first hidden layer and second hidden layer and $w_{i,j}^{1,1}$ are weights between the input layer and the first hidden layer. The activation functions are used to squash the output to a small range and to capture the nonlinear relationship present between input and output. The hyperbolic tangent (*Tanh*) transfer function produces values between −1 and 1 and is defined as $\mathrm{Tan}\,h(x) = (e^x - e^{-x})/(e^x + e^{-x})$. The log-sigmoid transfer (*logsig*) function defines as $logx(x) = (e^x\text{-}e^{\text{-}x})/(e^x + e^{\text{-}x})$ generate values between 0 and 1 and pure linear (*purelin*) activation function generates the neuron output by giving the same value passed to it and mathematically it can be written as $purelin(x) = x$.

3.3 | PSO

PSO is a population-based stochastic optimization method proposed by Kennedy and Eberhart in 1995.[53] PSO is inspired by the social behavior of bird flocking, fish schooling, and the swarm of insects. Suppose a scenario where swarms of birds are randomly looking for food in search space. Each bird in the swarm combines its own knowledge and neighbors intelligence while changing its velocity to search for the maximum amount of food. The idea of PSO resembles with the bird flocks where particle represent the bird and food represent the solution. The process starts with a population of random solutions called particles that form a swarm and search for the best solution by flying through the search space. Each particle acts as a candidate solution in *D*-dimensional search space which changes its velocity in accordance with its own position known as personal best, *Pbest* together with the position of other particle called as global best, *Gbest*. Figure 3 shows the principal architecture of PSO. Let P_t^i and V_t^i represent the position and velocity of particle *i*at iteration *t,* respectively. The velocity of *i*th particle in iteration $t + 1$ is updated by

$$V_{t+1}^i = \omega \times V_t^i + k_1 r_1 \times (Pbest^i - P_t^i) + k_2 r_2 \times (Gbest^i - P_t^i), \tag{9}$$

where, ω known as inertia coefficient, k_1 and k_2 called as personal coefficient and social coefficient, respectively, are constants, r_1 and r_2 are random numbers distributed uniformly in U(0,1). $Pbest^i$ is the best solution achieved so far by particle *i* and $Gbest^i$ is the best position attained so far by any other particle in the vicinity of particle *i.* Then the position of particle *i*, is updated by

$$P_{t+1}^i = P_t^i + V_{t+1}^i \tag{10}$$

for, $1 \le i \le D$.

3.4 | Hybrid evolutionary neural network

In this hybrid model, we use PSO to evolve the weights and bias between various layers in the FFNN. The process required to generate the weights and bias for the connection among neurons is shown in Figure 3. The steps involved in the process are explained below:

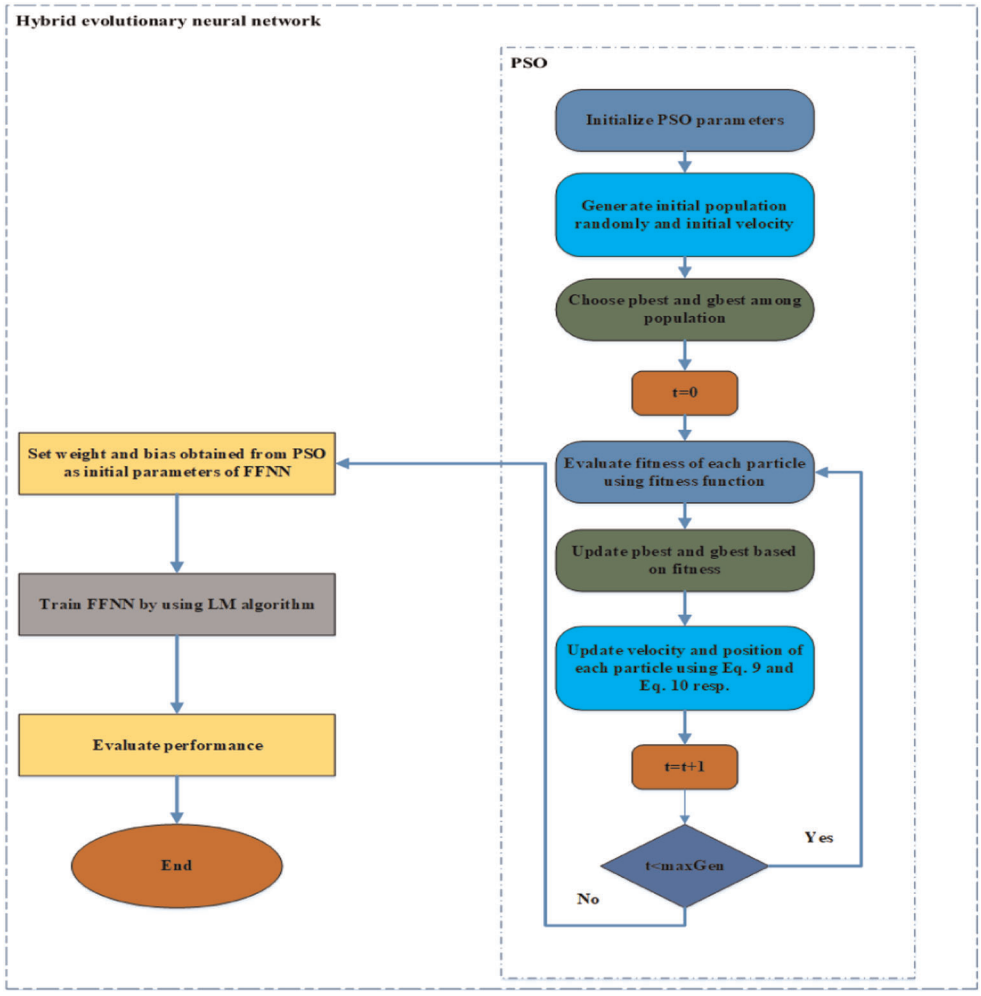

FIGURE 3 Architecture of hybrid evolutionary neural network. FNN, feed-forward neural network; PSO, particle swarm optimization [Color figure can be viewed at wileyonlinelibrary.com]

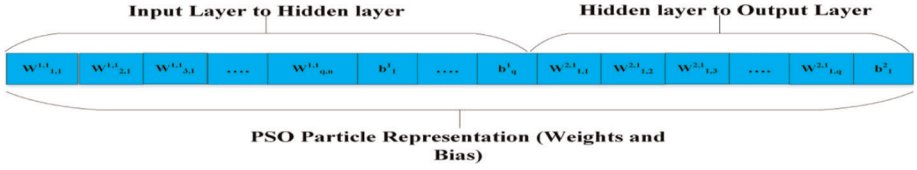

FIGURE 4 Particle representation in PSO. PSO, particle swarm optimization [Color figure can be viewed at wileyonlinelibrary.com]

(1) Encoding

Each particle in the PSO represents the weights and bias in different layers of FFNN. Thus dimensionality D of search space in PSO is equal to the number of parameters in FFNN to be optimized. The total number of weight and bias in this study is given

by $W = ((q_*n) + q) + ((k_*q) + k)$, where n, q, and k are the number of neurons in the input layer, hidden layer, and output layer, respectively. Figure 4 shows the representation of a single particle in PSO.

(2) Initialization

The initial weights and bias at $t = 0$ are generated randomly between $[-1\ 1]$. The initial velocity V_0^i of each particle is allocated the value of 0. To implement the PSO, various parameters such as inertia coefficient (ω), personal coefficient (k_1,), social coefficient (k_2), population size (P), the maximum number of generations (maxGen), and two random numbers (r_1 and r_2) distributed uniformly in U(0,1) should be properly determined in advance.

(3) Fitness evaluation

The fitness of each particle is computed by using root mean squared error calculated by Equation (14), where the predicted value is the output of ith training data obtained from FFNN by assigning the weight and bias generated at iteration t. Figure 5 shows the process of evaluating the fitness of each particle in the swarm.

(4) Update

Based on the fitness value, *Pbest* and *Gbest* are modified at each iteration, and the velocity and position of each particle at iteration $i + 1$ is updated using Equations (9) and (10), respectively. The process of updating *Pbest*, *Gbest*, velocity, and position of particle continues

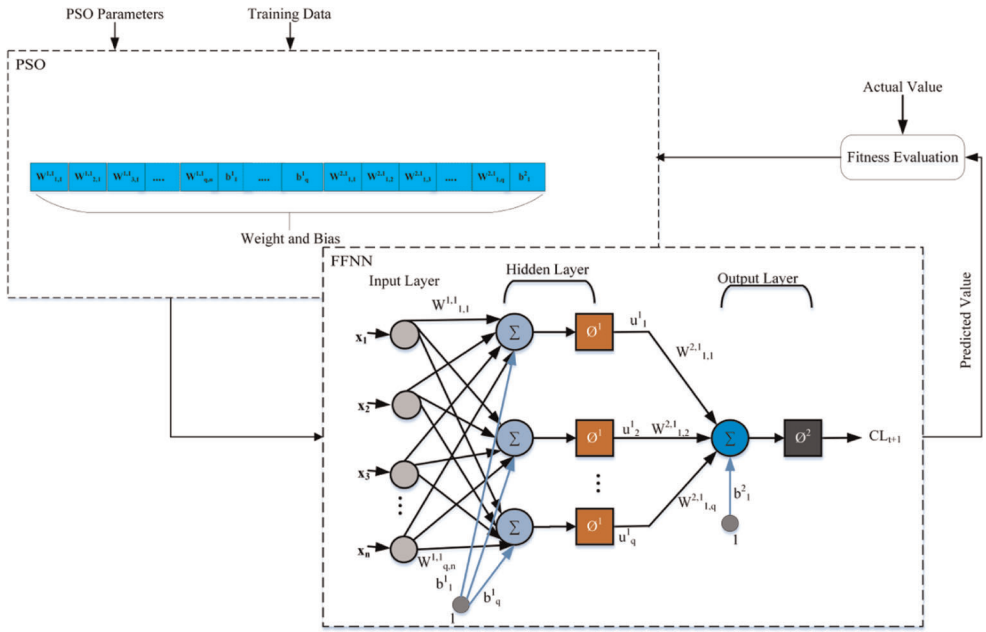

FIGURE 5 PSO-FFNN process for evaluating the fitness of each particle. PSO-FNN, particle swarm optimization-feed-forward neural network [Color figure can be viewed at wileyonlinelibrary.com]

until the count of iteration is equal to maximum generation number (maxGen), or tolerance reached to a minimum value.

The final global best *Gbest* particle is the optimal solution and is used as the initial weight and bias of FFNN, which is further tuned by LM algorithm.

3.5 | ARDL model

The ARDL model developed by Pesaran and Shin[54] is an approach to analyze time series data and to test cointegration among variables which are either $I(0)$ or $I(1)$. Variables are said to be co-integrated if there exists a long-term relationship among them. ARDL model has two components: AR (autoregressive) model, where the current value of time series is forecasted by using its own lagged value and DL (distributed lag) model, in which a regression equation, used to predict the current value of time series is based on both current value of time series together with the lagged value of exogenous variables. The general ARDL model for the response variable Y_t and a group of m independent variables $X_1, X_2, X_3, ..., X_m$ is represented as ARDL($p, q1, q2, q3, ..., qm$), in which p is the lag order of Y and $q1, q2, q3, ..., qm$ are the lag order of $X_1, X_2, X_3, ..., X_m$, respectively. ARDL ($p, q1, q2, ..., qm$) model is given as follows:

$$Y_t = \delta + \sum_{i=1}^{p}(\alpha_{0,i}Y_{t-i}) + \sum_{j=0}^{q1}(\beta_{0,j}X_{1,t-j}) + \sum_{l=0}^{q2}(\beta_{1,l}X_{2,t-l}) + \cdots + \sum_{k=0}^{qm}(\beta_{m,k}X_{m,t-k}) + \varepsilon_t, \quad (11)$$

where, Y_t indicates the time series, Y_{t-i} represent the ith period lagged value of Y_t, δ is intercept, $\alpha_{0,i}$ is the coefficient of the ith lagged value of Y_t,$\beta_{m,k}$ is the mth coefficient of kth lagged value of X_m, and ε_t is the error term.

Stationarity is a core concept for study the behavior of time series.[55] A time series data is said to be stationary if its mean, as well as variance, are independent of time. As a preliminary analysis, we require to verify if the time series data are stationary or they have unit root, that is, nonstationary. To check the stationarity in time series, we use the well-known augmented Dicky–Fuller (ADF) test[56] as it allows p lagged value of time series along with a constant, α as well as a linear time trend, ρt, which is given as follows:

$$\Delta Y_t = \alpha + \rho t + \gamma Y_{t-1} + \sum_{j=1}^{p}(\tau_t \Delta Y_{t-j}) + \varepsilon_t. \quad (12)$$

The null hypothesis of the ADF test says that there exists a unit root in Y_t and series is nonstationary. The alternative hypothesis says that there exist no unit root in Y_t and the series is stationary. With regard to testing the existence of unit root, the ADF test statistic is compared with the corresponding critical value. If the absolute value of the ADF test statistic is higher than that of the critical value, the alternative hypothesis is accepted, that is, time series is stationary; otherwise, the null hypothesis is accepted. The time series that is stationary at zero difference or level is known as $I(0)$. If the time series is not stationary at $I(0)$, then its first difference is used to achieve stationarity, called $I(1)$.

3.6 | Hybrid time series model

In multivariate regression analysis, one of the problems that arise in the least square estimation is the existence of multicollinearity, which occurs when an exact linear relationship exists among the explanatory variables as well as dependent variable.[57] In PCA, the eigenvectors obtained by the process are orthogonal, that is, they are independent of each other. Hence, the features obtained after applying PCA to original features are also independent of each other, that is, they are not highly correlated. In this paper, we use PCA to resolve the problem of multicollinearity in the ARDL model. To decrease the complexity of the model, PCA also reduces the dimensionality of the feature set. The resultant ARDL($p, q1, q2, ..., q\mathrm{n}$) model employing PCA is shown as follows:

$$CL_{t+1} = \delta + \sum_{i=0}^{p}(\alpha_{0,i}CL_{t-i}) + \sum_{j=0}^{q1}(\beta_{0,j}{PC^1}_{t-j}) + \sum_{k=0}^{q2}(\beta_{1,k}{PC^2}_{t-k}) + \cdots + \sum_{l=0}^{qn}(\beta_{n,l}{PC^n}_{t-l}) + \varepsilon_t. \tag{13}$$

In Equation (13), CL_t is the close price on a given trading day t, ${PC^n}_{t-l}$, is the lth lagged value of nth PCs where n<m and all other terms have the same meaning as in Equation (11).

4 | EXPERIMENTAL DATA DESCRIPTION

Deciding on the input features for the stock market forecasting problem is also a challenging task. To demonstrate and compare the forecasting ability of the proposed model, we have used historical data of three major stock indices viz. Nifty 50, Sensex, and S&P 500 for a period shown in Table 2 and compute various technical indicators which are applied as input to the proposed method. The data considered in this study is obtained from yahoo finance.[58–60] Each data set is divided into three subsets, viz. training set (70%), validation set (15%), and testing set (15%). The training data set is applied to learn the parameters of the proposed model. The testing data set is utilized to test and compare the forecasting capability of the proposed model, and the validation data set is employed to measure model generalization and to halt training when successive iterations stop improving the result.

The technical indicators for each trading day are computed from stock price time series data that includes open (OP), high (HI), low (LO), and close (CL) stock prices and trading volume (V). Some of these technical indicators are chosen from previous studies,[24,42,61] and others are selected in accordance with exploratory analysis and their popularity in relevance to technical analysis. A brief introduction about technical indicators and their mathematical formulations is given in Table 3. Different indicators are used in different market scenarios due to their inbuilt

TABLE 2 Description of stock market index data

Stock index name	From	To	No. of observations	Training set	Validation set	Testing set
Nifty 50	Jan 4, 2010	Sept 30, 2019	2417	1691	363	363
Sensex	Jan 2, 2010	Dec 31, 2019	2439	1707	366	366
S&P 500	Jan 2, 2010	Dec 30, 2019	2496	1748	374	374

TABLE 3 Technical indicators description and their formulation

Technical Indicator	Description	Formulation
OP_t	Open price at which a security is first traded on a given trading day t.	–
HI_t	Highest price on a given trading day t.	–
LO_t	Lowest price on a given trading day t.	–
CL_t	Price at which share is last traded on a given trading day t.	–
$SMA_t(n)$	Trend indicators smooth out the time series by reducing the noise from highly chaos stock price time series.	$SMA_t(n) = \frac{1}{n}\sum_{x=0}^{n-1} CL_{t-x}$
$EMA_t(n)$	Type of moving average, which assigns greater weight to most recent values of time series to make it respond more quickly to recent information.	$EMA_t(n) = \alpha(CL_t - EMA_{t-1}) + EMA_{t-1}$ $\alpha = \frac{2}{n+1}$
$MACD_t(n,k)$	Trend following momentum-based indicator which signifies the relationship between fast (n) and slow (k) moving averages.	$MACD_t = EMA_t(n) - EMA_t(k)$
ADX_t	Measure the strength of the trend in the stock market instead of trend direction. Crossover between DI^+ and DI^- indicates the trend in stock price. ADX > 25 indicates a strong trend, and ADX < 20 indicates a weak trend in the market.	$ADX_t = \frac{\lvert DI^+ - DI^- \rvert}{\lvert DI^+ + DI^- \rvert} \times 100$ $DI^+ = EMA(DM^+, n)/ATR(n)$ $DI^- = EMA(DM^-, n)/ATR(n)$
$CCI_t(n)$	Measures the divergences between stock price and moving averages over a certain period ago to determine the direction and strength of the trend and determine whether a stock is approaching a level of being overbought or oversold.	$CCI_t(n) = \frac{TP_t - SMA_t(TP,n)}{0.015 * \sum_{i=1}^{n} \lvert TP_{t-i+1} - SMA_t(TP,n) \rvert / n}$
$MTM_t(n)$	Measures the change in the price of shares over past n-trading days and indicate the rate at which price is changing to determine the trend in the market.	$MTM_t(n) = CL_t - CL_{t-n}$
$ROC_t(n)$	Momentum-based indicator which measures the percentage change in price over specific period of time for identifying overbought and oversold signals in the market.	$ROC_t(n) = \frac{CL_t - CL_{t-n}}{CL_{t-n}} \times 100$

(Continues)

TABLE 3 (Continued)

Technical Indicator	Description	Formulation
$RSI_t(n)$	Momentum indicator which measure the magnitude of relative gain to relative loss To identify the overbought and oversold signals in stock price. It is bounded between 0 and 100. When RSI is above 70, it gives the overbought signal, and RSI value below 30 gives oversold signals.	$RSI_t(n) = 100 - \frac{100}{1 + \frac{\sum_{i=0}^{n-1} UP_{t-i}/n}{\sum_{i=0}^{n-1} DN_{t-i}/n}}$
$TSI_t(n)$	Momentum indicator, which determines the oversold and overbought levels, indicates the reversal in trend direction using signal line crossover and depicts the strength of trend through divergence.	$TSI_t(n) = \frac{EMA(EMA(PC, n), k)}{EMA(EMA(\lvert PC\rvert, n), k)}$
$\%K_t(n)$	Depicts the position of current close price with respect to the previous lowest-low range over particular period of time. It is a range bounded indicator, and hence it also identifies oversold and overbought signals.	$\%K_t(n) = \frac{CL_t - LL_{t-(n-1)}}{HH_{t-(n-1)} - LL_{t-(n-1)}}$
$\%D_t(n)$	It is n-day simple moving average of stochastic %K indicator, which smooth out noise from it and is used as a signal line with %K indicator.	$\%D_t(n) = \frac{\sum_{i=0}^{n-1} \%K_{t-i}}{n}$
$\%R_t(n)$	Measures momentum in the stock market by comparing the position of current close with respect to previous highest-high range in look-back period say n.	$\%R_t(n) = \frac{HH_{t-(n-1)} - CL_t}{HH_{t-(n-1)} - LL_{t-(n-1)}}$
$ATR_t(n)$	Measures the strength of market volatility. It does not provide any information about the directional movement.	$ATR_t(n) = \frac{ATR_{t-1} * (n-1) + TR_t}{n}$
$UI_t(n)$	Volatility indicator that measure downside risk in stock market. A higher value of *UI* indicates higher downside risk, and a small value indicates lower risk in investment.	$UI_t(n) = \sqrt{\sum_{i=0}^{n-1} (R_t^i)^2}$ $R_t^i = (CL_t - \max(CL_{t-n}))/\max(CL_{t-n})$
ADI_t	Predict price trends and assess the forthcoming reversal in stock price time series. It is a volume-based indicator that gauges the supply and demand by finding whether trades are actually accumulating (buying) or distributing (selling).	$ADI_t = ADI_{t-1} + CMFV$ $CMFV = \left(\frac{CL_t - LO_t) - (HI_t - CL_t)}{(HI_t - LO_t)}\right) \times V_t$

TABLE 3 (Continued)

Technical Indicator	Description	Formulation
OBV_t	It measures the crowd sentiment that can predict upward and downward movement in the market. It uses the volume of shares traded to predict changes in stock prices.	$OBV_t = OBV_{t-1} + \begin{cases} V_t & \text{if } CL_t > CL_{t-1} \\ 0 & \text{if } CL_t = CL_{t-1} \\ V_t & \text{if } CL_t < CL_{t-1} \end{cases}$

Note: n is 10-day period times ago, $k > n$ is 15-day period ago, α is smoothing factor that decreases exponentially, DI^+ is a positive directional indicator and DI^- is negative directional indicators, $DM^+ = (HI_t - HI_{t-1})$ and $DM^- = (LO_{t-1} - LO_t)$ are positive and negative directional movements, $TP = (CL_t + HI_t + LO_t)/3$ is known as Typical price, $UP_t$is gain at time t and $DN_t$is loss at time t, $PC_t = CL_t - CL_{t-1}$ is a change in close price, HH_{t-n} and LL_{t-n} are the highest high price and lowest low price in last n trading day, $TR_t = \max[(HI_t - LO_t), |HI_t - CL_{t-1}|, |LO_t - CL_{t-1}|]$ is a true range, $CMFV$ is current money flow value, $\max(CL_{t-n})$ is a maximum close price in last n trading day and V_t, is the volume of shares traded on a given trading day t.

capability. Technical indicators[62,63] such as simple moving averages (*SMA*), exponential moving averages (*EMA*), moving averages convergence divergence (*MACD*), average directional index (*ADX*) and commodity channel index (*CCI*) are preferable for analyzing the trend in the market. Indicators like momentum (*MTM*), rate of change (*ROC*), relative strength index (*RSI*), true strength index (TSI_t), are used to measure the momentum, that is, speed at which price changes over a certain period of time, and to identify the oversold or overbought signals in a particular stock. Stochastic oscillators like stochastic %K (%*K*) and William %R (%*R*) are also momentum-based indicators used to identify overbought and oversold signals. %*D*is used as a signal line with %*K* to find trends in the market. Average true range (*ATR*) and ulcer index (*UI*) measure the market volatility. Average directional index (*ADI*) and on-balance volume (*OBV*) are volume-based indicators that use the volume of shares traded to measure the crowd sentiments.

5 | EVALUATION METRICS AND IMPLEMENTATION OF FORECASTING MODEL

5.1 | Evaluation metrics

To evaluate and compare the forecasting efficiency and robustness of the proposed model, we have used various metrics, including root mean square error (*RMSE*), mean absolute percentage error (*MAPE*), Theil's inequality coefficient (*U of Theil*), and average relative variance (ARV). The mathematical formulations of these metrics are defined as follows.

The *RMSE*, defined in Equation (14), is the square root of the mean of squared error. Error is defined as a deviation between the actual value and forecasted value. In this metric, square root is taken to bring the mean square error to the scale of data.[64] The *RMSE* value close to 0 indicates that the predicted value agrees with the actual value.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(A_i - P_i)^2}, \tag{14}$$

where A_i is the actual value and P_i is the forecasted value of ith observation of the test data set generated by the prediction model, and N is the total number of observations in the test data set or training data set.

The *MAPE*, Equation (15), evaluate the mean of absolute percentage error between the observed value and forecasted value. This metrics is easy to interpret as it gives error in terms of percentages. The required value of *MAPE* is 0, so closer the value of this measure to 0 shows that prediction performance is higher. Due to the absolute percentage error, this metric overcomes the issue of positive and negative values nullifying each other and hence avoiding misleading results.

$$MAPE = 100 \times \frac{1}{N}\sum_{i=1}^{N}\frac{|A_i - P_i|}{A_i}. \tag{15}$$

Theil's Inequility coefficient[65] given by Equation (16), determines how much a predicted time series is close to the actual time series. This metric relates the performance of the model with a random walk (RW) model.[47] If, $U\ \ of\ \ Theil > 1$, then the model has the worst performance compared to the RW model. If, $U\ \ of\ \ Theil = 1$, then the model's performance is equal to the RW model. If, $U\ \ of\ \ Theil < 1$, then the model performs better than the RW model. Hence, the ideal value of $U\ \ of\ \ Theil$ tends to 0.

$$UofTheil = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(A_i - P_i)^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(A_i)^2} + \sqrt{\frac{1}{N}\sum_{i=1}^{N}(P_i)^2}}. \tag{16}$$

The average relative variance (ARV) metrics is defined by Equation (17). This measure relates the model performance with the mean considering as forecasted value.[47] The performance of model is acceptable if, $ARV < 1$, and closer the value to 0, means that the model tends to be perfect.

$$ARV = \frac{\sum_{i=1}^{N}(P_i - A_i)^2}{\sum_{i=1}^{N}(P_i - \bar{A})^2}. \tag{17}$$

5.2 | Implementation of the proposed model

The simulation is performed on a system having a 2.10 GHz i3 processor with 4GB RAM and the proposed models are implemented in Matlab 2019b and EViews 10.

5.2.1 | Implementation of hybrid evolutionary neural network

In the first stage, we perform the technical analysis by using OHLC (open, high low close) data and volume of shares traded to compute the technical indicators. In this paper, a set of 16 technical indicators along with four daily prices are used to generate a feature set of twenty indicators which are passed to data pre-processing module. These technical indicators are computed by using the TA python library.[66]

In the data preprocessing step, we employ the Min-Max normalization technique to transform the original data in the range [−1 1] to maintain the actual relationship among

the original features set. Then, we use the PCA to generate the 20 PCs from the normalized data set obtained in the previous step. For the purpose of reducing the dimensionality of the feature set, we select the first nine PCs that explain 99% of the variation in the data set.

In the second stage, we apply PSO to evolve the weight and bias of FFNN by using an evolutionary neural network. To obtain the optimized weights with minimum error, the network structure comprises the number of hidden layers, number of neurons in each layer, type of activation function and different parameters of PSO have been investigated. In the next stage, we apply the LMBP algorithm to train the FFNN using the weights and bias obtained in the previous stage. The optimized features of the proposed model are shown in Table 4. The dimension of each particle in the population is equal to the number of network's parameters to be optimized. Finally, we evaluate the model and obtained the one-day ahead stock prices (CL_{t+1}).

5.2.2 | Implementation of the hybrid econometric model

To implement the ARDL model, it is important to verify the stationarity of the variables at zero difference or level, $I(0)$ and first difference, $I(1)$, as the model is not applicable at second

TABLE 4 Tuned parameters of the models

Features	Parameters	Value
PSO parameters	Population size	50
	Inertia coefficient, ω	0.9
	Personal coefficient, k_1	1.2
	Social coefficient, k_2	1.2
	maxGen	100
GA parameters	Selection	Roulette wheel
	Crossover	Blend crossover (Probability = 0.7)
	Mutation	Random mutation (probability = 0.3)
	Population size	50
	maxGen	100
DE parameters	Crossover probability	0.2
	Scaling factor	[0.2 0.8]
	Population size	50
	maxGen	100
Network topology	20-5-1 (Without PCA)	
(input-hidden-output)	9-5-1 (With PCA)	
Activation function	Tanh-purelin	
Learning rate	0.01	

Abbreviations: DE, differential evaluation; GA, genetic algorithm; PCA, principal component analysis; PSO, particle swarm optimization.

difference, $I(2)$. To deal with the problems of multicollinearity, we first apply the PCA on selected 20 features and also, to reduce the complexity of the model we select first 9 PCs that explain 99% variability in the data set. Then, we apply the ADF test on selected PCs to confirm the stationarity of the variables. After testing the stationarity of the variables, we apply the obtained PCs to the ARDL model to obtain the resultant model (PCA-ARDL). The optimum lags of dependent as well as explanatory variables in ARDL are chosen as per the minimum Akaike information criteria (AIC).[67]

6 | RESULTS AND DISCUSSION

In the data preprocessing stage, using PCA, the 20 PCs of the 20 features are generated and the resulting number of PCs that explained about 90% variability in the data set is 9. Table 5 shows the variance and cumulative percentage of variance capture by the first nine PCs for three data sets. The PCs are ordered on the basis of their importance (variance explained). It can be observed from Table 5 that the first nine PCs explain about 99% variability of data, and the resultant nine components are fed as input to the evolutionary neural network. Hence, the dimensionality of the original feature set is reduced from 20 features to 9 features, and the resultant network structure is also reduced from 20-5-1 to 9-5-1, which reduces the number of parameters in FFNN. Thus PCA has successfully reduced the complexity and computation cost of the network.

Table 6 shows the t statistic and p value for the ADF unit root test for the three data sets. From the result of ADF test, it can be seen that for the three data sets only two variables viz. principal component 1 (PC1) and the close price (CL) are nonstationary at the level and stationary at the difference, and the rest of the variables are stationary at level. Hence the variables are not integrated at second difference; therefore, the ARDL model has been applied successfully. The resultant ARDL model selected on the basis of AIC is ARDL (1,0,0,0,2,1,0,1,0,0), ARDL(4,4,0,2,1,0,0,4,3,0), and ARDL(1,0,0,0,0,0,0,1,1,1) for Nifty 50, Sensex, and S&P 500, respectively.

To evaluate the forecasting ability of the proposed model named PCA-PSO-FFNN, we have computed various metrics defined in Section 5.1 and compare its performance with FFNN, PSO-FFNN, PCA-ARDL, PCA-DE-FFNN, and PCA-GA-FFNN. The performance of the forecasting results of the proposed model using a test data set for Nifty 50, Sensex, and S&P 500 are depicted in Table 7. It is observed from the experimental results that PCA-PSO-FFNN has successfully improved over other forecasting techniques in terms of accuracy and outperforms other techniques considered in this study. From Theil's U metric, it is observed that PCA-PSO-FFNN is not a random walk (RW) model for three data sets and performs better than the RW model. The ARV metric shows that PSO-FFNN performs better than FFNN and PCA-ARDL model. However, the minor difference between PCA-PSO-FFNN and PSO-FFNN is due to the number of PCs selected, which may be improved if we select more features.

Figure 6 visualizes the comparison of RMSE performance of the proposed model, that is, PCA-PSO-FFNN, with other models for Nifty 50, Sensex, and S&P 500. It is observed from the figure that the RMSE of PCA-PSO-FFNN has decreased significantly compared with PCA-GA-FFNN, PCA-DE-FFNN, PSO-FFNN, regular FFNN, and PCA-ARDL for three data sets.

The one-day ahead forecasting of the closing price of Nifty 50, Sensex, and S&P 500 for test data set using proposed model are shown in Figures 7–9, respectively. The plots also display the comparison of the predictive ability of the proposed model with other models considered in this

TABLE 5 The results of PCA

Principal component	Nifty 50		Sensex		S&P 500	
	Variance (%)	Cumulative variance (%)	Variance (%)	Cumulative variance (%)	Variance (%)	Cumulative variance (%)
1	51.32237	51.32237	51.28061	51.28061	51.62824	51.62824
2	30.47538	81.79776	30.27916	81.55977	31.05269	82.68092
3	7.594664	89.39242	7.247534	88.8073	8.280542	90.96146
4	3.659191	93.05161	3.606305	92.41361	3.406657	94.36812
5	2.977108	96.02872	3.352967	95.76657	2.902535	97.27066
6	1.211185	97.2399	1.402963	97.16954	0.736229	98.00689
7	0.822062	98.06197	0.941563	98.1111	0.5301	98.53699
8	0.615494	98.67746	0.704611	98.81571	0.399366	98.93635
9	0.446139	99.1236	0.544038	99.35975	0.341479	99.27783

Abbreviation: PCA, principal component analysis.

TABLE 6 Results of ADF unit root test

	Levels		Differences	
Series	***t* Statistic**	***p* Value**	***t* Statistic**	***p* Value**
Nifty 50				
PC1	−1.63	0.47	−34.56	(0.00)***
PC2	−13.82	(0.00)***	–	–
PC3	−10-94	(0.00)***	–	–
PC4	−10.07	(0.00)***	–	–
PC5	−6.68	(0.00)***	–	–
PC6	−5.69	(0.00)***	–	–
PC7	−6.95	(0.00)***	–	–
PC8	−16.60	(0.00)***	–	–
PC9	−16.83	(0.00)***	–	–
CL	−0.37	0.91	−45.71	(0.00)***
Sensex				
PC1	−1.78	0.3926	−42.65	(0.00)***
PC2	−12.92	(0.00)***	–	–
PC3	−13.01	(0.00)***	–	–
PC4	−9.28	(0.00)***	–	–
PC5	−5.75	(0.00)***	–	–
PC6	−4.58	(0.00)***	–	–
PC7	−5.36	(0.00)***	–	–
PC8	−10.53	(0.00)***	–	–
PC9	−10.89	(0.00)***	–	–
CL	0.02	0.96	−46.52	(0.00)***
S&P 500				
PC1	−1.51	0.53	−50.07	(0.00)***
PC2	−13.71	(0.00)***	–	–
PC3	−11.83	(0.00)***	–	–
PC4	−6-68	(0.00)***	–	–
PC5	−9.31	(0.00)***	–	–
PC6	−11.79	(0.00)***	–	–
PC7	−10.79	(0.00)***	–	–
PC8	−4.34	(0.00)***	–	–
CL	0.09	0.96	−42.50	(0.00)***

Abbreviation: ADF, augmented Dicky–Fuller.

***Statistical significance at 1% level.

TABLE 7 Comparison of performance of the proposed model with other models

Stock market	Model	RMSE	MAPE	Theil's U	ARV
Nifty 50	PCA-ARDL	4.55E−02	2.80E−03	7.70E−03	2.80E−03
	FFNN	3.83E−02	2.73E−04	6.50E−03	6.93E−04
	PSO-FFNN	6.80E−03	2.72E−04	1.20E−03	6.13E−04
	PCA-DE-FFNN	2.07E−02	5.03E−04	1.73E−02	8.30E−04
	PCA-GA-FFNN	2.08E−02	4.92E−04	1.74E−02	9.34E−04
	PCA-PSO-FFNN	**6.00E−03**	**4.41E−05**	**1.00E−03**	**7.44E−04**
Sensex	PCA-ARDL	1.09E−02	5.32E−05	1.50E−03	7.04E−04
	FFNN	4.90E−03	7.72E−05	6.56E−04	9.27E−04
	PSO-FFNN	4.10E−03	5.81E−05	5.41E−04	8.64E−04
	PCA-DE-FFNN	2.00E−02	3.31E−04	1.72E−02	9.79E−04
	PCA-GA-FFNN	1.96E−02	2.24E−04	1.68E−02	9.32E−04
	PCA-PSO-FFNN	**1.40E−03**	**3.07E−04**	**1.88E−04**	**1.00E−04**
S&P 500	PCA-ARDL	4.50E−03	2.32E−04	8.94E−04	4.95E−04
	FFNN	1.65E−02	1.19E−04	3.30E−03	4.96E−04
	PSO-FFNN	4.60E−03	1.20E−04	9.19E−04	4.90E−04
	PCA-DE-FFNN	1.67E−02	1.33E−04	1.57E−02	5.58E−04
	PCA-GA-FFNN	1.68E−02	3.32E−04	1.53E−04	5.66E−04
	PCA-PSO-FFNN	**9.34E−04**	**1.16E−04**	**1.87E−04**	**4.20E−04**

Note: Bold values signifies that the proposed model i.e. PCA-PSO-FFNN shows the superior forecasting performance in comparison to other techniques considered in this study in terms of four performance metrics viz. RMSE, MAPE, Theil'U and ARV.

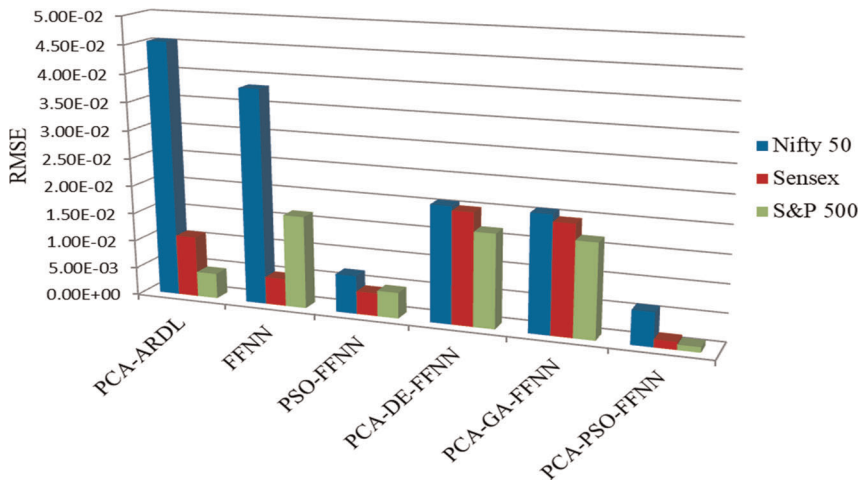

FIGURE 6 Performance comparison of proposed model with other models [Color figure can be viewed at wileyonlinelibrary.com]

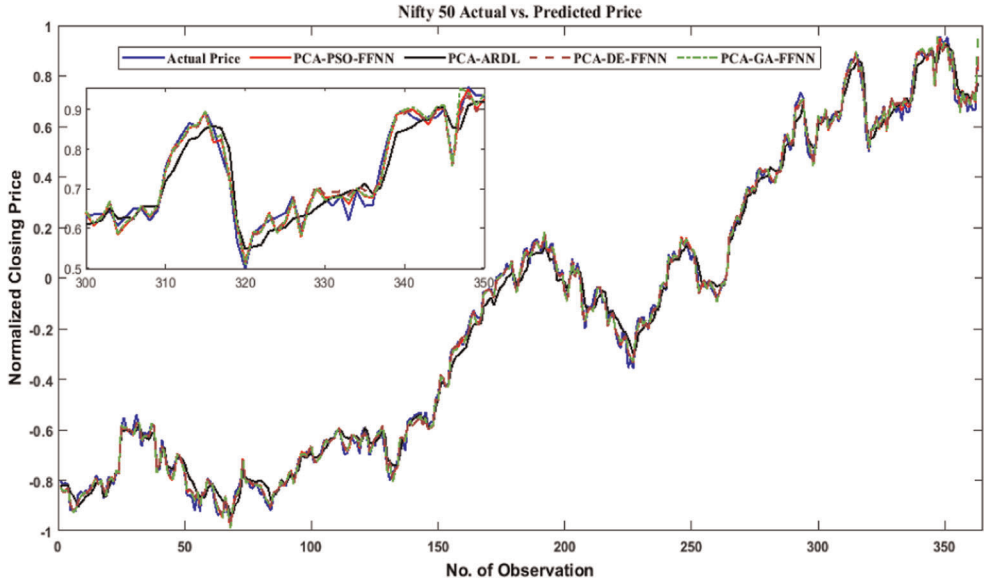

FIGURE 7 Forecasting of daily close price of Nifty 50 [Color figure can be viewed at wileyonlinelibrary.com]

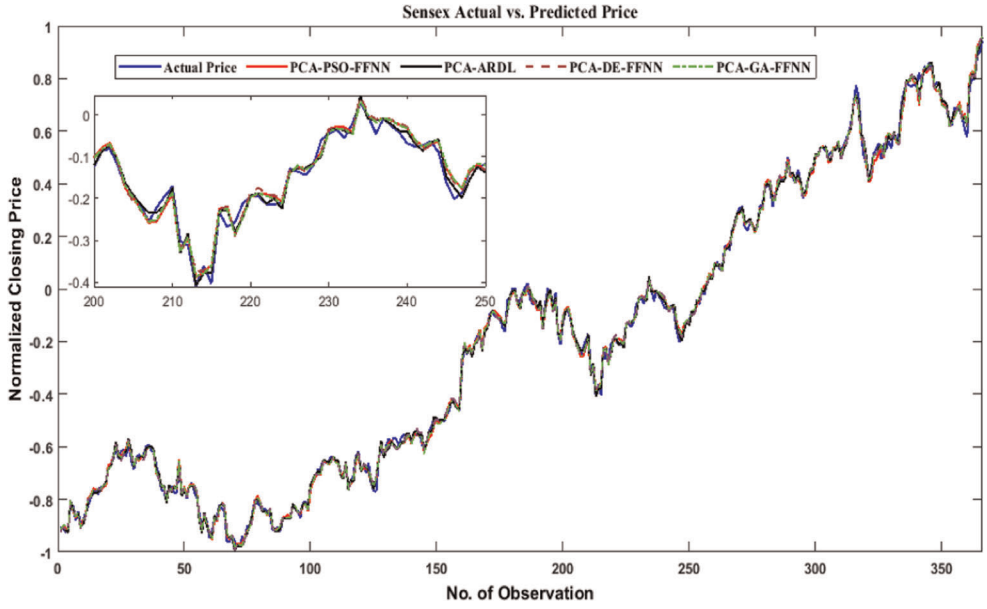

FIGURE 8 Forecasting of daily close price of Sensex [Color figure can be viewed at wileyonlinelibrary.com]

study. It is observed from the plot of forecasted price and the actual price that the proposed model shows the superior forecasting performance.

Figures 10–12 show the convergence of PSO, GA, and DE and also visualize the change of best fitness values over iterations for Nifty 50, Sensex, and S&P 500, respectively for PCA-PSO-

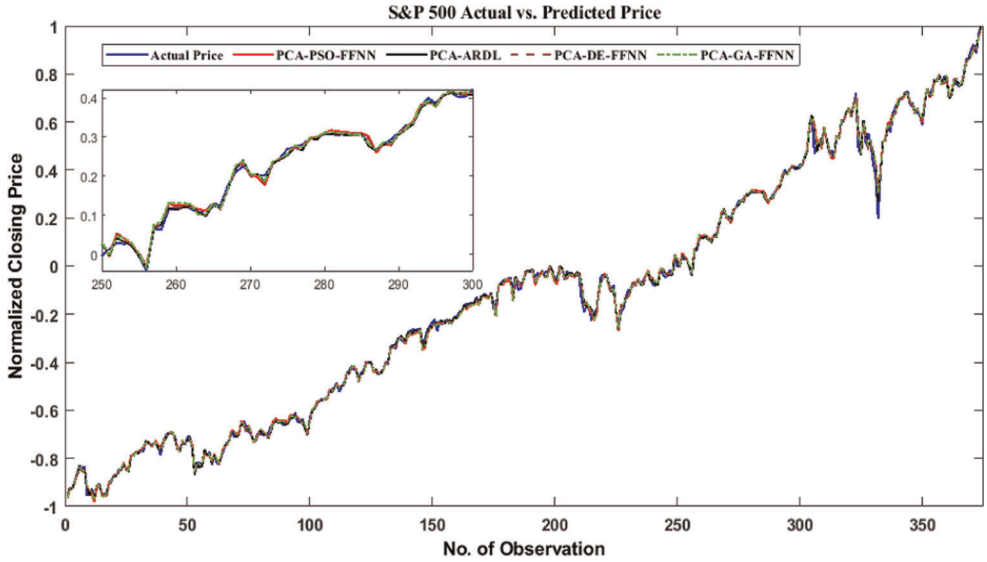

FIGURE 9 Forecasting of daily close price of S&P 500 [Color figure can be viewed at wileyonlinelibrary.com]

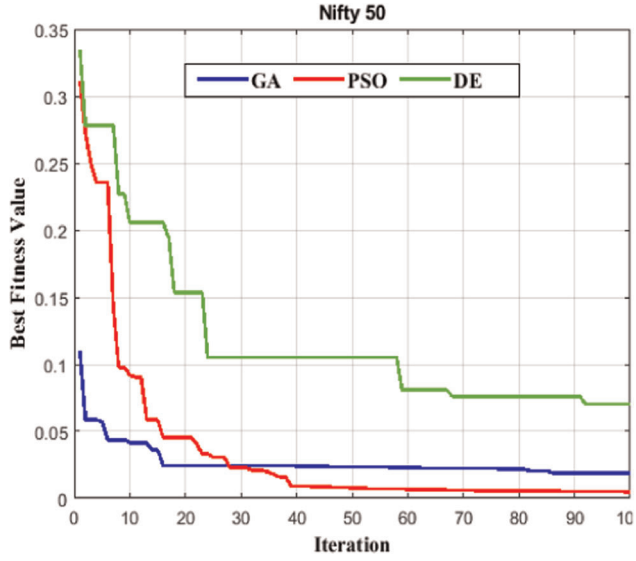

FIGURE 10 Fitness value over iterations for PSO, GA, and DE for Nifty 50. DE, differential evaluation; GA, genetic algorithm; PSO, particle swarm optimization [Color figure can be viewed at wileyonlinelibrary.com]

FFNN, PCA-GA-FFNN, and PCA-DE-FFNN. It is evident that PSO gives the minimum fitness values, that is, best fitness for three data sets and shows the superiority of PSO in comparison with GA and DE. By analyzing the best fitness value obtained by PSO in Figures 10–12, we can judge that after the iteration 40 in each data set there is very little change in MSE and Figures 13–15 show how the MSE is further improved by LMBP. Further, the LMBP algorithm starts training the network by using the optimized weights and bias obtained by PSO.

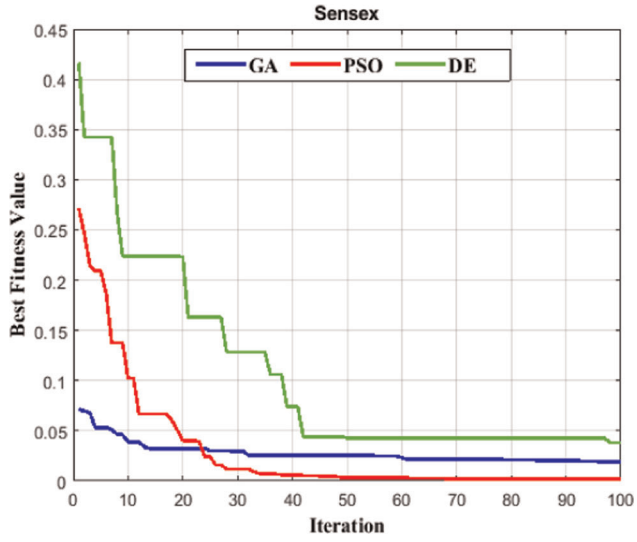

FIGURE 11 Fitness value over iterations for PSO, GA, and DE for Sensex. DE, differential evaluation; GA, genetic algorithm; PSO, particle swarm optimization [Color figure can be viewed at wileyonlinelibrary.com]

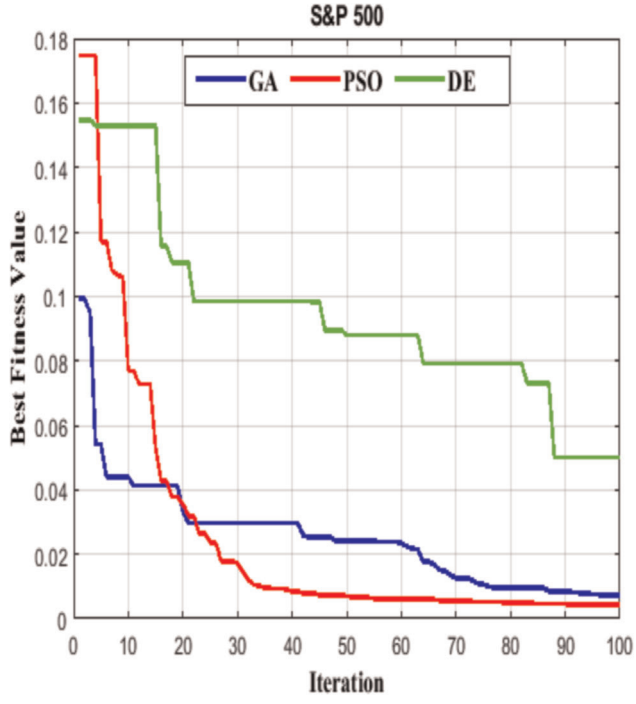

FIGURE 12 Fitness value over iterations for PSO, GA, and DE for S&P 500. DE, differential evaluation; GA, genetic algorithm; PSO, particle swarm optimization [Color figure can be viewed at wileyonlinelibrary.com]

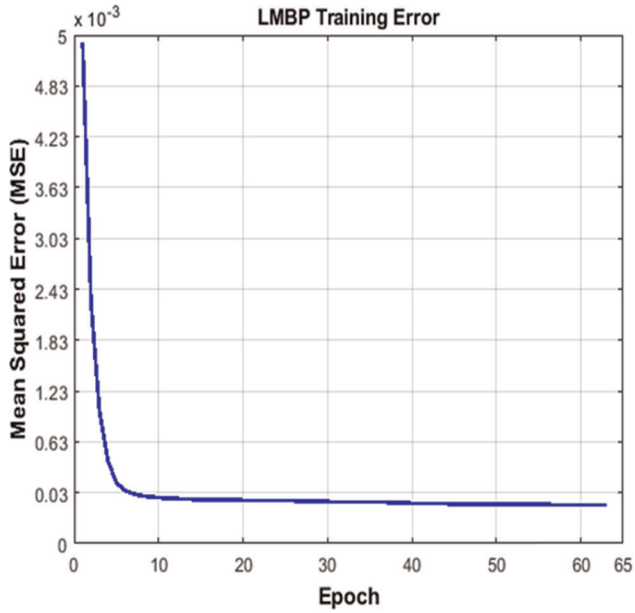

FIGURE 13 MSE of LMBP over training epochs for Nifty 50 [Color figure can be viewed at wileyonlinelibrary.com]

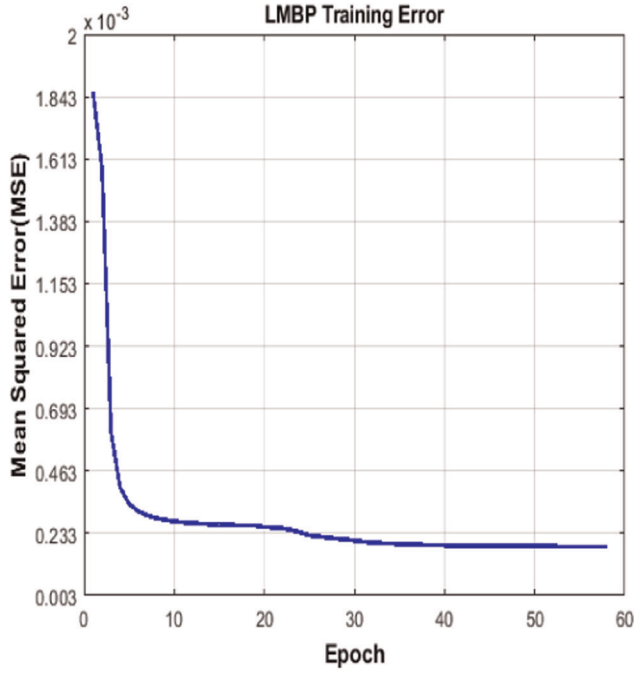

FIGURE 14 MSE of LMBP over training epochs for Sensex [Color figure can be viewed at wileyonlinelibrary.com]

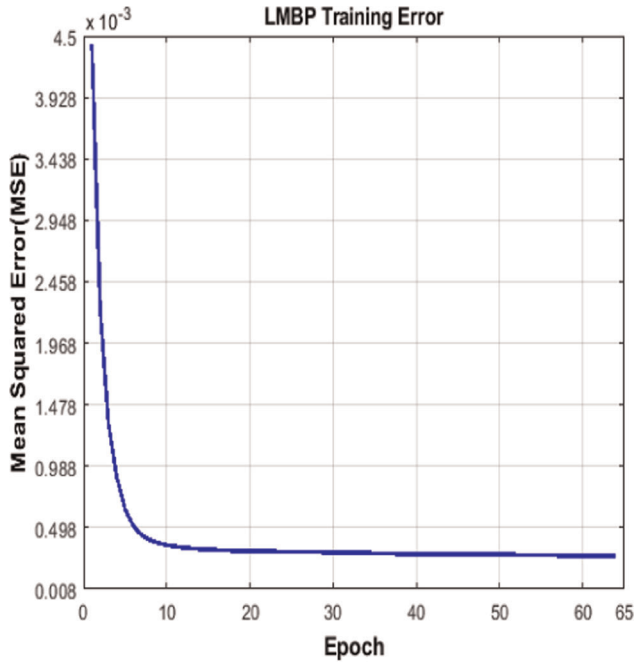

FIGURE 15 MSE of LMBP over training epochs for S&P 500 [Color figure can be viewed at wileyonlinelibrary.com]

Figures 13–15 depict the visualization of how the MSE is further improved in comparison to Figures 10–12, respectively; by LMBP algorithm in each training epoch after applying the PSO for obtaining the initial weights and bias of FFNN for Nifty 50, Sensex, and S&P 500, respectively.

6.1 | Complexity analysis

The computational complexity of the proposed model can be defined by the number of multiplication operation (ignoring the computation of nonlinear activation function) required by the proposed model, which is given as a function of the number of input instances (N), input features (n), hidden neurons (q), and output neurons (k) and size of Jacobian matrix required for updating weights in LM training algorithm in the backward pass. In FFNN, a single forward pass of the LMBP algorithm required $O(n \times q + q \times k)$ multiplication operations and in backward pass computational complexity depend upon the dimension of the Jacobian matrix, which is given by $N \times W$, where W is the number of parameters, tha is, weights and bias of FFNN. The number of parameters in FFNN is given as $W = ((q \times n) + q) + ((k \times q) + k)$. The number of multiplication operations for a single instance and the total number of instances of Nifty 50 training data set and dimension of the Jacobian matrix from input to output layer required by the proposed model, that is, PCA-PSO-FFNN and regular FFNN with one hidden layer is enlisted in Table 8. It is analyzed from the table that the proposed approach requires a less number of multiplication operations due to a less number of input features and number of weights elements in the Jacobian matrix. The computational cost of PSO is of the order of

TABLE 8 Computational complexity of the proposed model and regular FFNN

Model	N	n	q	k	$T = n \times q + q \times k$	Total number of multiplication operations ($N \times T$)	$W = ((q \times n) + q) + ((k \times q) + k)$.	$N \times W$
FFNN	1691	20	5	1	105	177,555	111	187,701
PCA-PSO-FFNN	1691	12	5	1	65	109,915	66	111,606

$O(G \times P \times b)$, where G is the maximum number of generations, P is the population size, and b is the number of bits used to encode element in a particle. Apart from this, fitness evaluation of each particle requires the execution of forward pass of backpropagation that demands $O(n \times k \times N \times E/2)$, operations, where E is the number of epochs, whereas LMBP without PSO requires $O(n \times k \times N \times E)$ operations.

6.2 | Statistical analysis of results

In this study, Friedman's rank test is applied to determine the significant difference among the performance of six techniques. Friedman's rank test[68] is a nonparametric statistical test alternative to one-way analysis of variance with repetitive measures. The aim of the test is to determine whether there is a significant difference among the models under study over given data sets. The test computes the ranks of the models for each data set, that is, rank 1 is assigned to the best performing model and rank 2 is assigned to second best and so on. Let r_{ij} be the rank of ith of p models on jth of q data sets. Friedman's test computes the average ranks of the models, $R_j = 1/q\sum_{i=1}^{q} r_{ij}$. The null hypothesis proposes that all the models have the same performance and therefore, their ranks are similar. Friedman's statistics is distributed by χ_F^2 with $p - 1$ degree of freedom and given as

$$\chi_F^2 = \frac{q}{p(p+1)}\left[\sum_j R_j^2 + \frac{p(p+1)^2}{4}\right]. \tag{18}$$

The critical value of Friedman's test statistic is approximated by χ^2 distribution when $q > 10$ and/or $p > 5$.[69] The calculated value for Friedman's test statistics is 10.05, whereas the critical value at $\alpha = 0.10$ is $\chi^2(5) = 9.24$, and hence the null hypothesis is rejected. Table 9 shows the average rank position of models and p value computed by Friedman's test. From Table 9, it is clear that the proposed model, that is, PCA-PSO-FFNN

TABLE 9 Average rank position of the models and p-value by Friedman's test

Rank	1st	2nd	3rd	4th	5th	6th	p Value
Model	PCA-PSO-FFNN	PSO-FFNN	FFNN	PCA-ARDL	PCA-DE-FFNN	PCA-GE-FFNN	0.0739
Avg. rank	1	2.33	4	4	4.67	5	–

performs significantly better in terms of average rank than other models considered in this study.

7 | CONCLUSION

This study proposes a hybrid evolutionary intelligent system by combining the PCA method for dimensionality reduction, PSO for evolving the initial weight and bias of FFNN and LMBP algorithm to train the FFNN to forecast the daily stock prices of three major stock indices. This study also compares the result of the proposed model with another hybrid model obtained by combining PCA with ARDL, PSO-FFNN, PCA-DE-FFNN, PCA-GA-FFNN, and FFNN. It is observed that PCA not only minimize the dimensionality of stock price time series data but also improve the accuracy of the model and deals with multicollinearity problem in time series econometric model (ARDL). PSO was employed as a global search technique to evolve the initial parameters of the FFNN. The local search capability of the LMBP algorithm has been improved by assigning the initial weight and bias obtained from PSO. The ARDL model can be significantly used along with PCA to model the stock price time series forecasting problem but perform inferior to the soft computing method considered in this study. Experimental results attained by applying the proposed model to three stock indices indicate the superior forecasting performance of the proposed model in comparison with other models considered in this study. In this paper, we considered only technical indicators; future research will be focused on fundamental indicators and macro as well as micro-economic data that impact the stock markets. The proposed model can also be used to deal with other financial time series problems such as inflation forecasting, exchange rate forecasting, volatility prediction, and so forth. In the future, other nature inspired and evolutionary algorithms will also be considered to improve the parameters of the model.

CONFLICT OF INTERESTS

The authors declare that there are no conflict of interests.

AUTHOR CONTRIBUTIONS

Conceptualization, writing—original draft preparation, software, visualization: Gourav Kumar. *Methodology, result validation, writing—reviewing and editing*: Uday Pratap Singh. *Writing—reviewing and supervision*: Sanjeev Jain.

ORCID

Gourav Kumar http://orcid.org/0000-0001-6167-9430
Uday Pratap Singh http://orcid.org/0000-0003-2077-0793
Sanjeev Jain http://orcid.org/0000-0001-5804-6982

REFERENCES

1. Lin CS, Chiu SH, Lin TY. Empirical mode decomposition-based least squares support vector regression for foreign exchange rate forecasting. *Econ Model*. 2012;29(6):2583-2590.
2. Zhang DY, Song HW, Chen P. Stock market forecasting model based on a hybrid ARMA and support vector machines. In: 2008 International Conference on Management Science and Engineering 15th Annual Conference Proceedings, Long Beach, CA, September 10–12, 2008:1312-1317.
3. Singh K, Shastri S, Bhadwal AS, et al. Implementation of exponential smoothing for forecasting time series data. *Int J Sci Res Comput Sci Appl Manag Stud*. 2019;8(1):1-8.

4. Adebiyi AA, Adewumi AO, Ayo CK. Comparison of ARIMA and artificial neural networks models for stock price prediction. *J Appl Math*. 2014;2014:1-7. https://doi.org/10.1155/2014/614342
5. Kristjanpoller W, Minutolo MC. A hybrid volatility forecasting framework integrating GARCH, artificial neural network, technical analysis and principal components analysis. *Exp Syst Appl*. 2018;109:1-11.
6. Box GE, Jenkins GM, Reinsel GC, Ljung GM. *Time series analysis: forecasting and control*. Hoboken, New Jersey: John Wiley & Sons; 2015.
7. Sakhre V, Singh UP, Jain S. FCPN approach for uncertain nonlinear dynamical system with unknown disturbance. *Int J Fuzzy Syst*. 2017;19(2):452-469.
8. Cui L, Yang S, Chen F, Ming Z, Lu N, Qin J. A survey on application of machine learning for Internet of Things. *Int J Mach Learn Cyber*. 2018;9(8):1399-1417.
9. D'Angelo G, Tipaldi M, Glielmo L, Rampone S. Spacecraft autonomy modeled via Markov decision process and associative rule-based machine learning. *IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. Padua, Italy, June 21-23, 2017:324-329. https://doi.org/10.1109/MetroAeroSpace.2017.7999589
10. Musumeci F, Rottondi C, Nag A, et al. An overview on application of machine learning techniques in optical networks. *IEEE Commun Surv Tutor*. 2018;21(2):1383-1408.
11. D'Angelo G, Palmieri F. Knowledge elicitation based on genetic programming for non destructive testing of critical aerospace systems. *Future Gener Comput Syst*. 2020;102:633-642.
12. Chouhan SS, Kaul A, Singh UP. Image segmentation using computational intelligence techniques. *Arch Computat Methods Eng*. 2019;26(3):533-596.
13. Mahajan S, Mittal N, Pandit AK. Image segmentation using multilevel thresholding based on type II fuzzy entropy and marine predators algorithm. *Multimed Tools Appl*. 2021;80:19335-19359. https://doi.org/10.1007/s11042-021-10641-5
14. Singh JP, Jain S, Arora S, Singh UP. Vision-based gait recognition: a survey. *IEEE Access*. 2018;6:70497-70527.
15. Florez-Lozano J, Caraffini F, Parra C, Gongora M. Cooperative and distributed decision-making in a multiagent perception system for improvised land mines detection. *Inf Fusion*. 2020;64:32-49.
16. Guresen E, Kayakutlu G, Daim TU. Using artificial neural network models in stock market index prediction. *Exp Syst Appl*. 2011;38(8):10389-10397.
17. Lu CJ, Lee TS, Chiu CC. Financial time series forecasting using independent component analysis and support vector regression. *Decis Support Syst*. 2009;47(2):115-125.
18. Atsalakis GS, Valavanis KP. Surveying stock market forecasting techniques–Part II: Soft computing methods. *Exp Syst Appl*. 2009;36(3):5932-5941.
19. Yao X. Evolving artificial neural networks. *Proc IEEE*. 1999;87(9):1423-1447.
20. da Silveira Bohrer J, Grisci BI, Dorn M. Neuroevolution of neural network architectures using CoDeepNEAT and Keras, arXiv preprint arXiv: 2002.04634, 2020.
21. D'Angelo G, Palmieri F. GGA: a modified genetic algorithm with gradient-based local search for solving constrained optimization problems. *Inf Sci (Ny)*. 2021;547:136-162.
22. Roodschild M, Sardiñas JG, Will A. A new approach for the vanishing gradient problem on sigmoid activation. *Prog Artif Intell*. 2020;9(4):351-360.
23. Reddy GT, Reddy MPK, Lakshmanna K, et al. Analysis of dimensionality reduction techniques on big data. *IEEE Access*. 2020;8:54776-54788.
24. Kumar D, Meghwani SS, Thakur M. Proximal support vector machine based hybrid prediction models for trend forecasting in financial markets. *J Comput Sci*. 2016;17:1-13.
25. Guyon I, Elisseeff A. An introduction to variable and feature selection. *J Mach Learn Res*. 2003;3:1157-1182.
26. Wright AH. Genetic algorithms for real parameter optimization. In: Rawlins GJE, eds. *Foundations of Genetic Algorithms*. (Vol. 1, pp. 205-218). Burlington, Massachusetts, USA: Morgan Kaufmann Publishers; 1991.
27. Yarpiz Binary and Real-Coded Genetic Algorithms; 2021. https://www.mathworks.com/matlabcentral/fileexchange/52856-binary-and-real-coded-genetic-algorithms. MATLAB Central File Exchange. Accessed March 15, 2021.
28. Storn R, Price K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J Glob Optim*. 1997;11(4):341-359.

29. Yarpiz. Differential Evolution (DE); 2021. https://www.mathworks.com/matlabcentral/fileexchange/52897-differential-evolution-de. MATLAB Central File Exchange. Accessed March 10, 2021.
30. Kumar G, Jain S, Singh UP, Kumar. Stock market forecasting using computational intelligence: a survey. *Arch Computat Methods Eng*. 2021;28:1067-1101. https://doi.org/10.1007/s11831-020-09413-5
31. Yang F, Chen Z, Li J, Tang L. A novel hybrid stock selection method with stock prediction. *Appl Soft Comput*. 2019;80:820-831.
32. Gocken M, Ozcalici M, Boru A, Dosdogru AT. Stock price prediction using hybrid soft computing models incorporating parameter tuning and input variable selection. *Neural Comput Appl*. 2019;31(2):577-592.
33. Nadkarni J, Neves RF. Combining neuro evolution and principal component analysis to trade in the financial markets. *Exp Syst Appl*. 2018;103:184-195.
34. Senapati MR, Das S, Mishra S, Novel A. Model for stock price prediction using hybrid neural network. *J Inst Eng Ser B*. 2018;99(6):555-563.
35. Weng B, Lu L, Wang X, Megahed FM, Martinez W. Predicting short-term stock prices using ensemble methods and online data sources. *Exp Syst Appl*. 2018;112:258-273.
36. Singh UP, Jain S. Optimization of neural network for nonlinear discrete time system using modified quaternion firefly algorithm: case study of Indian currency exchange rate prediction. *Soft Comput*. 2018;22(8):2667-2681.
37. Ulke V, Sahin A, Subasi A. A comparison of time series and machine learning models for inflation forecasting: empirical evidence from the USA. *Neural Comput Appl*. 2018;30(5):1519-1527.
38. Zhong X, Enke D. Forecasting daily stock market return using dimensionality reduction. *Exp Syst Appl*. 2017;67:126-139.
39. Pradeepkumar D, Ravi V. Forecasting financial time series volatility using particle swarm optimization trained quantile regression neural network. *Appl Soft Comput*. 2017;58:35-52.
40. Chiang WC, Enke D, Wu T, Wang R. An adaptive stock index trading decision support system. *Exp Syst Appl*. 2016;59:195-207.
41. Qiu MY, Song Y, Akagi F. Application of artificial neural network for the prediction of stock market returns: the case of the Japanese stock market. *Chaos Solitons Fract*. 2016;85:1-7.
42. Inthachot M, Boonjing V, Intakosum S. Artificial neural network and genetic algorithm hybrid intelligence for predicting Thai stock price index trend. *Comput Intell Neurosci*. 2016;2016:1-8. https://doi.org/10.1155/2016/3045254
43. Gocken M, Ozcalici M, Boru A, Dosdogru AT. Integrating metaheuristics and artificial neural networks for improved stock price prediction. *Exp Syst Appl*. 2016;44:320-331.
44. Reza H, Shahrabi J, Hadavandi E. A bat-neural network multi-agent system (BNNMAS) for stock price prediction: case study of DAX stock price. *Appl Soft Comput*. 2015;29:196-210.
45. Naranjo R, Meco A, Arroyo J, Santos M. An intelligent trading system with fuzzy rules and fuzzy capital management. *Int J Intell Syst*. 2015;30(8):963-983.
46. Bisoi R, Dash PK. A hybrid evolutionary dynamic neural network for stock market trend analysis and prediction using unscented Kalman filter. *Appl Soft Comput*. 2014;19:41-56.
47. Asadi A, Hadavandi E, Mehmanpazir F, Nakhostin MM. Hybridization of evolutionary Levenberg–Marquardt neural networks and data pre-processing for stock market prediction. *Knowl-Based Syst*. 2012;35:245-258.
48. More JJ. The Levenberg-Marquardt algorithm: implementation and theory. In: Watson GA, eds. *Numerical Analysis*. (Vol. 630, pp. 105-116). Berlin, Heidelberg: Springer; 1978.
49. Jumadinova J, Dasgupta P. A multi-agent system for analyzing the effect of information on prediction markets. *Int J Intell Syst*. 2011;26(5):383-409.
50. Wang J, Wang J. Forecasting stock market indexes using principle component analysis and stochastic time effective neural networks. *Neurocomputing*. 2015;156:68-78.
51. Singh UP, Jain S. Modified chaotic bat algorithm-based counter propagation neural network for uncertain nonlinear discrete time system. *Int J Comput Intell Appl*. 2016;15(3):1650016.
52. Goh AT. Back-propagation neural networks for modeling complex systems. *Artif Intell Eng*. 1995;9(3):143-151.
53. Kennedy J, Eberhart R. Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks, Perth, WA, November 17–December 1, 1995;9:1942-1948.

54. Pesaran MH, Shin Y. An autoregressive distributed-lag modelling approach to cointegration analysis. *Econ Soc Monogr*. 1998;31:371-413.
55. Van DTB, Bao HHG. A nonlinear autoregressive distributed lag (NARDL) analysis on the determinants of Vietnam's stock market. In: International Econometric Conference of Vietnam; 2019;809:363-376.
56. DA, Dickey, WA, Fuller. Distribution of the estimators for autoregressive time series with a unit root. *J Am Stat Assoc*. 1979;74(366a):427-431.
57. Jolliffe IT. Principal components in regression analysis. In: Buhlmann P, Diggle P, Gather U, Zeger S, eds. *Principal Component Analysis*. (pp. 29-155). New York, NY: Springer; 1986. https://doi.org/10.1007/978-1-4757-1904-8_8
58. Nifty 50. https://in.finance.yahoo.com/quote/%5ENSEI/history/. Accessed April 20, 2020.
59. Sensex. https://in.finance.yahoo.com/quote/%5EBSESN/history/. Accessed April, 30 2020.
60. S&P 500. https://finance.yahoo.com/quote/%5EGSPC/history/. Accessed May, 10 2020.
61. Shynkevich Y, McGinnity TM, Coleman SA, Belatreche A, Li Y. Forecasting price movement using technical indicators: investing the impact of varying window length. *Neurocomputing*. 2017;264:71-88.
62. Technical Analysis. https://www.investopedia.com/technical-analysis-4689657. Accessed November 11, 2019.
63. Technical Indicators. https://school.stockcharts.com/doku.php?id=technicalindicators. Accessed November 11, 2019.
64. Botchkarev A. Performance metrics (error measures) in machine learning regression, forecasting and prognostics: properties and typology. arXiv preprint arXiv: 1809.03006, 2018:1-37.
65. Theil H. *Applied Economic Forecasting*. Amsterdam: North-Holland Pub. Co.; 1966.
66. Technical analysis library in python. https://technical-analysis-library-inpython.readthedocs.io/en/latest/ta.html. Accessed December 30, 2019.
67. Akaike H. Fitting autoregressive models for prediction. *Ann Inst Stat Math*. 1969;21(1):243-247.
68. Friedman M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *J Am Stat Assoc*. 1937;32(200):675-701.
69. Garcia S, Herrera F. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *J Mach Learn Res*. 2008;9(12):2677-2694.

How to cite this article: Kumar G, Singh UP, Jain S. Hybrid evolutionary intelligent system and hybrid time series econometric model for stock price forecasting. *Int J Intell Syst*. 2021;36:4902-4935. https://doi.org/10.1002/int.22495