International Journal of Computational Intelligence Systems
Vol. **14(1)**, 2021, *pp.* 1337–1355
DOI: https://doi.org/10.2991/ijcis.d.210409.002; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Research Article

A Repairing Artificial Neural Network Model-Based Stock Price Prediction

S. M. Prabin[1,*], M. S. Thanabal[2]

[1]*Assistant Professor, Computer Science and Engineering, PSNA College of Engineering and Technology, Dindigul, Tamil Nadu, India*
[2]*Professor, Computer Science and Engineering, PSNA College of Engineering and Technology, Dindigul, Tamil Nadu, India*

ARTICLE INFO

Article History
Received 23 Jul 2020
Accepted 31 Mar 2021

Keywords
Stock price
RANN
Learning algorithms
Self-organizing
Dynamic

ABSTRACT

Predicting the stock price movements based on quantitative market data modeling is an open problem ever. In stock price prediction, simultaneous achievement of higher accuracy and the fastest prediction becomes a challenging problem due to the hidden information found in raw data. Various prediction models based on machine learning algorithms have been proposed in the literature. In general, these models start with the training phase followed by the testing phase. In the training phase, the past stock market data are used to learn the patterns toward building a model that would then use to predict future stock prices. The performance of such learning algorithms heavily depends on the quality of the data as well as optimal learning parameters. Among the conventional prediction methods, the use of neural network has greatest research interest because of their advantages of self-organizing, distributed processing, and self-learning behaviors. In this work, dynamic nature of the data is mainly focused. In conventional models the retraining has to be carried out for two cases: the data used for training has higher noise and outliers or model trained without preprocessing; the learned data has to update dynamically for recent changes. In this sense, propose a self-repairing dynamic model called repairing artificial neural network (RANN) that correct such errors effectively. The repairing includes adjusting the prediction model from noise, outliers, removing a data sample, and adjusting an attribute value. Hence, the total reconstruction of the prediction model could be avoided while saving training time. The proposed model is validated with five different real-time stock market data and the results are quantified to analyze its performance.

1. INTRODUCTION

Financial data gains more attention by the research and business people across the world, it is highly volatile time series data, analyzing it would help to improve the commerce in many ways. The market process that handles the long term financial instrument is known as capital market. These instruments could be stocks, mutual funds, bonds, equities, trades, and derivatives. These funds are the investments received from the public and used as the company's capital resource. People make their investment decisions based on returns and risk of each instrument. One major instrument is stock, buying and selling are the two major activities performed over these funds. Every day, the stock price varies depends on its demand and supply that are influenced by various factors like currency rate, inflation, and social and political conditions. The stock market index represents the trend of stock price oscillation. Predicting the direction of such a trend is attracted by many researchers. The public or the investor can be benefited from an accurate forecast of the stock index trends. The stock market forecasting could be divided into two types of analysis: fundamental analysis and technical analysis. Fundamental analysis dealt with the basic financial data of organizations like money supply, earnings, interest rate, inflation rate, cash flow, price-earnings ratio, the book to market ratio, and returns. Technical analysis is to handle rational data like the correlation between the indicators, price, and volume.

Stock price forecasting is achieved by identifying the patterns and trends of the time series data. Hair *et al.* [1], reported that multiple regression methods are suitable for financial data analysis, especially for stock price forecasting. However, the performance could be degraded when the data become more complex [2]. Mendenhall *et al.* [3], testified that the regression models are helpful to prone linear patterns only, whereas the stock market data is nonlinear in general. Hence, the neural network (NN)-based approaches are more appropriate for forecasting the series. The main advantage of NN is that they are able to build a nonlinear model without requiring a priori knowledge about the transactions. Moreover, the NN follows a nonparametric approach, as it reduces the complexity of the training. Other models such as support vector machines (SVMs) with handcrafted features [4,5], random forests [6,7], ensemble of the same classifiers [8], integration of different classifiers [9], and the most advanced and recent one is the deep learning networks [10,11] are the evidence of the progress in stock price forecasting problem. In continuation, the other recent works Chong *et al.* [12], Gudelek *et al.* [13], Hiransha *et al.* [14], and Barra *et al.* [15], depict

*Corresponding author. Email: smprabinphd@gmail.com

the research focuses on exploring various network framework and methods in stock market domain.

Most of the prediction models continued after ensuring that the data is clean and noise-free. However, when some of the data samples could be found incorrect or corrupted while building the model, at that time the erroneous sample would have been used for training the prediction model. Now, it is necessary to remove or correct the model as the incorrect data might have infected the model. One solution is to rebuild the model from scratch to resolve this issue, however, it involves greater time complexity. This work, propose a new solution called a repairing artificial neural network (RANN) for handling such dynamic situation with the trained model without rebuilding it from the scratch. The complete repairing model is to adopt the dynamic nature of the data. Though the data could be pre-processed in the early step, there might be need for adding or removing attributes over a period of time. For adding such trendy data or removing the outdated data from the trained model, RANN would be more useful as demonstrated.

The rest of the paper is organized as follows: The following section presents a summary of related works on neural-network-based stock price forecasting methods. Section 3 introduces the concept of an artificial NN. Section 4 explains the proposed RANN to handle dynamic data changes while learning. Section 5 discusses the experimental setup. Section 6 presents the quantified results and discussions. Section 7 concludes the paper.

2. RELATED WORKS

Stock selection has become a gradually hot subject in the field of finance research, as current interesting studies and works reviewed below:

Vaisla and Bhatt [16] have compared the stock price prediction performance from the NN and statistical technique. The performance is analyzed with various parameters like mean square error (MSE), mean absolute error (MAE) and root mean square error (RMSE), the results indicate that the NN-based stock market forecasting has better accuracy than the statistical methods like regression analysis. Dase and Pawar [17] have conducted a study on the application of ANN for stock market forecasting. This study concludes that the ANN-based stock price prediction models have the ability to achieve better accuracy than other techniques. Liao and Wang [18] proposed a NN model with stochastic time effective function. The investigation results indicated that this model is more effective than conventional NN models. Mostafa [19] has applied both multi-layer perceptron (MLP) NN and GRNN for forecasting the Kuwait stock exchange (KSE) data. The simulation results shown that GRNN outperforms the conventional regression and ARIMA models in stock price prediction. Lu [20] have presented an NN-based stock price forecasting model integrated with a denoising method. Independent component analysis (ICA) is used here for denoising. The investigation's results indicate that the denoised data improves the NN prediction, and reported that ICA-based denoising outperforms the wavelet denoising technique.

The major objective of stock price forecasting is to reach better prediction accuracy with minimum training data as well as with a simple learning model. The authors [21,22] have proposed an prediction models called and autoregressive integratedmoving average (ARIMA) and cerebellar model articulation controller neural network (CAMC NN) for stock index prediction. The experimental results estimated with the Taiwan stock exchange (TSE) index indicate that CAMC-NN achieves better prediction than SVR and BPNN models.

Hadavandi *et al.* [23], have presented a hybrid forecasting model, where ANN is integrated with genetic fuzzy systems (GFS). The investigation results indicate this model outperforms the other existing models in terms of MAPE.

Different hybrid prediction models were used in different studies by integrating with optimization and intelligence algorithms. The authors [24–26] have proposed a models by using artificial fish swarm algorithm (AFSA), genetic algorithm (GA), and generalized auto regressive conditional heteroscedasticity (GARCH). The results have shown that the proposed models outperforms than existing ones. Chopra *et al.* [27], presented a hybrid MLP-NN with LM algorithm for stock market prediction, and the results indicate the significance of adding more neurons in the hidden layer. Chandar *et al.* [28], have implemented a hybrid NN model with a wavelet transform technique for stock index prediction. Here, the wavelet transform is used to decompose the time series data, further, the received wavelet components are used as input variables for building the forecasting model based on NN. Investigation results indicate the significance of wavelet components toward improving greater forecasting accuracy.

Fang *et al.* [29], have proposed a hybrid wavelet neural network (WNN) model with hierarchical GA (HGA) for stock index prediction. Here, the WNN architecture is evaluated with minimal hidden neurons and reported a better prediction accuracy. Chen *et al.* [30], applied recurrent neural network (RNN) for stock price forecasting and reported better performance with Chinese stock market data. Pang *et al.* [31], have presented an long short-term memory (LSTM) model for stock index prediction and evaluated through financial analysis. The experimental results are received from real-time data and reported that the LSTM model outperforms the artificial neural network (ANN) model. Chi [32] have presented a BPN for stock price forecasting and reported with better prediction accuracy, and it is suggested to add more attributes which are having a strong influence on the stock market. Lei [33] have proposed a hybrid model with rough set (RS) and WNN for stock price prediction. Here, RS is used for dimensionality reduction, followed by WNN is implemented as a prediction model. The performance of RS-WNN is studied and compared with other classifiers such as SVM and conventional WNN, the results indicate the effectiveness of the proposed method.

Yang *et al.* [34], have presented an extreme learning machine (ELM) model for stock price forecasting as well as for stock scoring. The experimental results indicate the effectiveness of the learning model and state the robustness of ELM with greater prediction accuracy. Li *et al.* [35], proposed a deep reinforcement learning (DRL) architecture for stock price prediction problem and demonstrated the feasibility of the DRL method against the Adaboost algorithms. Qiu *et al.* [36], compared the stock price prediction performance of LSTM, LSTM with wavelet transform, and gated recurrent unit (GRU) NN model. The simulation results indicate that the LSTM model achieves better prediction accuracy of 94% while reducing the error rate to 0.05. Kim and Kim [37] proposed an

integrated LSTM-CNN model for stock price prediction. The experimental results indicate that this integrated model outperforms the individual model. Dai and Zhu [38] fused the sum-of-the-parts (SOP) method with ensemble empirical mode decomposition (EEMD) to forecast stock market returns. The simulation results indicate the promising results on return predictions. Carta *et al.* [39], proposed an ensemble of deep Q-learning for stock price forecasting. This method avoids over-fitting, and do not use market annotations, and used ensemble of reinforcement learning for better prediction. The performance study indicate the significance of Q-learning with higher prediction accuracy.

Ibidapo *et al.* [40], presented a comprehensive review on applying soft computing techniques for stock market prediction and reported that the hybridization of ANN with soft computing algorithms outperforms better than the conventional machine learning algorithms. Yoon *et al.* [41] investigated NN and GA in short-term stock forecasting domain. The experimental results show that the GA-based on backpropagation NN model has a significant improvement in stock price index series forecasting accuracy. Cai *et al.* [42] presented a new fuzzy time series model combined with ant colony optimization (ACO) and auto-regression. The ACO is adopted to obtain a suitable partition of the universe of discourse to promote the forecasting performance. Siddique*et al.* [43], proposed a hybrid stock value forecasting model with support vector regression (SVR) integrated with particle swarm optimization (PSO). Empirical results show that the proposed model enhances the performance of the previous prediction model.

3. ARTIFICIAL NEURAL NETWORK

NN models have been widely applied for stock market forecasting as they are robust with noisy data, moreover, the generalized regression neural network (GRNN) has been successfully implemented for a wide range of stock market predictions applications. Devadoss and Ligori [44] listed the features of ANN as follows:

- ANN doesn't require any prior knowledge about the data as it learns from samples.
- ANN models can produce better generalization even with noisy data.
- ANNs are nonlinear, hence it is more suitable for Stock price forecasting.

Figure 1 depicts a function of a single artificial neuron, which is also known as a perceptron in the ANN domain.

The connection between each neuron is assigned with a real number known as weights (w), and the output, (y) of each neuron is estimated by using a nonlinear function (f) of the sum of its inputs (x) as shown in Figure 1. Sometimes the neurons are associated with an additional input called bias (b). The function f is known as the activation function, used to introduce nonlinearity into the output y. In this way, ANNs are more suitable for stock market data as they are nonlinear. A neuron can either be activated or not based on the output of the activation function.

Artificial neuron or perceptron is the basic computation element of an ANN. The objective is to learn the optimal weights between perceptron, this process is known as a training or learning step. There are two types of architecture: single-layer perceptron (SLP) and MLP. SLP consists of two layers: the input and output layer, the input layers receive the input data samples and the output layer estimates the result of the NN.

The SLP architecture is able to build a decent generalization model, however, they are suitable for linearly separable problems. MLP is used to learn nonlinear data samples. A backpropagation learning-based neural network (BPN) is utilized as a classifier. The BPN follows the three-layer architecture of MLP, consists of an input, a hidden, and an output layer. For the stock price forecasting problem, the number of neurons in the input layer and the hidden layer is kept equal to the number of input variables (n) in the dataset, and the output layer consists of only one neuron as the expected output is the future stock price. The input variables are later discussed in Section 4.2. A typical three-layer BPN architecture is shown in Figure 2.

Once the architecture is established, then the two-weight matrices: weights between the input layer and hidden layer $[U]$ and the weights between the hidden layer and output layer $[V]$ are initialized with random values between 0 and 1. This state is defined as

$$[U]^0 = rand\ (0, 1) \tag{1}$$

$$[V]^0 = rand\ (0, 1) \tag{2}$$

After the initialization step, a set of data samples is partitioned from the dataset that is to be used for training. With the set of training samples, a single learning step is described in the following text.

From this training dataset, a data sample of input variables is fed to input layer I_I, the output of the input layer O_I is estimated with a

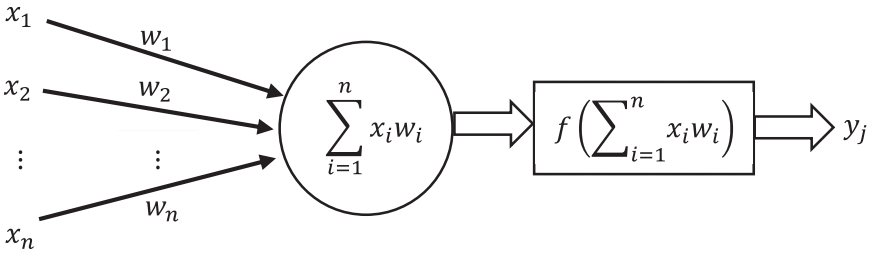

Figure 1 | Systematic diagram for neural network.

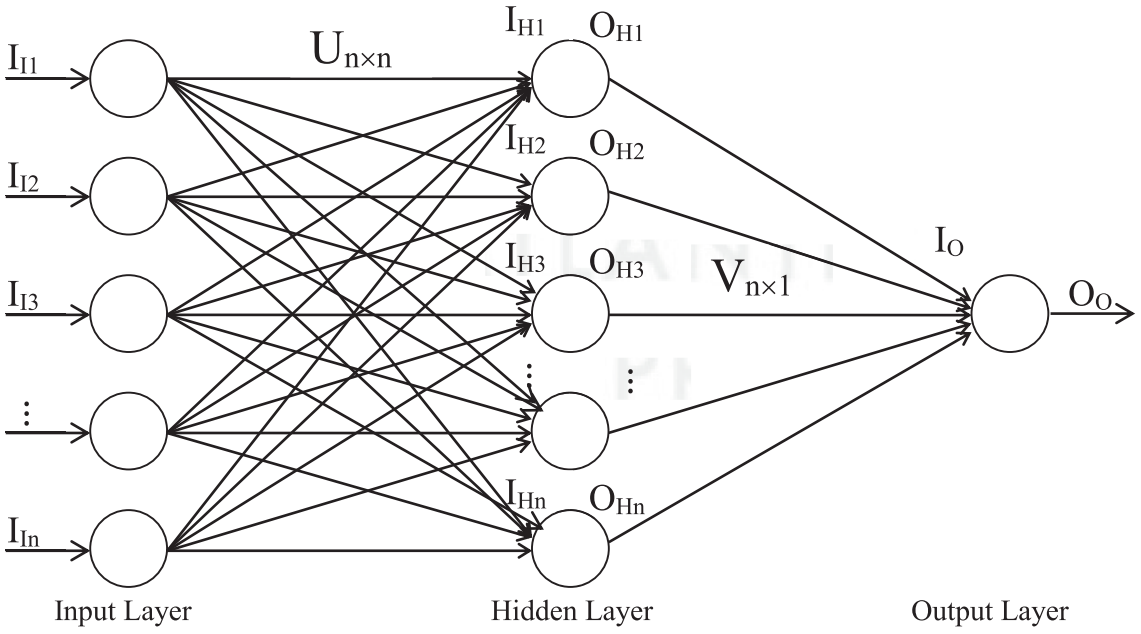

Figure 2 Backpropagation neural network.

"purelin" activation function as the first step. The purelin function transfers the input as the output, defined as

$$O_I = I_I \tag{3}$$

In the next step, the input to the hidden layer I_H layer is computed with a matrix multiplication between the output of the input layer and the weight matrix [U], defined as

$$I_H = \left[U^T\right] O_I \tag{4}$$

Then the output of the hidden layer is estimated with a "sigmoid" activation function as given below

$$O_H = f\left(I_H\right) = \frac{1}{\left(1 + e^{-I_H}\right)} \tag{5}$$

Further, the input to the output layer, I_O is computed with a matrix multiplication between the output of the hidden layer and the weight matrix [V], defined as

$$I_O = \left[V^T\right] O_H \tag{6}$$

And the output of the output layer is estimated by applying sigmoid function over its input, defined as

$$O_O = f\left(I_O\right) = \frac{1}{\left(1 + e^{-I_O}\right)} \tag{7}$$

The output of the NN O_O is compared with the actual output of the corresponding training sample to estimate the error of the network e. For example, for the k^{th} training sample, the error is computed as

$$e = |T_k - O_{ok}| \tag{8}$$

where T_k is the actual (target) output of the k^{th} training sample, and O_{ok} is the predicted output value for the kth training sample. A threshold is set for a negligible error (e.g., 0.0001), if the error is lower than the threshold, then the training is continued with the next training sample, else the weights are updated relative to the error and the output of the NN is computed again. This weight update procedure would be repeated until the error becomes lower than the threshold.

The weight updation based on the backpropagation learning approach is given below. At first, the weights between the hidden layer and output layer [V] are updated as

$$\delta = \left(T_k - O_{Ok}\right) O_{Ok} \left(1 - O_{Ok}\right) \tag{9}$$

$$[Y] = O_H.\delta \tag{10}$$

$$[\Delta V] = \varphi \left[V^T\right] \xi [Y] \tag{11}$$

where φ and ξ are the momentum and learning rate parameters that are used to avoid local convergence. In the next step, the [U] weight matrix is updated with the backpropagated error δ, as estimated between output and hidden layer, and its impact on [V] matrix, estimated as given in Eq. (12). With the ρ value, the error propagation term ω is estimated as given in Eq. (13).

$$\rho = [V]\,\delta \tag{12}$$

$$\omega = \rho_k \left(O_{Hk}\right) \left(1 - O_{Hk}\right) \tag{13}$$

$$[X] = O_I\omega = I_I\omega \tag{14}$$

$$[\Delta U] = \varphi \left[U^T\right] \xi [X] \tag{15}$$

With the weight update estimates, the weight matrices are updated.

$$[U]^{t+1} = [U]^t + [\Delta U] \tag{16}$$

$$[V]^{t+1} = [V]^t + [\Delta V] \tag{17}$$

where t and $t+1$ are the current and next iterations of the training step. This learning procedure is then repeated for all the data samples in the training dataset.

ANN models have many advantages as highlighted in the related works section, however, these models can't handle dynamic changes of the dataset. For example, while learning the kth sample, it is found that the $(k-i)$th training sample has fed with incorrect data values. It is difficult to adjust the weights at present as $(k-i)$th samples have been learned already. There is no such mechanisms are found in the literature to adopt the dynamic changes for the learned samples. In the proposed ANN model the dynamic data changes could be adopted effectively, without rebuilding the model as discussed in the following section.

4. PROPOSED RANN MODEL

In conventional models the retraining has to be carried out for two cases: the data used for training has higher noise and outliers or model trained without preprocessing, the learned data has to update dynamically for recent changes. In this sense, propose a self-repairing dynamic model called RANN that correct such errors effectively. The following are the chances of dynamic changes which would infect the learning model, as the samples have been learned already

- Case 1 – There might be a change in an attribute value
- Case 2 – There might be a change in more attribute values
- Case 3 – There might be a change in the decision values
- Case 4 – There might be a need for eliminating row(s) of the sample, as it might be outdated
- Case 5 – There might be a need for removing an attribute (column)
- Case 6 – There might be a need for adding an attribute (column)

These problems are addressed in the proposed RANN model.

The proposed RANN learning model focuses on renovating the ANN learning model while adopting the dynamic behavior of the data samples. For the proposed RANN model, the initial weight matrices $[U]^0$ and $[V]^0$ are stored for further use. The abovementioned chances of repairing are discussed in detail in the following text.

Case 1 – There might be a change in an attribute value, as it has been learned already

Once the training is over or in progress, when it is found that an attribute value x_{ej} of a data sample I_e fed earlier to the current NN training model $\left(NN_c\right)$ is incorrect, then the training is suspended immediately. Another learning model NN_r is constructed with the same architecture as the trained model NN_c, initialized with the weights $[U]^0$, and $[V]^0$ is fed with the correct data sample I_r. The learning is continued with NN_r to estimate the optimal weights between input-hidden and hidden-output layers which minimizes the training error. These optimal weights from NN_r model $[U]^r$ and $[V]^r$ are saved. Then the NN_r learning model is reinitialized with the weights $[U]^0$ and $[V]^0$ and the training phase is executed for the incorrect data sample, I_e, the back-propagated weight matrices with the optimal weights are stored under $[U]^e$ and $[V]^e$.

With these two sets of weight matrices, the difference of weights are estimated, and added to the current weigh matrices $[U]^c$ and $[V]^c$ of the learning model NN_c. The weight difference between the weight matrices $[U]^r$ and $[U]^e$ for the j^{th} input variable is estimated and each difference amount is divided with the total number of samples learned so far (s), defined as

$$[U]_j^c = [U]_j^c + \left[\frac{\left([U]_j^r - [U]_j^e\right)}{s}\right] \tag{18}$$

The weight difference between the weight matrices $[V]^r$ and $[V]^e$ are estimated and added to the current learning model as

$$[V]^c = [V]^c + \left(\frac{[V]^r - [V]^e}{s}\right) \tag{19}$$

After the adjustment of current weight matrices, the training phase continued either with the last training sample if the training is over, or with the next training sample if the training is in progress.

Case 2 – There might be a change in more attribute values

The previous case deals with the change in the single value of a single attribute. The dynamic changes could happen with more values of the same attribute or multiple attributes. To adopt such changes to the learning model, the current learning model NN_c is temporarily suspended and its weight matrices $[U]^c$ and $[V]^c$ are stored. The samples to be updated $I_{r_i}, i = 1, 2, \ldots, u$ with their correct attribute values are collected and stored as a separate dataset, D_r. For the same set of samples, the incorrect samples are collected and buffered in another dataset, D_e. Similar to case-1 solution, a new learning model NN_r is constructed and initialized with $[U]^0$ and $[V]^0$. With this repairing model, the samples from the dataset D_r are fed to estimate the optimal weights $[U]^r$ and $[V]^r$, and they are saved. Again, the NN_r model is reinitialized, and this time it is fed with the samples from the dataset D_e, then the corresponding weights $[U]^e$ and $[V]^e$ are stored. Once the weight matrices are estimated for both repaired and incorrect samples, their differences are measured to update the original weight matrices of NN_c learning model as given in the following equations. Then the learning process is continued with the recent sample of data. Here the weights are divided with the ratio between the number of incorrect samples (u) and the number of samples learned already.

$$[U]_i^c = [U]_i^c + \left[\frac{\left([U]_i^r - [U]_i^e\right)}{(u/s)}\right] \tag{20}$$

$$[V]^c = [V]^c + \left(\frac{[V]^r - [V]^e}{(u/s)}\right) \tag{21}$$

Case 3 – There might be a change in the decision values or output classes.

For the real-time data, the dynamic changes could happen with the class labels of decision values. For the past two cases, the changes in the input variables are handled, this case deals with the output variable of the network. The basic weight updation procedure similar to the previous two cases is followed, except that all the elements of $[U]$ matrix are updated rather than for a specific or set of input variables. The weight matrices for the correct and incorrect samples are computed and the current weight matrices are updated as

$$[U]^c = [U]^c + \left[\frac{([U]^r - [U]^e)}{s}\right] \tag{22}$$

$$[V]^c = [V]^c + \left(\frac{[V]^r - [V]^e}{s}\right) \tag{23}$$

Case 4 – There might be a need for eliminating row(s) of the sample, as it might be outdated.

For a nonlinear time-series data, the learning model has to adopt the new samples every time, and the same time the past or outdated data has to be removed from the trained model to align the model perfectly with the current trend. Hence, it is necessary to remove the outdated sample from the learned model. Once again, the similar weight updation process is followed with minimal changes from the previous equations. The outdated samples are received from the original dataset and stored as a separate dataset D_o. Initially, the current learning model NN_c is suspended and its weight matrices $[U]^c$ and $[V]^c$ are preserved. Then a fresh model, NN_e is constructed and initialized with the weights $[U]^0$ and $[V]^0$, then the outdated samples from D_o are fed to this NN to estimate the optimal weights $[U]^e$ and $[V]^e$. The difference of weights is estimated between the optimal weights of NN_e model and the initial weights, then the difference is added to the current weight matrices to update the NN_c learning model. The weight updation is defined as

$$[U]^c = [U]^c + \left[\frac{([U]^0 - [U]^e)}{(u/s)}\right] \tag{24}$$

$$[V]^c = [V]^c + \left(\frac{[V]^0 - [V]^e}{(u/s)}\right) \tag{25}$$

where u is the total number of outdated samples.

Case 5 – There might be a need for removing an attribute (column)

Recently the dimensionality of the data is growing exponentially that increases the memory and computation requirements. One common solution is to apply feature selection methods to identify and keep the most relevant features and discard the remaining. In general, feature selection methods are to be executed before the classification process starts. Suppose a relevant feature could be identified as outdated later on, in such a situation the architecture of the classifier has to be renovated to remove such features from the current learning model. This case addresses the issue of removing an input variable from the learned NN model. At first, the current learning model is suspended and the current weight matrices $[U]^c$ and $[V]^c$ are stored. Let us consider that the jth attribute is to be removed from the learned model, then the input neuron representing that attribute I_{Ij}, is detached from the network. Correspondingly, the neuron at the same place in the hidden layer H_{Hj} also removed from the network model. Then the weight matrices are repaired as follows. The $[U]$ matrix of size $n \times n$ is reduced to $(n-1) \times (n-1)$ by removing the jth row and column of the weight matrix. The average fraction of each element in the eliminated row is added to the corresponding row elements in the scaled-down matrix $[U]^d$, and the average fraction of each element in the removed column is added to the corresponding column elements of $[U]^d$.

For an illustrative example, consider the given weight matrix of size 5×5,

$$[U]^c = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ u_{21} & u_{22} & u_{23} & u_{24} & u_{25} \\ u_{31} & u_{32} & u_{33} & u_{34} & u_{35} \\ u_{41} & u_{42} & u_{43} & u_{44} & u_{45} \\ u_{51} & u_{52} & u_{53} & u_{54} & u_{55} \end{bmatrix}_{5\times5} \tag{26}$$

To remove the 3rd attribute, the corresponding row & column is removed the weight matrix to scale it down to 4×4.

$$[U]^d = \begin{bmatrix} u_{11} & u_{12} & \cancel{u_{13}} & u_{14} & u_{15} \\ u_{21} & u_{22} & \cancel{u_{23}} & u_{24} & u_{25} \\ \cancel{u_{31}} & \cancel{u_{32}} & \cancel{u_{33}} & \cancel{u_{34}} & \cancel{u_{35}} \\ u_{41} & u_{42} & \cancel{u_{43}} & u_{44} & u_{45} \\ u_{51} & u_{52} & \cancel{u_{53}} & u_{54} & u_{55} \end{bmatrix}_{5\times5} = \begin{bmatrix} u_{11} & u_{12} & u_{14} & u_{15} \\ u_{21} & u_{22} & u_{24} & u_{25} \\ u_{41} & u_{42} & u_{44} & u_{45} \\ u_{51} & u_{52} & u_{54} & u_{55} \end{bmatrix}_{4\times4} \tag{27}$$

Further, each element of the reduced matrix is added with the average fraction of the elements from the removed row and column, defined as

$$[U]^d = \begin{bmatrix} u_{11} + \left(\frac{u_{31}}{4}\right) + \left(\frac{u_{13}}{4}\right) & u_{12} + \left(\frac{u_{32}}{4}\right) + \left(\frac{u_{13}}{4}\right) \\ u_{21} + \left(\frac{u_{31}}{4}\right) + \left(\frac{u_{23}}{4}\right) & u_{22} + \left(\frac{u_{32}}{4}\right) + \left(\frac{u_{23}}{4}\right) \\ u_{41} + \left(\frac{u_{31}}{4}\right) + \left(\frac{u_{43}}{4}\right) & u_{42} + \left(\frac{u_{32}}{4}\right) + \left(\frac{u_{43}}{4}\right) \\ u_{51} + \left(\frac{u_{31}}{4}\right) + \left(\frac{u_{53}}{4}\right) & u_{52} + \left(\frac{u_{32}}{4}\right) + \left(\frac{u_{53}}{4}\right) \end{bmatrix} \tag{28}$$

$$\begin{bmatrix} u_{14} + \left(\frac{u_{34}}{4}\right) + \left(\frac{u_{13}}{4}\right) & u_{15} + \left(\frac{u_{35}}{4}\right) + \left(\frac{u_{13}}{4}\right) \\ u_{24} + \left(\frac{u_{34}}{4}\right) + \left(\frac{u_{23}}{4}\right) & u_{25} + \left(\frac{u_{35}}{4}\right) + \left(\frac{u_{23}}{4}\right) \\ u_{44} + \left(\frac{u_{34}}{4}\right) + \left(\frac{u_{43}}{4}\right) & u_{45} + \left(\frac{u_{35}}{4}\right) + \left(\frac{u_{43}}{4}\right) \\ u_{54} + \left(\frac{u_{34}}{4}\right) + \left(\frac{u_{53}}{4}\right) & u_{55} + \left(\frac{u_{35}}{4}\right) + \left(\frac{u_{53}}{4}\right) \end{bmatrix}$$

The $[V]$ matrix of size $n \times 1$ is reduced to $(n-1) \times 1$ by removing the jth element of the weight matrix. The average fraction of the jth element is added to each element in the scaled-down matrix $[V]^d$.

Consider the given weight matrix, and the 3rd neuron representing the input variable to be eliminated.

$$[V]^c = \begin{bmatrix} v_1 \\ v_2 \\ \cancel{v_3} \\ v_4 \\ v_5 \end{bmatrix} \tag{29}$$

$$[V]^d = \begin{bmatrix} v_1 \\ v_2 \\ v_4 \\ v_5 \end{bmatrix} = \begin{bmatrix} v_1 + \left(\frac{v_3}{4}\right) \\ v_2 + \left(\frac{v_3}{4}\right) \\ v_4 + \left(\frac{v_3}{4}\right) \\ v_5 + \left(\frac{v_3}{4}\right) \end{bmatrix} \tag{30}$$

With the scaled-down weight matrices $[U]^d$ and $[V]^d$, the learning is continued by backpropagating the weights until the error is minimized.

Case 6 – There might be a need for adding an attribute (column)

Scalability is an important issue of real-time datasets since the dimensionality could be increased in horizontal and vertical directions. In the case of a vertical increase, the number of samples gets added to the dataset, and for horizontal growth, the number of attributes is added to the dataset. The NN model can adopt the new samples incrementally as they are received, this way the vertical growth of the data can be handled effortlessly. However, for the horizontal growth of the data, when a new attribute is about to be added to the existing dataset, the NN architecture should be repaired to accept the new input variable. Adding an artificial neuron to the input and hidden layer would make the NN model accept the new attribute. However, assigning the weights for the new connections from the recently added neuron to the other neurons in the hidden layer, and similarly assigning the weights from the existing input neurons to the latest neuron in the hidden layer is the problem to be addressed here. For this case, the current weight matrices are scaled up from $n \times n$ to $(n+1) \times (n+1)$ for the $[U]$ matrix, and $n \times 1$ to $(n+1) \times 1$ for the $[V]$ matrix. The newly introduced connections are assigned with the mean weights estimated from the other elements that exist in the same row and column of the scaled-up matrices $[U]^p$ and $[V]^p$.

For an illustrative example, consider a $[U]^c$ weight matrix of size 4×4,

$$[U]^c = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ u_{21} & u_{22} & u_{23} & u_{24} \\ u_{31} & u_{32} & u_{33} & u_{34} \\ u_{41} & u_{42} & u_{43} & u_{44} \end{bmatrix}_{4\times4} \tag{31}$$

Adding a new element at the last would introduce the 5th row and column, the weights assigned for them are the mean of elements in the corresponding row and column. The (5, 5) diagonal element is assigned with the average of elements that exist in the primary diagonal.

$$[U]^p = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & + \\ u_{21} & u_{22} & u_{23} & u_{24} & + \\ u_{31} & u_{32} & u_{33} & u_{34} & + \\ u_{41} & u_{42} & u_{43} & u_{44} & + \\ + & + & + & + & + \end{bmatrix}_{5\times5} \tag{32}$$

$$[U]^p = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & \frac{1}{4}\sum_{i=1}^{4} u_{1i} \\ u_{21} & u_{22} & u_{23} & u_{24} & \frac{1}{4}\sum_{i=1}^{4} u_{2i} \\ u_{31} & u_{32} & u_{33} & u_{34} & \frac{1}{4}\sum_{i=1}^{4} u_{3i} \\ u_{41} & u_{42} & u_{43} & u_{44} & \frac{1}{4}\sum_{i=1}^{4} u_{4i} \\ \frac{1}{4}\sum_{j=1}^{4} u_{j1} & \frac{1}{4}\sum_{j=1}^{4} u_{j2} & \frac{1}{4}\sum_{j=1}^{4} u_{j3} & \frac{1}{4}\sum_{j=1}^{4} u_{j4} & \frac{1}{4}\sum_{j=1}^{4} u_{jj} \end{bmatrix}_{5\times5} \tag{33}$$

Similarly the other weight matrix $[V]^c$ is scaled up as given below

$$[V]^c = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \tag{34}$$

$$[V]^p = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ + \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ \frac{1}{4}\sum_{j=1}^{4} v_j \end{bmatrix} \tag{35}$$

After scaling up the weight matrices, the learning process is continued with the current sample consists of the new variable. The backpropagation procedure estimates the optimal weights as discussed before.

The following algorithm summarizes the proposed RANN model-based learning for all the six cases to handle the dynamic data.

Algorithm – Repairing Artificial Neural Network Model

Inputs – Initial weights $[U]^0$ and $[V]^0$, the current learning model NN_c and its weight matrices $[U]^c$ and $[V]^c$

Outputs – Repaired weight matrices

For any of the dynamic data updates, suspend the current learning model, and store its weights.

Follow the corresponding cases to handle the dynamic update.

Case 1 – To correct an erroneous attribute value

Construct and initialize a new learning model, NN_r with $[U]^0$ and $[V]^0$

Learn the weights for the correct sample, $[U]^r$ and $[V]^r$

Re-initialize the learning model, NN_r with $[U]^0$ and $[V]^0$

Learn the weights for the erroneous sample, $[U]^e$ and $[V]^e$

Update the corresponding element's weight at the current weight matrices $[U]^c$ and $[V]^c$ based on Equations (18) and (19)

Case 2 – To correct a set of an erroneous attribute value

Collect the data samples with the incorrect attribute values, D_e

Learn the optimal weights with a new learning model, $[U]^e$ and $[V]^e$

Collect the data samples with the correct attribute values, D_r

Learn the optimal weights with a new learning model, $[U]^r$ and $[V]^r$

Update the corresponding element's weight at the current weight matrices $[U]^c$ and $[V]^c$ based on Equations (20) and (21)

Case 3 – To correct a set of erroneous samples with incorrect decision values

Collect the data samples with the incorrect decision attribute values, D_e

Learn the optimal weights with a new learning model, $[U]^e$ and $[V]^e$

Collect the data samples with the decision correct attribute values, D_r

Learn the optimal weights with a new learning model, $[U]^r$ and $[V]^r$

Update all the elements of the current weight matrices $[U]^c$ and $[V]^c$ based on Equations (22) and (23)

Case 4 – To eliminate a set of data samples

Collect the outdated data samples, D_o

Learn the optimal weights with a new learning model, $[U]^e$ and $[V]^e$

Update all the elements of the current weight matrices $[U]^c$ and $[V]^c$ based on Equations (24) and (25)

Case 5 – To remove an attribute from the dataset

Remove the corresponding row and column of weight matrices representing the input variable

Update the other elements by distributing the average fraction of eliminated elements from the corresponding rows and column as given in the Equations (27) and (30)

Case 6 – To add an attribute from the dataset

Add a new row and column to the current weight matrices.

Assign the new elements by taking the mean of existing elements in the corresponding rows and column as given in the Equations (33) and (35)

After updating the current weight matrices, the learning procedure is continued with the current learning model NN_c with the latest data sample.

5. EXPERIMENTAL SETUP

5.1. Datasets

The performance of the proposed RANN model is evaluated with five different stock market index datasets: Nifty50 (N50), Nifty-Bank (NB), Nifty-Pharma (NP) from www.nseindia.com, and BSE-IT (BIT) & BSE-Oil-Gas (BOG) from www.bseindia.com. For each dataset, a collection of the last five years' historical index data from January 1, 2015 to December 31, 2019 is collected. The offered datasets contain attributes like index data, open, high, low, and close indices. The stock index variables such as open, high, low, and close are considered as input variables to the NN. Along with the index variables, *few more technical indicators are estimated and considered as input variables for the prediction model.* The closing stock index trend for all the five market data are shown in the following Figure 3.

5.2. Technical Indicators

Three sets of technical indicators (input variables) are used to evaluate the performance of the proposed RANN model. Group 1 variables consist of 4 index variables and 12 indicators as listed in Table 1 as given in Shen *et al.* [24], Group 2 variables consists of 4 index variables and 24 indicators as listed in Table 2 as presented in Chang [45], and Group 3 variables consist of 4 index variables and 12 indicators as presented in Table 3 as listed in Mingyue *et al.* [46].

C_t is the closing price and L_t is the lowest price of the stock index at time t. L_n is the lowest low price in the last n days, H_t is the highest price at time t, H_n is the highest high price of the index in the last n days. MA_n is the moving average of the stock price value in the last n days: $MA_n = \left(\sum_{i=1}^{n} C_{t-i+1}\right)/n$, $M_t = \left(H_t + L_t + C_t\right)/3$, $SM_t = \left(\sum_{i=1}^{n} M_{t-i+1}\right)/n$, $D_t = \left(\sum_{i=1}^{n} |M_{t-i+1} - SM_t|\right)/n$. Up_t is the upward price change of the stock index at time t, and Dw_t is the downward price change of the index at time t. The following section presents the quantified results and their investigation.

6. RESULTS AND DISCUSSIONS

The prediction performance of the proposed RANN model is evaluated with the following measures including the RMSE, mean absolute difference (MAD), mean absolute percentage error (MAPE), directional accuracy (DA), correct up trend (CP), and correct down trend (CD) as discussed in Lu and Wu [22] and Dai *et al.* [47]. The measures RMSE, MAD, and MAPE are used to analyze the stock price forecasting error, and DA, CP, and CD are used to measure the prediction accuracy. The measures are defined as below.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\left(T_i - P_i\right)^2}{N}} \tag{36}$$

$$MAD = \frac{\sum_{i=1}^{N}|T_i - P_i|}{N} \tag{37}$$

$$MAPE = \frac{\sum_{i=1}^{N}|| \frac{T_i - P_i}{T_i}}{N} \tag{38}$$

$$DA = \frac{100}{N}\sum_{i=1}^{N} d_i, where\ d_i = \begin{cases} 1 & \left(P_i - P_{i-1}\right)\left(T_i - T_{i-1}\right) \geq 0 \\ 0 & otherwise \end{cases} \tag{39}$$

$$CP = \frac{100}{N_1}\sum_{i=1}^{N} d_i, where\ d_i = \begin{cases} 1 & \left(P_i - P_{i-1}\right)\left(T_i - T_{i-1}\right) \geq 0 \\ 0 & otherwise \end{cases} \tag{40}$$

$$CD = \frac{100}{N_2}\sum_{i=1}^{N} d_i, w \tag{41}$$

$$here\ d_i = \begin{cases} 1 & \left(P_i - P_{i-1}\right)\left(T_i - T_{i-1}\right) \geq 0 \\ 0 & otherwise \end{cases}$$

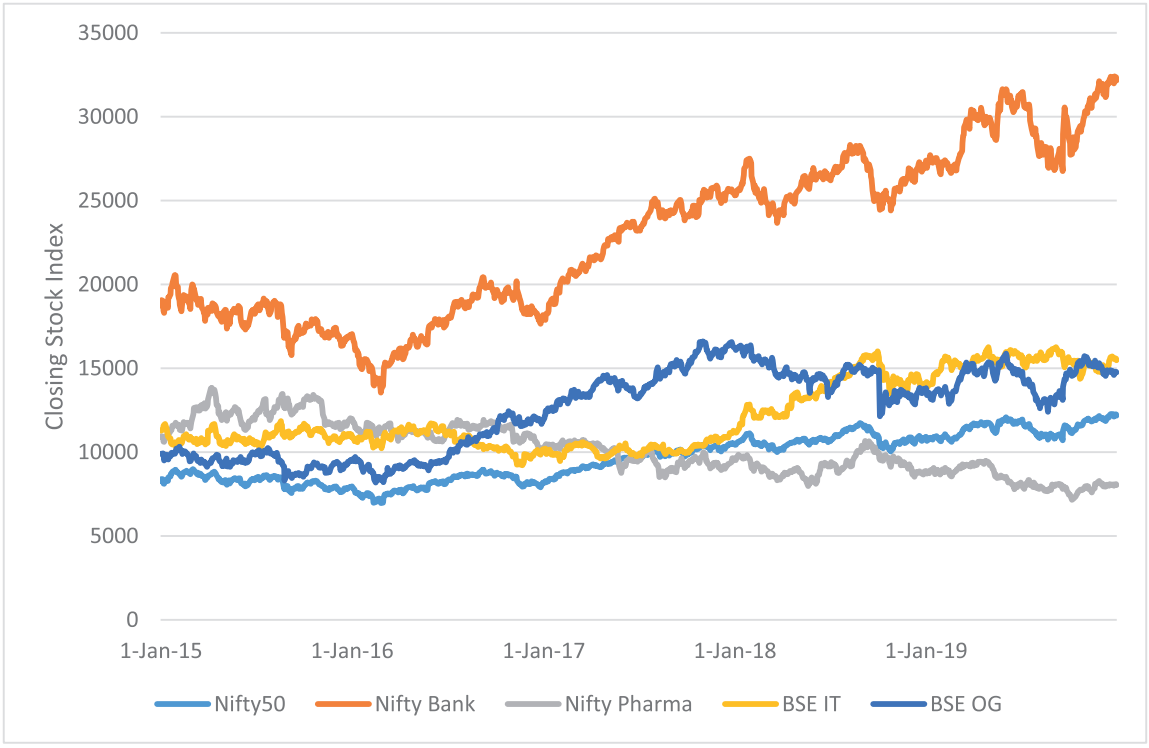

Figure 3 | Closing stock price trend.

Table 1 | List of technical indicators group-1.

Input Variables	Technical Indicators	Formulas
OBV	On balance volume	$v_i + v_{i-1}$, v_i represents the trade volume of the present day
MA5	Moving average for 5 days	$\left(C_t + C_{t-1} + C_{t-2} + C_{t-3} + C_{t-4}\right)/5$, C_t is the closing index of the current day
BIAS6	Bias	$C_t\left[\left(C_t - MA_6\right)/MA_6\right] \times 100$, bias is the difference between the closing price and moving average line
PSY12	The psychological line for 12 days	$\left(Up_{12}/12\right) \times 100$, Up_{12} means the number of times when stock price going up within 12 days
ASY5	Average stock yield of 5 days before the forecasting date	$\left(SY_{t-1} + SY_{t-2} + SY_{t-3} + SY_{t-4} + SY_{t-5}\right)/5$ $SY = \left(\ln C_t - \ln C_{t-1}\right) \times 100$
ASY4	ASY of 4 days before the forecasting date	$\left(SY_{t-1} + SY_{t-2} + SY_{t-3} + SY_{t-4}\right)/4$
ASY3	ASY of 3 days before the forecasting date	$\left(SY_{t-1} + SY_{t-2} + SY_{t-3}\right)/3$
ASY2	ASY of 2 days before the forecasting date	$\left(SY_{t-1} + SY_{t-2}\right)/2$
ASY1	ASY of 1 day before the forecasting date	SY_{t-1}
CI3	Closing indices of 3 days before the forecasting date	
CI2	Closing indices of 2 days before the forecasting date	
CI1	Closing indices of 1 day before the forecasting date	

The symbols T and P represent the target and predicted value, respectively, N represents the total number of data samples used for training or testing phase, N_1 and N_2 are the numbers of data samples belonging to uptrend and downtrend, respectively.

The input variables are normalized with the Z-score normalization method that transforms the stock price index to a distribution with a mean of 0 and a standard deviation of 1. The operators μ and σ signifies the mean and standard deviation of input variables, respectively:

$$X'_k = \frac{X_k - \mu}{\sigma} \tag{42}$$

Z-score normalization does not guarantee a common numerical range for the normalized stock indices. X_k is the kth input variable and X'_k is the normalized index or indicator. Ten-fold cross-validation is performed to partition the stock market datasets

Table 2 List of technical indicators group-2.

Input Variables	Technical Indicators	Description
5MA, 6MA, 10MA, 20MA	Moving average (MA)	Moving average is used to highlight the direction of a trend and smooth out the price, volume variation that can lead to misinterpretation.
5BIAS, 10BIAS	Bias	Bias represents the characteristics of the stock prince to return to the average price.
6RSI, 12RSI	Relative strength index (RSI)	RSI compares the magnitude of recent gains to recent losses in an attempt to determine overbought and over-sold conditions of an asset.
K, D	Nine-day stochastic line	The stochastic K & D lines are utilized to estimate the trends of over-purchasing, over-selling or deviation.
MACD	Moving average convergence and divergence (MACD)	MACD shows the difference between a fast and slow exponential moving average (EMA) of closing prices.
12W%R	Williams %R (W%R)	W%R is the ratio of the number of rising duration and the total number of days. It represents the ratio between buying and selling power.
K – D	Differences of a technical index between K and D	Difference between the technical indicators K and D line
Δ5MA, Δ6MA, Δ10MA, Δ5BIAS, Δ10BIAS, Δ6RSI, Δ12RSI, Δ9K, Δ9D, ΔMACD, Δ12W%R	Differences in technical index $(\triangle)$	Differences between the technical index of *t*he day and $(t+1)$ st day.

Table 3 List of technical indicators group-3.

Input Variables	Formulas
Stochastic %K	$\left(C_t - L_n\right) / \left(H_n - L_n\right) \times 100$
Stochastic %D	$\sum_{i=0}^{n-1} \%K_{t-i}/n$
Stochastic slow %D	$\sum_{i=0}^{n-1} \%D_{t-i}/n$
Momentum	$C_t - C_{t-4}$
ROC (rate of change)	$C_t/C_{t-n} \times 100$
LW%R (Larry William's %R)	$\left(H_n - C_t\right) / \left(H_n - L_n\right) \times 100$
A/D oscillator (accumulation/distribution oscillator)	$\left(H_t - C_{t-1}\right) / \left(H_t - L_t\right)$
Disparity in 5 days	$C_t/MA_5 \times 100$
Disparity in 10 days	$C_t/MA_{10} \times 100$
OSCP (price oscillator)	$MA_5 - MA_{10}/MA_5$
CCI (commodity channel index)	$\left(M_t - SM_t\right) / \left(0.015 \times D_t\right)$
RSI (relative strength index)	$100 - 100/\left(1 + \frac{\sum_{i=0}^{n-1} Up_{t-i}}{n} / \frac{\sum_{i=0}^{n-1} Dw_{t-i}}{n}\right)$

for training and testing samples. Initially, the dataset is divided into ten equal partitions known as folds. For an evaluation step, 9-folds of data samples are used for training, and the remaining one fold is used for validation. This procedure is repeated ten times, where every single fold is used for validation, and the other 9-folds are used for training. The final performance is the mean of all ten validation.

For the dynamic changes, the stock market datasets are manually corrupted for 10% of data samples, and for the corrupted dataset RANN model is employed. The performance of the proposed RANN model-based closing stock price prediction is evaluated with the above measures and compared with the other existing models from the literature: ELM [34], LSTM-based model [36], WNN [33], GA-based ANN (GA-ANN) [48], dynamic neural network (DNN) [49], NN hybrid with nonlinear independent component analysis (NN-NLICA) [47], and radial basis function neural network (RBFNN) [24]. Table 4 illustrates the architecture and their hyper parameters value of each model used for performance study. The size of input neuron depends on number of index and indicators belongs to the group which is used for prediction.

Table 5 presents the quantified performance measures of various NN methods with Nifty 50 dataset. Among all the methods, the proposed RANN model is outperforming with lower error rates of 70.96, 15.56, 0.69 and higher prediction accuracy of 88.50, 76.93, 69.07 with Group-2 input variables. A comparison between a sample of target price and the corresponding predicted stock price from the RANN model with Group-2 variables are depicted in Figure 4. The solid line represents the target price and the red

Table 4 Architecture of stock price prediction model.

Hyper Parameters	RANN	LSTM	ELM	WNN	GA-ANN	DNN	NN-NLICA	RBFNN
#Input Neurons	16 / 28	16 / 28	16 / 28	16 / 28	16 / 28	16 / 28	16 / 28	16 / 28
#Hidden Neurons	10% training size	10 nodes/layer	10% training size	10% training size	10% training size	10% training size	7 – 10	10% training size
#Output Neuron	1	1	1	1	1	1	1	1
Activation Function	Sigmoid function	Softmax function	Sigmoid function	Wavelet basis function	Sigmoid function	Softmax function	Sigmoid function	Radial basis function
Learning Rate	0.05	0.001	–	0.05	0.05	–	0.05	0.05
Momentum	0.6	0.6	–	0.75	0.50	–	0.50	0.60
#epochs	500	500	–	500	500	500	500	500

Table 5 Performance comparison of the NN model with Nifty50 dataset.

NN Models	Input Variables	RMSE	MAD	MAPE	DA	CP	CD
	Group-1	87.57	21.42	1.53	87.73	76.31	68.53
RANN	Group-2	70.96	15.56	0.69	88.50	76.93	69.07
	Group-3	72.59	17.01	0.74	82.80	75.43	68.64
	Group-1	86.83	23.63	1.28	66.85	73.56	63.75
LSTM	Group-2	97.73	29.26	1.98	69.57	73.43	63.64
	Group-3	84.63	22.14	1.41	85.48	74.68	64.72
	Group-1	102.11	29.93	1.26	81.75	74.93	64.94
ELM	Group-2	95.31	27.26	0.90	79.76	73.56	63.75
	Group-3	100.65	29.30	0.88	71.08	73.06	63.32
	Group-1	92.78	27.80	1.99	80.43	72.81	63.10
WNN	Group-2	98.04	29.20	1.85	83.81	73.81	63.97
	Group-3	104.54	32.21	0.85	61.49	72.19	62.56
	Group-1	110.46	36.59	1.50	75.52	72.93	63.21
GA-ANN	Group-2	110.12	35.92	1.85	85.85	73.68	63.86
	Group-3	108.61	36.19	1.21	80.77	72.93	63.21
	Group-1	125.49	46.68	1.73	80.42	74.06	64.18
DNN	Group-2	120.55	43.44	1.81	76.69	73.31	63.53
	Group-3	122.73	45.80	0.78	67.87	74.80	64.83
	Group-1	124.98	44.73	2.84	69.87	74.30	64.40
NN-NLICA	Group-2	122.17	45.55	1.80	61.66	72.93	63.21
	Group-3	117.63	43.04	1.08	67.21	73.18	63.43
	Group-1	129.34	49.87	0.94	72.20	71.93	65.80
RBFNN	Group-2	127.14	47.92	1.81	74.15	74.55	64.61
	Group-3	129.17	50.98	2.21	71.03	72.92	66.67

circle markers represent the predicted price. Presenting the prediction results of all the classifiers makes the plot complex, hence it is compared only with the proposed model alone.

Table 6 list the estimated performance measures of various NN methods with Nifty Bank dataset. Comparatively, the proposed RANN model is outperforming with lower error rates of 70.82, 15.09, 0.23 and higher prediction accuracy of 81.58, 85.43, 73.64 with Group-2 input variables. A comparison between a sample of target price and the corresponding predicted stock price from RANN model with Group-2 variables are depicted in Figure 5.

Table 7 presents the estimated performance measures of various NN methods with Nifty Pharma dataset. Comparatively, the proposed RANN model shows a significant performance with lower error rates of 65.39, 13.96, 0.41 and higher prediction accuracy of 89.30, 86.55, 78.56 with Group-2 input variables. A comparison between a sample of target price and the corresponding predicted stock price from RANN model with Group-2 variables are depicted in Figure 6.

Table 8 list the estimated performance measures of various NN methods with the BSE-IT dataset. Comparatively, the proposed RANN model is outperforming with lower error rates of 61.31,

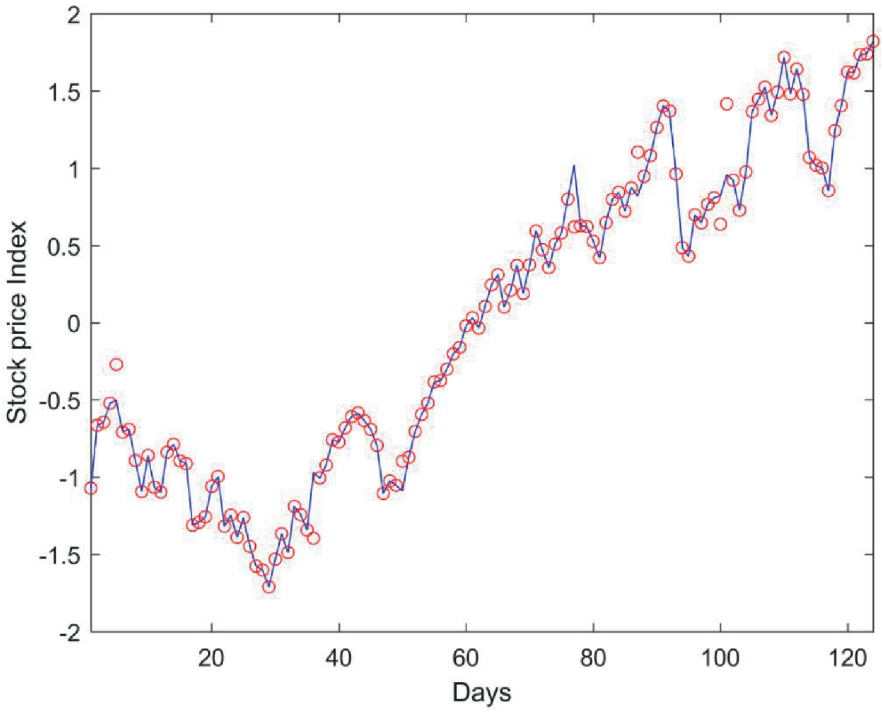

Figure 4 Target and predicted stock price comparison for Nifty 50 Dataset with repairing artificial neural network (RANN) model.

Table 6 Performance comparison of the NN model with Nifty-Bank dataset.

NN Models	Input Variables	RMSE	MAD	MAPE	DA	CP	CD
	Group-1	76.51	18.66	0.33	81.17	83.31	73.53
RANN	Group-2	70.82	15.09	0.23	81.58	85.43	73.64
	Group-3	77.95	19.72	0.27	69.12	84.68	72.99
	Group-1	93.16	25.65	0.41	62.12	74.18	64.29
LSTM	Group-2	91.09	25.93	0.41	59.81	73.06	63.32
	Group-3	94.05	26.76	2.57	54.57	73.68	63.86
	Group-1	97.93	28.50	0.51	52.54	71.44	61.91
ELM	Group-2	102.34	29.96	0.64	54.06	73.56	63.75
	Group-3	94.52	26.81	1.55	75.16	73.93	64.07
	Group-1	94.94	28.07	0.55	79.57	74.06	64.18
WNN	Group-2	97.59	29.54	2.03	75.52	72.43	62.78
	Group-3	101.92	31.76	0.54	70.22	74.80	64.83
	Group-1	117.84	39.61	0.65	66.38	74.30	64.40
GA-ANN	Group-2	114.65	37.49	1.24	68.47	72.43	62.78
	Group-3	111.16	36.96	2.30	64.94	74.06	64.18
	Group-1	121.17	44.53	0.89	58.31	73.06	63.32
DNN	Group-2	123.79	45.25	0.96	74.19	73.31	63.53
	Group-3	126.19	47.52	0.89	63.56	75.93	65.80
	Group-1	121.69	44.63	1.21	75.30	73.81	63.97
NN-NLICA	Group-2	126.60	47.47	1.06	69.22	72.93	63.21
	Group-3	124.19	46.12	1.08	67.21	74.18	64.29
	Group-1	129.60	50.61	1.88	53.80	73.81	63.97
RBFNN	Group-2	122.86	47.04	2.37	74.17	75.30	65.26
	Group-3	137.38	55.02	1.48	64.43	74.93	64.94

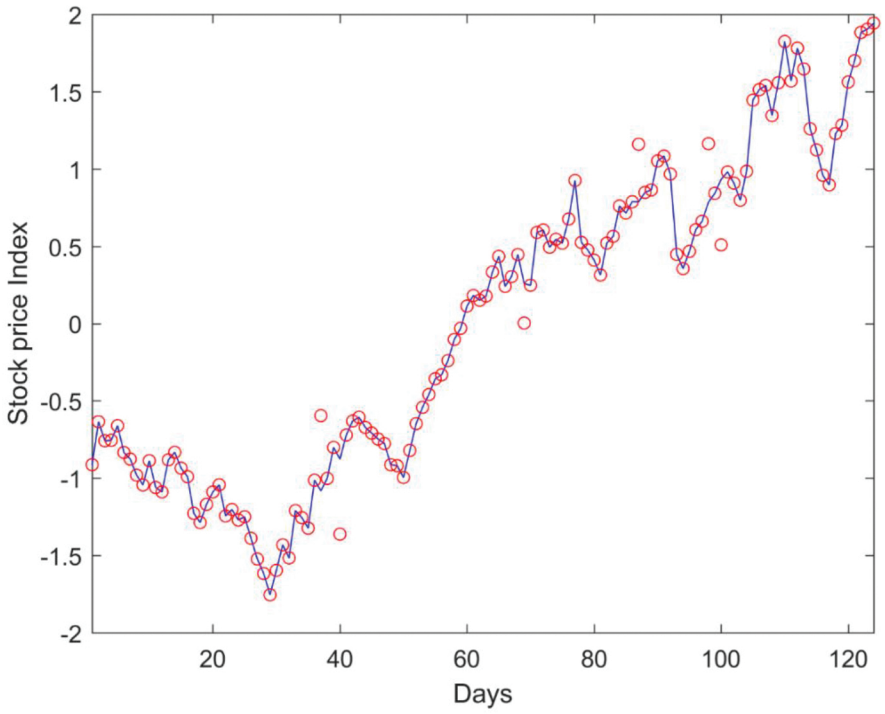

Figure 5 Target and predicted stock price comparison for Nifty Bank Dataset with repairing artificial neural network (RANN) model.

Table 7 Performance comparison of the NN model with Nifty-Pharma dataset.

NN Models	Input Variables	RMSE	MAD	MAPE	DA	CP	CD
	Group-1	87.66	21.41	0.49	87.91	81.30	76.13
RANN	Group-2	65.39	13.96	0.41	89.30	86.55	78.56
	Group-3	80.00	18.93	0.47	81.96	84.80	77.34
	Group-1	88.23	24.01	0.53	79.45	77.17	66.88
LSTM	Group-2	89.80	25.27	0.48	82.30	77.30	66.99
	Group-3	93.14	25.74	0.66	77.55	75.80	65.69
	Group-1	94.31	26.50	1.93	68.98	76.92	66.67
ELM	Group-2	92.47	26.09	0.85	80.66	76.18	66.02
	Group-3	90.97	25.72	1.97	64.64	78.05	67.64
	Group-1	102.44	31.52	0.79	77.31	76.30	66.13
WNN	Group-2	98.24	29.13	0.67	60.62	78.17	67.75
	Group-3	102.44	31.64	0.70	61.78	79.04	68.50
	Group-1	114.33	38.70	0.86	52.83	76.80	66.56
GA-ANN	Group-2	106.05	34.02	0.69	81.38	78.42	67.96
	Group-3	111.25	37.29	0.98	52.82	76.92	66.67
	Group-1	113.84	41.07	1.59	57.13	75.55	65.48
DNN	Group-2	119.70	43.71	1.35	78.34	78.79	68.29
	Group-3	128.83	48.51	2.74	80.94	76.18	66.02
	Group-1	123.58	45.13	1.66	64.60	75.80	65.69
NN-NLICA	Group-2	120.25	44.53	1.55	60.21	77.42	67.10
	Group-3	121.61	43.96	1.72	49.62	76.05	65.91
	Group-1	124.03	47.16	1.58	66.10	75.93	65.80
RBFNN	Group-2	130.34	50.87	2.45	60.86	76.67	66.45
	Group-3	135.17	53.57	2.06	66.04	78.29	67.86

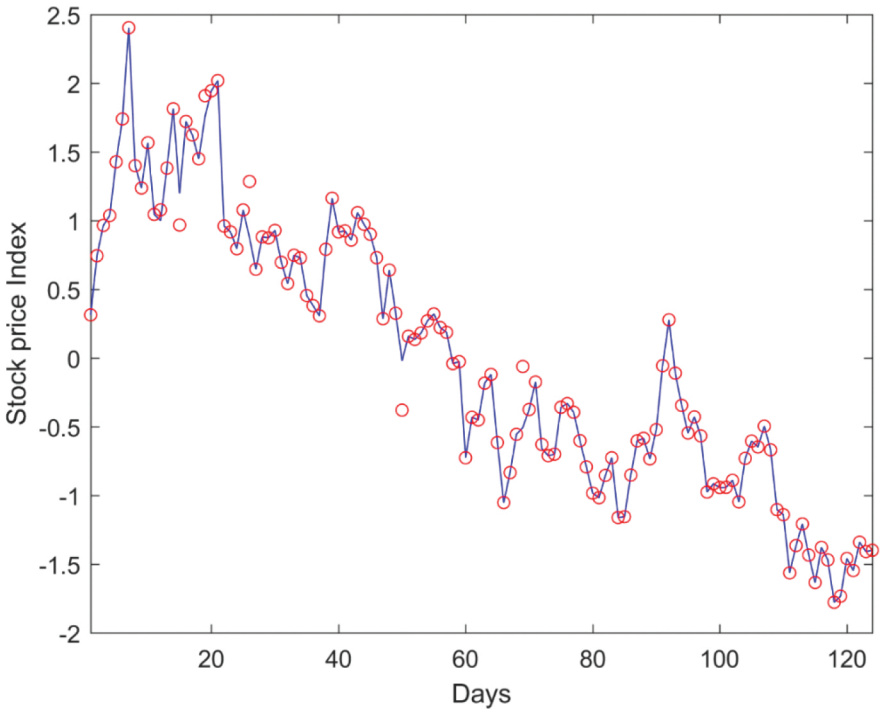

Figure 6 | Target and predicted stock price comparison for Nifty Pharma Dataset with repairing artificial neural network (RANN) model.

Table 8 | Performance comparison of the NN model with BSE-IT dataset.

NN Models	Input Variables	RMSE	MAD	MAPE	DA	CP	CD
	Group-1	61.31	13.38	0.26	87.53	77.93	74.94
RANN	Group-2	63.82	19.90	0.83	86.86	76.81	73.97
	Group-3	79.82	18.98	0.54	83.81	75.06	74.18
	Group-1	88.65	23.82	0.39	77.94	73.81	63.97
LSTM	Group-2	99.28	28.75	1.19	71.23	74.93	64.94
	Group-3	96.16	26.67	0.44	59.52	74.18	64.29
	Group-1	91.19	26.38	0.41	79.70	74.68	64.72
ELM	Group-2	100.93	29.26	0.51	81.18	75.30	65.26
	Group-3	92.65	26.70	1.18	75.40	73.06	63.32
	Group-1	89.95	25.87	0.46	77.09	73.81	63.97
WNN	Group-2	107.36	32.98	1.24	80.73	73.93	64.07
	Group-3	104.46	32.61	0.61	63.71	74.55	64.61
	Group-1	116.01	38.91	1.17	65.83	75.68	65.59
GA-ANN	Group-2	109.00	35.69	0.66	73.52	72.06	62.45
	Group-3	113.86	38.46	1.83	60.06	73.43	63.64
	Group-1	126.47	46.52	0.96	54.49	72.68	62.99
DNN	Group-2	116.87	41.35	0.76	71.82	74.30	64.40
	Group-3	122.44	46.17	1.07	80.48	72.31	62.67
	Group-1	122.40	44.26	1.10	79.73	74.68	64.72
NN-NLICA	Group-2	126.22	47.57	1.38	61.78	73.81	63.97
	Group-3	124.12	46.09	1.30	66.11	75.05	65.05
	Group-1	132.30	51.69	1.33	62.20	74.55	64.61
RBFNN	Group-2	136.06	53.87	1.29	60.70	73.18	63.43
	Group-3	129.93	50.55	1.45	67.28	74.18	64.29

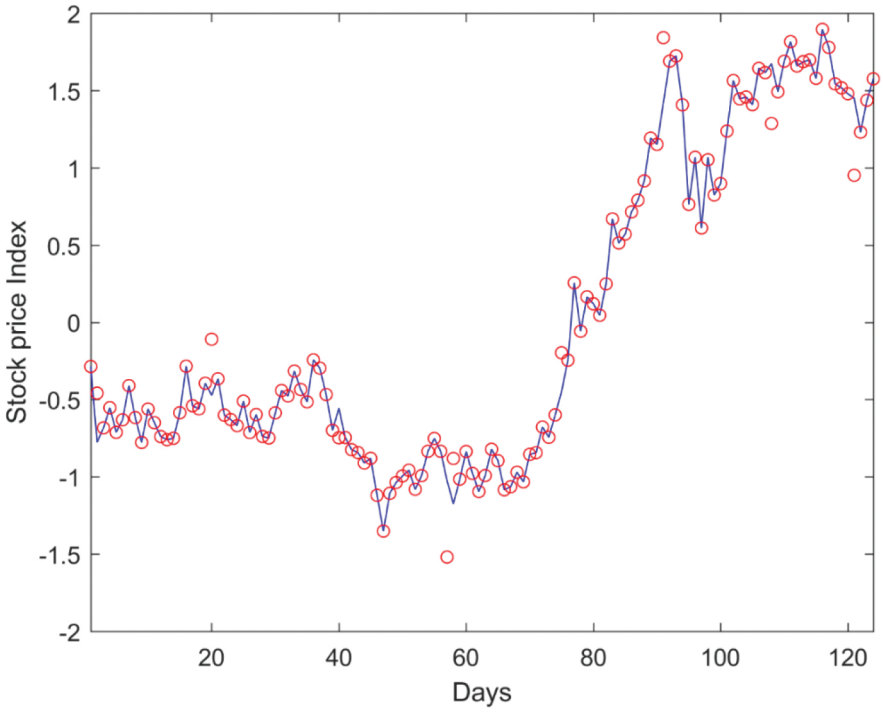

Figure 7 | Target and predicted stock price comparison for BSE IT Dataset with repairing artificial neural network (RANN) model.

13.38, 0.26 and higher prediction accuracy of 87.53, 77.93, 74.94 with Group-1 input variables. A comparison between a sample of target price and the corresponding predicted stock price from the RANN model with Group-1 variables are depicted in Figure 7.

Table 9 presents the estimated performance measures of various NN methods with the BSE-Oil and Gas dataset. Comparatively, the proposed RANN model shows a significant performance with lower error rates of 65.01, 13.63, 0.35 and higher prediction accuracy of 85.02, 78.43, 79.64 with Group-2 input variables. A comparison between a sample of target price and the corresponding predicted stock price from the RANN model with Group-2 variables are depicted in Figure 8.

In addition to comparing the proposed RANN model based prediction with the other models, it is also compared against with the repaired and corrupted data. Table 10 illustrates this comparison between RANN and ANN with unrepaired data (ANN_e), the results demonstrates that the unrepaired data achieves lower performance than the repaired one.

The prediction models are implemented with Matlab R2018® environment, and executed in 64-bit Windows 8.1 operating system, Intel-i5 processor, 8-GB memory configuration. Table 11 presents the comparison study on average computation time taken by each model from 10 trial runs. Comparatively the proposed RANN method achieves better prediction in minimum computation time.

Predicting the stock price with most relevant technical indicators would improve the prediction accuracy, in this way the computation time required by the prediction model could be reduced as the learning is proceeded with minimum inputs [50]. The concept of choosing more relevant input variables is known as feature selection or dimensionality reduction, aims to remove irrelevant or redundant features from a dataset to improve prediction performance. Zhong and Enke [51], applied principal component analysis (PCA) based feature selection methods for feature subset selection in stock forecasting domain. The experimental results indicated that the reduced feature set from PCA with ANN-based classification achieved higher prediction accuracy than with the complete features. Gündüz *et al.* [52], proposed a feature selection approach using gain ratio and relief function and reported significant performance improvement in stock forecasting. Haq *et al.* [53], proposed a multi-filter feature selection (MFFS) approach for choosing relevant technical indicators. Three different feature selection methods such as: L1 regularized logistic regression (L1-LR), SVM, and random forest (RF) are applied to rank the technical indicators and the top-ranked indicators are chosen and grouped to form the optimal feature subset. The investigation results shown that the input variables selected from MFFS approach outperforms with greater stock price prediction accuracy than the other approaches.

Here, the technical indicators from all the three groups are combined to form one complete set of 48 features. In preliminary step, one indicator, MA5 from Group-1, two indicators K & D from Group-2 and 5 indicators (LW%R, Disparity in 5 Days, Disparity in 10 Days, OSCP and RSI) from Group-3 are removed from the feature set as they were redundant. With the pre-filtered set of 40 features, the MFFS approach as discussed in Haq *et al.* [53], is employed to choose the most relevant features. The feature ranking is estimated for all five datasets and found that the ranking is different for each stock dataset. The mean ranking is estimated for each indicator and they are normalized between the range [0, 1]. Once the normalized mean ranking is computed, a threshold value is fixed to drop the features whose ranking are lower than the threshold. For the three approaches, the threshold values 0.03, 0.1 and 0.2

Table 9 | Performance comparison of the NN model with BSE oil & gas dataset.

NN Models	Input Variables	RMSE	MAD	MAPE	DA	CP	CD
	Group-1	75.38	17.72	0.36	80.36	76.93	78.07
RANN	Group-2	65.01	13.63	0.35	85.02	78.43	79.64
	Group-3	82.88	20.50	0.37	84.51	74.93	75.07
	Group-1	86.98	23.54	0.45	72.84	74.80	64.83
LSTM	Group-2	101.34	29.74	0.69	79.45	72.93	63.21
	Group-3	91.58	25.31	0.54	79.26	72.43	62.78
	Group-1	99.14	28.72	1.82	76.29	73.56	63.75
ELM	Group-2	97.53	28.29	0.75	75.35	74.06	64.18
	Group-3	93.41	27.45	1.47	75.34	74.30	64.40
	Group-1	101.69	30.42	0.62	84.52	72.56	62.88
WNN	Group-2	97.40	28.87	0.60	80.76	74.68	64.72
	Group-3	107.75	33.70	0.60	80.64	73.43	63.64
	Group-1	112.01	37.44	1.34	69.18	74.43	64.51
GA-ANN	Group-2	105.69	34.46	0.65	74.94	73.06	63.32
	Group-3	116.07	39.82	1.40	63.28	73.31	63.53
	Group-1	124.75	45.72	1.44	63.10	73.06	63.32
DNN	Group-2	119.88	43.19	1.85	71.70	74.93	64.94
	Group-3	125.07	47.15	1.41	70.44	71.81	62.24
	Group-1	121.41	45.51	1.02	78.23	72.19	62.56
NN-NLICA	Group-2	120.87	45.02	0.96	76.57	72.18	65.15
	Group-3	121.86	44.75	1.83	80.36	74.80	64.83
	Group-1	130.31	51.25	0.89	73.31	73.56	63.75
RBFNN	Group-2	126.68	49.93	1.17	75.18	73.18	63.43
	Group-3	129.44	51.04	1.22	52.50	70.94	61.48

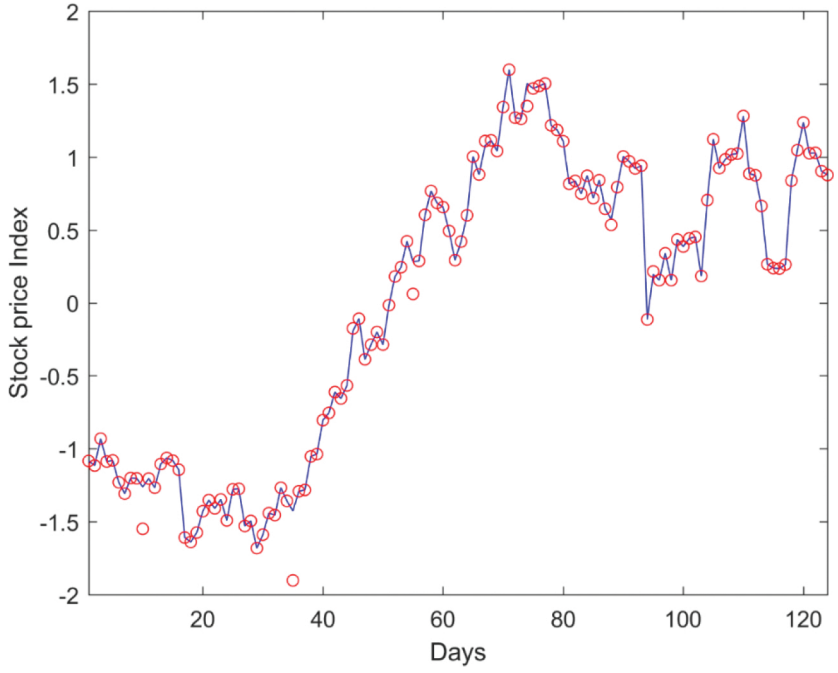

Figure 8 | Target and predicted stock price comparison for BSE Oil & Gas Dataset with repairing artificial neural network (RANN) model.

are fixed for L1-LR, SVM and RF respectively. The technical indicators having higher ranking than these threshold values are chosen from all the three feature selection approaches are merged to form the optimal feature subset. MFFS approach results in four subset of features, top-ranked features from three different feature selection approach and the combined subset. Table 12 lists the features selected from MFFS.

The stock price prediction performance based on directional accuracy (DA) measure is analyzed with these reduced feature subset and the results are depicted in Table 13. The results indicate that the reduced features from MFFS approach is achieving higher prediction accuracy than the other methods.

7. CONCLUSIONS

The ANN models are proven to be effective for stock price forecasting. However, they are not capable of handling dynamic data updates means that, whenever there is a change in the existing dataset, the learning process has to be repeated once again. Here, a novel NN model, RANN is proposed to adopt the dynamic nature

Table 10 Performance comparison of prediction model with corrupted and repaired data.

Dataset	NN Models	RMSE	MAD	MAPE	DA	CP	CD
Nifty 50	RANN	70.96	15.56	0.69	88.50	76.93	69.07
	ANN_e	128.40	48.75	0.74	50.22	61.13	62.50
Nifty Bank	RANN	70.82	15.09	0.23	81.58	85.43	73.64
	ANN_e	133.87	50.25	1.84	63.34	79.34	69.44
Nifty Pharma	RANN	65.39	13.96	0.41	89.30	86.55	78.56
	ANN_e	182.38	45.18	6.49	60.49	61.81	46.20
BSE IT	RANN	61.31	13.38	0.26	87.53	77.93	74.94
	ANN_e	162.44	57.75	2.83	51.87	53.18	46.79
BSE Oil & Gas	RANN	65.01	13.63	0.35	85.02	78.43	79.64
	ANN_e	160.31	71.52	1.99	43.11	53.65	43.57

Table 11 Performance comparison of stock prediction models based on computation cost.

	Computation Time (sec)							
Dataset	**RANN**	**LSTM**	**ELM**	**WNN**	**GA-ANN**	**DNN**	**NN-NLICA**	**RBFNN**
Nifty 50	78.12	89.05	105.21	134.01	138.20	226.10	136.55	141.08
Nifty Bank	81.82	88.86	105.61	132.54	147.61	217.08	145.16	142.23
Nifty Pharma	94.78	99.96	112.39	137.29	145.27	216.68	144.83	137.87
BSE IT	94.78	102.58	110.33	136.51	145.51	219.62	160.31	128.82
BSE Oil & Gas	93.61	96.49	100.13	140.32	155.03	212.47	161.91	129.79

Table 12 Optimal technical indicators identified by feature selection methods.

Methods	#Features	Selected Features
L1-LR	13	OBV, ASY5, 5MA, 6MA, 5BIAS, 10BIAS, 12W%R, MACD, 6RSI, 12RSI, Stochastic %K, Stochastic %D, A/D Oscillator
SVM	10	OBV, 5MA, 6MA, 5BIAS, 10BIAS, MACD, Stochastic %K, Stochastic %D, ROC, CCI
RF	9	OBV, 5MA, 6MA, 5BIAS, 10BIAS, MACD, Stochastic %K, Stochastic %D, CCI
MFFS	15	OBV, ASY5, 5MA, 6MA, 5BIAS, 10BIAS, 12W%R, MACD, 6RSI, 12RSI, Stochastic %K, Stochastic %D, A/D Oscillator, ROC, CCI

Table 13 Stock price predication performance analysis on feature selection methods.

Dataset	Directional Accuracy				
	MFFS	**RF**	**SVM**	**L1-LR**	**Original**
Nifty 50	98.37	92.87	93.28	92.30	88.50
Nifty Bank	88.04	86.48	86.43	85.49	81.58
Nifty Pharma	95.20	94.07	93.97	93.25	89.30
BSE IT	92.29	91.13	91.90	90.98	87.53
BSE Oil & Gas	90.35	89.30	89.26	88.49	85.02

of the dataset. The learning step could be suspended at any point in time, and the dynamic changes like changing attribute values, change of decision labels, removing outdated data samples, removing an attribute, inserting an attribute, could be adapted to the existing learned model. The dynamic changes are adopted by renovating the NN architecture or by adjusting the weights. In this way, it is not required to repeat the complete learning procedure, hence it reduces the computation cost. The performance of the proposed model is validated with five standard stock market datasets such as Nifty 50, Nifty Bank, Nifty Pharma, BSE IT, and BSE Oil and Gas. Five years of data are collected for each dataset, and the stock price forecasting performance is measure with three error rates and three prediction accuracy measures. The RANN model is compared with the existing five different NN models. The investigated results have shown that the RANN model is achieving lower error rates and higher prediction accuracy while adopting dynamic changes. In future, it is planned to explore the other learning models such as deep learning, LSTM, and convolutional neural network for stock price forecasting as well as to contribute toward achieving better prediction rate.

REFERENCES

[1] J.F. Hair, W.C. Black, B.J. Babin, R.E. Anderson, R.L. Tatham, Multivariate Data Analysis, vol. 5, Prentice Hall, Upper Saddle River, NJ, USA, 1998, pp. 207–219.

[2] K. Dacha, Causal modeling of stock market prices using neural networks and multiple regression: a comparison report, Finance India. 21 (2007), 923.

[3] W. Mendenhall, R.J. Beaver, B.M. Beaver, Introduction to Probability and Statistics, Cengage Learning, Canada, 2012.

[4] K.J. Kim, Financial time series forecasting using support vector machines, Neurocomputing. 55 (2003), 307–319.

[5] W. Huang, Y. Nakamori, S.Y. Wang, Forecasting stock market movement direction with support vector machine, Comput. Oper. Res. 32 (2005), 2513–2522.

[6] P. Ładyżyński, K. Żbikowski, P. Grzegorzewski, Stock trading with random forests, trend detection tests and force index volume indicators, in International Conference on Artificial Intelligence and Soft Computing, Zakopane, Poland, 2013, pp. 441–452.

[7] T. Manojlović, I. Štajduhar, Predicting stock market trends using random forests: a sample of the Zagreb stock exchange, in 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE, Opatija, Croatia, 2015, pp. 1189–1193.

[8] B. Weng, L. Lu, X. Wang, F.M. Megahed, W. Martinez, Predicting short-term stock prices using ensemble methods and online data sources, Expert Syst. Appl. 112 (2018), 258–273.

[9] M. Asad, Optimized Stock market prediction using ensemble learning, in 2015 9th International Conference on Application of Information and Communication Technologies (AICT), IEEE, Rostov on Don, Russia, 2015, pp. 263–268.

[10] T. Jagric, S. Bojnec, V. Jagric, Optimized spiral spherical self-organizing map approach to sector analysis–the case of banking, Expert Syst. Appl. 42 (2015), 5531–5540.

[11] T. Jagric, S. Bojnec, V. Jagric, A map of the European insurance sector – are there any borders?, Econ. Comput. Econ. Cybern. Stud. Res. 52 (2018), 283–298.

[12] E. Chong, C. Han, F.C. Park, Deep learning networks for stock market analysis and prediction: methodology, data representations, and case studies, Expert Syst. Appl. 83 (2017), 187–205.

[13] M.U. Gudelek, S.A. Boluk, A.M. Ozbayoglu, A deep learning based stock trading model with 2-D CNN trend detection, in 2017 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, Honolulu, HI, USA, 2017, pp. 1–8.

[14] M. Hiransha, E.A. Gopalakrishnan, V.K. Menon, K.P. Soman, NSE stock market prediction using deep-learning models, Procedia Comput. Sci. 132 (2018), 1351–1362.

[15] S. Barra, S.M. Carta, A. Corriga, A.S. Podda, D.R. Recupero, Deep learning and time series-to-image encoding for financial forecasting, IEEE/CAA J. Automat. Sinica. 7 (2020), 683–692.

[16] K.S. Vaisla, A.K. Bhatt, An analysis of the performance of artificial neural network technique for stock market forecasting, Int. J. Comput. Sci. Eng. 2 (2010), 2104–2109.

[17] R.K. Dase, D.D. Pawar, Application of artificial neural network for stock market predictions: a review of literature, Int. J. Mach. Intell. 2 (2010), 14–17.

[18] Z. Liao, J. Wang, Forecasting model of global stock index by stochastic time effective neural network, Expert Syst. Appl. 37 (2010), 834–841.

[19] M.M. Mostafa, Forecasting stock exchange movements using neural networks: empirical evidence from Kuwait, Expert Syst. Appl. 37 (2010), 6302–6309.

[20] C.J. Lu, Integrating independent component analysis-based denoising scheme with neural network for stock price prediction, Expert Syst. Appl. 37 (2010), 7056–7064.

[21] Y.B. Wijaya, S. Kom, T.A. Napitupulu, Stock price prediction: comparison of Arima and artificial neural network methods-An Indonesia Stock's Case, in 2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies, IEEE, Jakarta, Indonesia, 2010, pp. 176–179.

[22] C.J. Lu, J.Y. Wu, An efficient CMAC neural network for stock index forecasting, Expert Syst. Appl. 38 (2011), 15194–15201.

[23] E. Hadavandi, H. Shavandi, A. Ghanbari, Integration of genetic fuzzy systems and artificial neural networks for stock price forecasting, Knowl.-Based Syst. 23 (2010), 800–808.

[24] W. Shen, X. Guo, C. Wu, D. Wu, Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm, Knowl.-Based Syst. 24 (2011), 378–385.

[25] J.Z. Wang, J.J. Wang, Z.G. Zhang, S.P. Guo, Forecasting stock indices with back propagation neural network, Expert Syst. Appl. 38 (2011), 14346–14355.

[26] E. Guresen, G. Kayakutlu, T.U. Daim, Using artificial neural network models in stock market index prediction, Expert Syst. Appl. 38 (2011), 10389–10397.

[27] S. Chopra, D. Yadav, A.N. Chopra, Artificial neural networks based indian stock market price prediction: before and after demonetization, J. Swarm Intell. Evol. Comput. 8 (2019), 2.

[28] S.K. Chandar, M. Sumathi, S.N. Sivanandam, Prediction of stock market price using hybrid of wavelet transform and artificial neural network, Indian J. Sci. Technol. 9 (2016), 1–5.

[29] Y. Fang, K. Fataliyev, L. Wang, X. Fu, Y. Wang, Improving the genetic-algorithm-optimized wavelet neural network for stock market prediction, in 2014 International Joint Conference on Neural Networks (IJCNN), IEEE, Beijing, China, 2014, pp. 3038–3042.

[30] W. Chen, Y. Zhang, C.K. Yeo, C.T. Lau, B.S. Lee, Stock market prediction using neural network through news on online social networks, in 2017 International Smart Cities Conference (ISC2), IEEE, Wuxi, China, 2017, pp. 1–6.

[31] X. Pang, Y. Zhou, P. Wang, W. Lin, V. Chang, An innovative neural network approach for stock market prediction, J. Supercomput. 76 (2020), 2098–2118.
[32] W. Chi, Forecasting stock index based on BP neural network algorithm, in 2018 International Conference on Mechanical, Electronic, Control and Automation Engineering (MECAE 2018), Qingdao, China, Atlantis Press, 2018.
[33] L. Lei, Wavelet neural network prediction method of stock price trend based on rough set attribute reduction, Appl. Soft Comput. 62 (2018), 923–932.
[34] F. Yang, Z. Chen, J. Li, L. Tang, A novel hybrid stock selection method with stock prediction, Appl. Soft Comput. 80 (2019), 820–831.
[35] Y. Li, P. Ni, V. Chang, Application of deep reinforcement learning in stock trading strategies and stock forecasting, Computing. 102 (2020), 1305–1322.
[36] J. Qiu, B. Wang, C. Zhou, Forecasting stock prices with long-short term memory neural network based on attention mechanism, PloS One. 15 (2020), e0227222.
[37] T. Kim, H.Y. Kim, Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data, PloS One. 14 (2019), e0212320.
[38] Z. Dai, H. Zhu, Forecasting stock market returns by combining sum-of-the-parts and ensemble empirical mode decomposition, Appl. Econ. 52 (2020), 2309–2323.
[39] S. Carta, A. Ferreira, A.S. Podda, D.R. Recupero, A. Sanna, Multi-DQN: an ensemble of Deep Q-learning agents for stock market forecasting, Expert Syst. Appl. 164 (2020), 113820.
[40] I. Ibidapo, A.A. Adebiyi, O. Okesola, Soft computing techniques for stock market prediction: a literature survey, Covenant J. Inf. Commun. Technol. 5 (2017), 1–28.
[41] Y. Yoon, N.R. Jo, S.D. Lee, Forecasting algorithm using an improved genetic algorithm based on backpropagation neural network model, J. Korean Data Inf. Sci. Soc. 28 (2017), 1327–1336.
[42] Q. Cai, D. Zhang, W. Zheng, S.C. Leung, A new fuzzy time series forecasting model combined with ant colony optimization and auto-regression, Knowl.-Based Syst. 74 (2015), 61–68.
[43] M. Siddique, S. Mohanty, D. Panda, A hybrid forecasting model for prediction of stock value of TATA steel using support vector regression and particle swarm optimization, Int. J. Pure Appl. Math. 119 (2018), 1719–1727.
[44] A.V. Devadoss, T.A.A. Ligori, Stock prediction using artificial neural networks, Int. J. Web Technol. 2 (2013), 42–48.
[45] P.C. Chang, A novel model by evolving partially connected neural network for stock price trend forecasting, Expert Syst. Appl. 39 (2012), 611–620.
[46] Q. Mingyue, L. Cheng, S. Yu, Application of the artificial neural network in predicting the direction of stock market index, in 2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS), IEEE, Fukuoka, Japan, 2016, pp. 219–223.
[47] W. Dai, J.Y. Wu, C.J. Lu, Combining nonlinear independent component analysis and neural network for the prediction of Asian stock market indexes, Expert Syst. Appl. 39 (2012), 4444–4452.
[48] M. Qiu, Y. Song, Predicting the direction of stock market index movement using an optimized artificial neural network model, PloS One. 11 (2016), 1–11.
[49] R. Bisoi, P.K. Dash, A hybrid evolutionary dynamic neural network for stock market trend analysis and prediction using unscented Kalman filter, Appl. Soft Comput. 19 (2014), 41–56.
[50] G.S. Atsalakis, K.P. Valavanis, Surveying stock market forecasting techniques–Part II: soft computing methods, Expert Syst. Appl. 36 (2009), 5932–5941.
[51] X. Zhong, D. Enke, Forecasting daily stock market return using dimensionality reduction, Expert Syst. Appl 67 (2017), 126–139.
[52] H. Gündüz, Z. Çataltepe, Y. Yaslan, Stock daily return prediction using expanded features and feature selection, Turk. J. Electr. Eng. Comput. Sci. 25 (2017), 4829–4840.
[53] A.U. Haq, A. Zeb, Z. Lei, D. Zhang, Forecasting daily stock trend using multi-filter feature selection and deep learning, Expert Syst. Appl. 168 (2021), 114444.