New Generation Computing
https://doi.org/10.1007/s00354-020-00104-0

Enhanced Prediction of Intra-day Stock Market Using Metaheuristic Optimization on RNN–LSTM Network

Krishna Kumar[1] · Md. Tanwir Uddin Haider[1]

Received: 5 March 2020 / Accepted: 21 August 2020

Abstract
Deep Learning provides useful insights by analyzing information especially in the field of finance with advanced computing technology. Although, RNN–LSTM network with the advantage of sequential learning has achieved great success in the past for time series prediction. Conversely, developing and selecting the best computational optimized RNN–LSTM network for intra-day stock market forecasting is a real challenging task as a researcher. Since it analyses the most volatile data, requires to cope with two big factors such as time lag and the large number of architectural hyperparameters that affect the learning of the model. Furthermore, in addition to the design of this network, several former studies use trial and error based heuristic to estimate these factors which may not guarantee the most optimal network. This paper defines the solution to solve the above-mentioned challenging problems using the hybrid mechanism of the RNN–LSTM network integrating with a metaheuristic optimization technique. For this, a two-hybrid approach namely RNN–LSTM with flower pollination algorithm and RNN–LSTM with particle swarm optimization has been introduced to develop an optimal deep learning model to enhance the intra-day stock market prediction. This model suggests a systematic method which helps us with an automatic generation of optimized network. As the obtained network with tuned hyper parametric values-led towards a more precise learning process with the minimized error rate and accuracy enhancement. In addition, the comparative results evaluated over six different stock exchange datasets reflect the efficacy of an optimized RNN–LSTM network by attaining maximum forecasting accuracy approximately increment of 4–6% using the metaheuristic approach.

Keywords Intra-day stock market · Deep learning · Recurrent neural network (RNN) · Long short term memory (LSTM) · Flower pollination algorithm · Particle swarm optimization

✉ Krishna Kumar
krishna.cspg16@nitp.ac.in

[1] Department of Computer Science and Engineering, National Institute of Technology Patna, Patna, India

Published online: 05 September 2020

Introduction

Data Science provides convenient perceptions by exploring information with the help of modern computing technology. This advanced technology favors the most complex domain of financial time series prediction by providing powerful techniques to adapt the non-linear correlation and effectively adjust the infrequently changeable environment. Among these deep learning techniques, the RNN–LSTM network has succeeded well in time series prediction [1]. The key advantage of this network is to have a memory cell and long term dependency processing ability which supports sequential learning. Despite their great success, it has always been a real overhead for the scientist to cope with the architectural factors of this network. Since many of these hyperparameters are very sensitive to the model building process as if the architectural factors are not appropriate it directly degrades the performance metrics of the predictive model [2]. The adverse effects of these sensitive hyperparameters say lag time (time steps), number of hidden layers, number of neurons in the hidden layers, batch size, and number of epochs leads towards imprecise model building process with a less predictive ability such as decreases the accuracy, increases the time complexity, causes the model to overfit to the training dataset.

The complication of developing the RNN–LSTN network is very adaptive especially in the field of intraday stock market forecasting. The design problem of this network to perform precise learning can be taken care of by these important hyperparameters. They are favorable to analyze the most volatile data that changes very frequently and needs to be adjusted dynamically to identify market behavior. The intention of tuning these values is as follows: first, the time lag: although stock data reflects the entire stock market behavior in the form of market trend and momentum in their past, present, and upcoming future. But still, accurate prediction can be made by opting for the appropriate lag time (time steps) so that it can be able to capture conditional dependencies between successive periods by discarding the outdated data and keeping the most recent sufficient informative data. In addition, the disadvantage of many lagged values increases the dimensionality problem, which causes the deep model to overfit as well as difficult to train. Second, the number of hidden layers: a network with too few or too many layers can degrade the model performance when working with the test dataset. As it allows the model to overfit to the training dataset which learns the training data but it doesn't have the potential to generalize the new unseen data. Third, an inappropriate number of neurons in the hidden layers has a tremendous influence on the final output which must be carefully considered. For instance, the network with very few neurons fails to adequately detect the signals in a complicated dataset which causes underfitting. Likewise, the network with too many neurons leads to overfitting, since it possesses high information processing capacity which in turn fails to train all the neurons in the hidden layer with the limited amount of information present in the training set. In addition, If at all, we provide a sufficient amount of training data to an enormous amount of neuron, there is yet another problem to cope with as the training time exceeds to a limit that it becomes impossible to train the network. Fourth, the batch size impacts learning significantly: a network with too large or too low batch size shows a negative effect on the precision of the learning during training as it decreases the stochasticity of the gradient descent.

Finally, the number of epochs: as these networks do not have an advantage of early stop which avoids the model to overfit the training data with too many epochs. So training the model with much more than a sufficient number of epochs helps the model to memorizes the data rather than learning the data.

To resolve this topology challenge of the model, many studies have adopted a trial and error based heuristic approach to estimate the architectural factors of the LSTM network. There are many thumb's rules methods adopted by the researchers that provide a starting point for the selection of architectural factors of the network and ends with the trial and error based searching process which may not guarantee an optimal solution. As the rule states that the no. of a neuron can be computed using either method, firstly it must lie in between the size of the input and output layer, secondly, it should be the summation of two-third of the input layer size to the output layer size. Apart from this, there is another method that claims it should be less than twice the input layer size. Thus, this approach may not provide an optimized deep learning RNN–LSTM network which can give an optimal solution.

To overcome this approach of trial and error based, there are different traditional optimization algorithms such as grid search and random search which helps to find a hyper-parametric setup combination of the network. But these algorithms fail to tackle and yields an inappropriate hyper-parametric combination with the complicated dataset. As the knowledge acquisition is not generalized, further these interventions turn the overall optimization process into something with more resemblance to an adaptive sequential algorithm.

Recently researchers have been exploring new mechanisms to resolve optimization problems as the traditional optimization algorithm was not very precise in solving complex problems. In addition, the gradual development of metaheuristic search strategies has led to a huge advancement in the domain of computational optimization. The reason behind the popularity of nature-inspired algorithms is to adopt the functionality of biological systems such that it can effectively adjust to a constantly volatile environment.

Particle swarm optimization (PSO) intelligence is all about the collective behavior of animal societies and social insect colonies. Former research scientists have put their effort into this optimization strategies and claimed it as optimistic designing algorithms or distributed problem-solving devices [3]. The fundamental of this learning is to evolve a meta-heuristics approach implementing the insect problem-solving abilities. Here, the effects of the resultant momentum on the particle movement which uses a concept that begins with global search and ends with local search in the last generation allow much rapid convergence on the optimum solution.

Flower pollination search algorithm (FPA) is a newly developed nature-inspired metaheuristic evolutionary optimization method founded based on the pollination process of flowering plants. It seems to be simpler to implement and also yields improved performance to converge on the optimum solution. There are several features of FPA and the most effective one is the Global pollination rule which uses the concept of levy flights, to imitate the travel pollinators in the process of carrying pollen grain. This stage replicates the mobility of the pollen by living agents around flowers to enhance their reproductive efficiency. In addition, the implementation of another productive feature known as local pollination which adopts the flower constancy behavior within the pollinator, the alikeness among flower species instigating the exploitation phase to be more capable to discover superior solutions. Apart from these features of global pollination and local pollination, FPA using the concept of one key parameter switch probability (*p*) which controls the degrees of explorations and exploitations. The concept of switch probability helps to create a balance between assessments and oppression that are carried out in each generation during the global and local pollination. This functionality allows the algorithm to more likely to escape locally optimal points and yields a globally optimum solution. As a result, FPA provides a layout to the foundation of a robust and dominant technique with skillful functionality to solve the global minima optimization problem. Madasu et al. illustrate that FPA provides a better quality solution and robustness for automatic power system control over the Genetic algorithm [4].

In this paper, we propose the hybrid mechanism of the RNN–LSTM network integrating with metaheuristic optimization techniques to cope with the architectural factors of this network and to select the best topology for the model building. Hereby, two-hybrid approach RNN–LSTM with flower pollination algorithm (FPA) and RNN–LSTM with particle swarm optimization (PSO) has been implemented to develop an optimal deep learning model for the intraday stock market prediction by suggesting a systematic method that helps us with the automatic generation of optimized RNN–LSTM network. Our proposed framework covers up with two research challenges: first, the baseline feature engineering process for important predictor variable selection using the wrapper method. Second, tuning these sensitive hyperparameters: lag time, number of hidden layers, number of neurons in the hidden layers, batch size, and number of epochs for automatic generation of optimized RNN–LSTM network. The performances of forecasting techniques are measured by using several evaluation metrics such as precision, recall, F1-score, and error rate. We tested our proposed approach on six different stock exchange datasets extracted from google finance. The results clearly reveal that both the RNN–LSTM with FPA and RNN–LSTM with PSO based forecasting model outperforms the existing approach by attaining the most optimistic solution with their evaluation metrics.

The rest of the paper is organized as follows. "Literature survey" briefly describes the overview of the literature. "Methods" describes the methodologies that are implemented in this study. "Proposed framework" describes the proposed framework. "Experimental results and analysis" shows the experimental results and analysis. Finally, "Conclusion and future work" summarizes our research work.

Literature Survey

Stock market prediction with several research practices over a decades based on traditional computing and advance computing technology, conveys that although the effective market hypothesis (EMH) emphasizes price changes are an independent aspect in market but still several studies illustrated that it can be forecasted [5].

Traditional approaches that are based on statistical analysis over the historical stock data has been widely used in the earlier studies named as the autoregressive integrated moving average model (ARIMA) and the generalized autoregressive conditional heteroscedasticity (GRACH) model [6]. These techniques lack to develop a precise model with several limitations claimed in most of the studies since they requisite more historic information to encounter statistical assumptions, sequential information processing is the mirror reflection of the stock market behavior but these methods do not support sequential learning [7]. Some other key factors to learn, regarded as non-linear correlation between features, elimination of noisy features and complex dimensionality problems etc.

With these drawbacks, machine learning techniques has been rapidly explored by the scientist in the financial domain [8]. Also among them, artificial neural network (ANN) and support vector machine (SVM) have been widely adopted with great success for forecasting financial time series data. Since these techniques provide the potential to learn complex pattern in data by extracting non-linear correlation between data, investigating the noisy behavior of data [9]. The experimental results show outstanding performances as compared with statistical models for stock return in the proposed work of Yu et al. which uses SVM to construct stock selection and principal component analysis (PCA) to reduce dimensionality problem with most informative financial time series data [10]. Yet again, Gheyas and Smith [11] claimed that these model using individual machine learning approaches always suffers from parametric setup problems which degrades the forecasting model performances.

To overcome this limitation, Patel et al. [12] proposed a hybrid mechanism with two stage scheme using support vector regression to prepare input vector incorporated with random forest and SVR to deploy the forecasting model to predict upcoming market values. Ballings et al. [13] illustrated the advantage of using ensemble methods over individual machine learning classifier based on a comparative study of forecasting model for stock price movement prediction. Ensemble methods such as random forest, adaboost and kernel factory were dominating over benchmark models like ANN and SVM.

With advance computing techniques like deep learning facilitates financial domain exploring the characteristics of extracting useful features automatically during learning process [14]. Ding et al. [15] applied an integrated mechanism of neural network and deep convolutional neural network to examine the effect of model building process for stock price movement prediction. Yoshihara et al. [16] predicted future price movements by examining the effect of grouping of the restricted Boltzmann machine learning and deep belief network integrating together to deal with the temporal effect of market data. Sezer et al. [17] deployed deep neural network using the concept of genetic algorithm to optimize the technical indicators parameters and to develop stock trading system for buy-sell-hold predictions.

In recent years, deep learning techniques, RNN including LSTM network provide sequential processing capability with the unique characteristics of the memory cell that allows to store stimulated temporal patterns from data that has always been the big factor for time-series analyses to enhance the model predictive ability. Fischer and Krauss [18] deployed RNN–LSTM network for directional movement prediction. They claimed based on the comparative results of memory classifier (LSTM model) over memory-free classifiers (random forest, deep neural network, logistic regression) that LSTM model outperformed the other comparative models. In addition, these RNN–LSTM network are very sensitive in the financial domain and shows the adverse effect with inappropriate parametric setup of the network. Hsieh et al. [19] examined the mechanism of using artificial bee colony to optimize the connection weight of an incorporated RNN for market prediction. Rather et al. [20] proposed hybrid mechanism to predict stock returns using a genetic algorithm for optimal weight selection of RNN and two linear models (ARIMA) and exponential smoothing. Chung and Shin [2] proposed a systematic approach to optimize the RNN–LSTM network using genetic algorithm for stock market forecasting. However, literature remains limited and the overhead of optimizing hyperparameter of LSTM network still remains an open challenge for the researcher to provide a systematic approach.

Methods

In our work, we have used the RNN–LSTM network which helps to perform sequential learning for time series prediction. The processing of the RNN–LSTM network has been discussed as follows:

RNN–LSTM Network

An RNN is a special type of ANN which support to process sequences of inputs using the concept of internal feedback between neurons. Since the directed connections are established between the units which provide the unique feature of inputting the output from the previous step to the current step. This RNN comprises of the input layer, hidden layers, output layers such that the hidden state can store some information about a sequence to make use of sequential information [21]. However, RNN has a limitation of learning short term dependencies because it suffers from a vanishing gradient problem by increasing the time step in length. LSTM designed by introducing three gate concepts in the memory cell which overcomes the challenges of preserving long term dependencies and controlled flow over the gradient. LSTM designed by introducing three gates concept in the memory cell which overcomes the challenges of preserving long term dependencies and controlled flow over the gradient. Each memory cell consists of three gates: forget gate (f_t), input gate (i_t), and an output gate (o_t) which are responsible for updating its cell state (s_t) at every time step t with new input x_t. The computational steps to update its cell state by LSTM block are as follows.

Processing start with forget gate using sigmoid function to decide for the information which has to be preserved and to be thrown away from previous cell state s_{t-1}. The forget gate contains current input x_t at time step t and output h_{t-1} at time step $(t-1)$ with the activation values f_t between range 0 (completely discard) and

1 (completely keep) where $W_{f,x}$, $W_{f,h}$ and b_f are parameters of the LSTM unit, f_t defines activation value for this gate.

$$f_t = \text{sigmoid}(W_f.[h_{t-1}, x_t] + b_f) \tag{1}$$

In the second step, the input gate decides which value will be updated and compute the new value s'_t to be added in the cell using a sigmoid function and tanh function respectively, where $W_{f,x}$, $W_{f,h}$ and b_f are parameters of the LSTM unit, it defines activation value for this gate, s'_t is a vector of cell candidate.

$$i_t = \text{sigmoid}(W_i.[h_{t-1}, x_t] + b_i) \tag{2}$$

$$s'_t = \tan h(W_s.[h_{t-1}, x_t] + b_s) \tag{3}$$

In the third step, the results of prior two steps are used to compute the new value of cell state s_t using the Hadamard product (o), where s_t is a vector of cell state.

$$s_t = (f_t \, o \, s_{t-1}) + (i_t \, o \, s'_t) \tag{4}$$

Finally, the output gate used sigmoid function and followed by computing Hadamard product of the results from the previous step value using tanh function with the results of the output gate, where o_t defines activation value for this gate, b_o are parameters of the LSTM unit, h_t is a vector of the output.

$$o_t = \text{sigmoid}(W_o.[h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = (o_t \, o \tan h(s_t)) \tag{6}$$

Proposed Framework

In this section, we describe the proposed framework processing for intra-day stock market time series forecasting using the RNN–LSTM model as shown in Fig. 1. The key purpose of this research is to enhance the model building process by investigating the two most important aspects named as baseline feature engineering for important predictor variable selection using the wrapper method and optimization of most sensitive hyperparameters of RNN–LSTM network for automatic generation of optimized RNN–LSTM network. Accordingly, we systematically control these aspects of the model building process that significantly helps us to enhance the predictive ability of the time series forecasting model. This study suggests a fusion mechanism that incorporates the RNN–LSTM network with an optimization technique to search for an optimistic model to forecast the next value movement of time series for the intra-day stock market.

The process of the proposed framework comprises three processing modules namely, data preparation, hyper-parameter tuning RNN–LSTM model, and optimized RNN–LSTM network model evaluation. In addition, the first processing

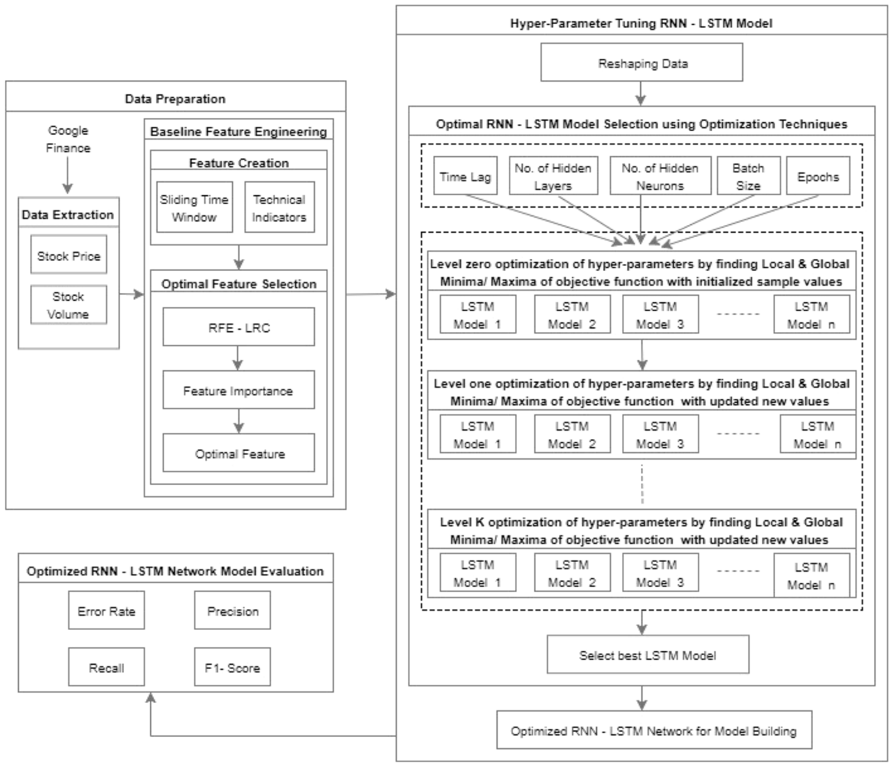

Data Preparation
Google Finance
Data Extraction
Stock Price
Stock Volume
Baseline Feature Engineering
Feature Creation
Sliding Time Window
Technical Indicators
Optimal Feature Selection
RFE - LRC
Feature Importance
Optimal Feature
Hyper-Parameter Tuning RNN - LSTM Model
Reshaping Data
Optimal RNN - LSTM Model Selection using Optimization Techniques
Time Lag
No. of Hidden Layers
No. of Hidden Neurons
Batch Size
Epochs
Level zero optimization of hyper-parameters by finding Local & Global Minima/ Maxima of objective function with initialized sample values
LSTM Model 1
LSTM Model 2
LSTM Model 3
LSTM Model n
Level one optimization of hyper-parameters by finding Local & Global Minima/ Maxima of objective function with updated new values
Level K optimization of hyper-parameters by finding Local & Global Minima/ Maxima of objective function with updated new values
Select best LSTM Model
Optimized RNN - LSTM Network for Model Building
Optimized RNN - LSTM Network Model Evaluation
Error Rate
Precision
Recall
F1- Score

Fig. 1 Proposed framework for optimization

module data preparation comprises of two submodules named as data extraction and baseline feature engineering. Here, the data extraction process collects the raw data say market price and volume of stock companies from google finance. Followed by the execution of the baseline feature engineering process which has the key advantage to assess the non-linear correlation among the set of features and to eliminate the noisy feature to select the optimal features. In addition, this approach is quite handy for intra-day stock market forecasting since these correlated set of features all together leads towards substantial signal of a price movement while learning the model. Next, the second processing module is a hyper-parameter tuning of the RNN–LSTM model. The job of this module is to tune the hyper-parameter of the RNN–LSTM network in such a way that it automates the searching process using optimization techniques (FPA, PSO)for the RNN–LSTM network. In this phase of implementation, we are tuning the five most sensitive hyper-parameters of the RNN–LSTM network say time lags, number of hidden layers, number of hidden neurons, batch size, and number of epochs by performing level-*k* optimization w.r.t objective function. At the end of the second processing module, we are left behind with an optimized RNN–LSTM network for the model building which is further validated in the third module with the different performance metrics.

Data Preparation

In this module, our research mainly focuses on enhancing the baseline input vector and to provide the best platform to develop the RNN–LSTM model with the ability to give the most optimistic forecasting results. Since several studies have claimed that feature engineering with optimistic feature selection is the key factor for success to develop the precise forecasting model [22]. Hence, the processing has been categorized in two phases namely, data extraction and baseline feature engineering.

Data Extraction

In this study, the research data has been extracted from the google finance database. The advantage of using the google finance database is that it manages all the stock exchange data and also keeps the data in a well-organized manner which helps researchers to track down the relevant data as per their research work requirements. Intra-day forecasting performs analysis over mainly two factors such as market trend and momentum to identify the market behavior. Since, these are the two key parameters that reflect the market behavior of the stock in the past, present, and upcoming future. In addition, these two aspects have the most dynamic values which change very continuously every hour, days, or weeks. Hence, we require the most recent trending data to identify the upcoming future market behavior [8]. Our study involves the dataset of 18 listed companies from six stock exchanges as shown in Table 6 comprises of the most recent 90 days (22 March 2018 to 22 May 2018) stock price and volume fluctuation with a time resolution of 1-min along with date and time.

Baseline Feature Engineering

It has always been a real threat to find the non-linear correlation among the set of features and to select the optimal combination of feature set for intra-day stock market forecasting. In addition, the forecasting model fails to adapt market trends and momentum with the combination of noisy features due to the volatile nature of the market behavior. Thus, the baseline feature engineering processing unit all together helps us to provide the baseline to the RNN–LSTM network designing module by discovering the solution to the mentioned threat of isolating the non-linear correlation, eliminating the noisy features, and also select the optimal combination of feature sets. But, the same experiment carried out using RNN–LSTM deep network shows a negative impact due to its enormous number of parameters that have to be learned, which results in very expensive and time-consuming computation. Here, the processing has been categorized in two phases namely, feature creation and optimal feature selection.

Feature creation In this section, prior to starting the processing, we need to transform the extracted raw data in the form of relevant informative attributes such that the market analysis which identifies the hidden pattern can be taken care of in a more precise custom. Here, 1-min interval data is converted into 10-min time resolution interval data since interval with data fluctuation of 1-min time resolution does not provide sufficient conclusive information to identify the pattern and also fails to lead us with an optimistic forecasting model. After conversion to 10-min time resolution data (one interval) as shown in Fig. 2, we calculate informative attributes for each interval such as interval open price (say 1st min price), interval close price (10th min price), interval low (lowest price of that interval say between 1 and 10 min data), interval high (lowest price of that interval), interval average (average price of that interval), volume (10th min volume), price (10th min price), time (interval closing time say 10th min) and date (interval closing data).

Further, these mentioned attributes are used to create features that contain lots of information to train the model to perform the forecasting. The feature creation task has been accomplished using a sliding time window approach with the list of technical indicators. These technical indicators are mathematically calculated using the mentioned informative attributes which perform technical analysis that helps to identify the market behavior in the form of market trend and momentum for future prediction.

Sliding time window: Sliding time window approach has the key advantage to assess the most recent trending data by choosing an optimal period (appropriate number of an interval as one period). Since it helps us to capture much effective information for feature creation by discarding the outlier's data to perform the technical analysis through which the exact behavior of the price fluctuation can be assimilated. Intra-day forecasting expertise their processing with a window size of most recent 14 intervals as one period which is suggested as an ideal period by the researchers in their studies. Subsequently, the former studies claim that window size with more than 14 intervals is said to be outdated data and window size with less than 14 intervals does not have sufficient information to analyze the previous trend and market fluctuation which degrades the model accuracy [8]. Hence, in our study we have opted for the sliding window size of 14 intervals as one period with each interval containing a time resolution of 10 min as shown in Fig. 2.

The working of the sliding time window approach has been shown in Fig. 3 respectively to calculate the technical indicators with most recent trending data. Here, after every iteration, we are continuously updating the window values by sliding the one window to append next interval value and discard the most outdated value from the window to preserves the window size with 14 intervals values.

Technical indicators: Technical indicators [23] are heuristic or mathematical calculations based on the study of past trading activity, primarily price changes of a security and trading volume to accomplish technical analysis for forecasting the future price movement. Here, in our study we have calculated these statistics based

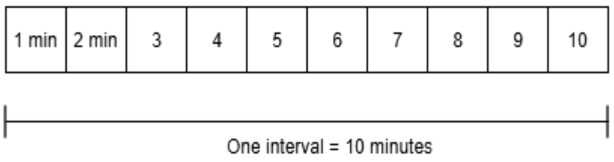


Fig. 2 One interval with time resolution of 10 min [21]

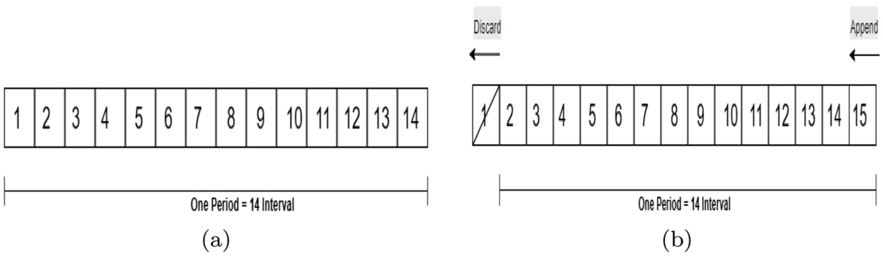


Fig. 3 Sliding time window processing after every iteration to preserves 14 intervals with most trending data [21]

on intra-day trading strategies, obtaining the set of technical indicators used by skilled traders that can be considered as best predictor variables to train the forecasting model and to improve the predictive ability as illustrated in the earlier research [21]. The computation of listed technical indicators (14 features) incorporated in our study are named as money flow index (MFI), simple moving average (SMA), stochastic oscillator ($\%K$ and $\%D$), intra-day momentum index (IMI), relative strength index (RSI), rate of change (ROC), commodity channel index (CCI), force index (FI), average true range (ATR), ease of movement (EMV), directional movement index (DX), exponential moving average (EMA) and price has been shown in Table 1.

Optimal Feature Selection In this phase of processing, we are aiming to explore the optimistic set of combinations between the features to identify the non- linear correlation among the features, eliminate the noisy features that lead towards the imprecise model development process. Hence, to assess the optimal feature subset to improve the predictor accuracy, we are using a wrapper method namely, recursive feature elimination with logistic regression classifier (RFE-LRC) that follows a greedy optimization algorithm approach. The key advantage of using the RFE algorithm is that it directly works with the accuracy measure of a given classifier based on feature importance factor to find the optimal feature subset. In addition, the former studies show the comparisons among three machine learning algorithms that have achieved great success in the past for stock market prediction and claimed that RFE-LRC is best suited for binary classification with high accuracy measure [21].

Recursive feature elimination with logistic regression classifier (RFE-LRC): The optimal feature selection process has been incorporated in the feature engineering process because experimenting the same with RNN–LSTM deep network which comprises a large number of parameters that have to be learned resulting in very expensive and time-consuming computation.

Preceding module produces the generalized features that can look over all the aspects of the market behavior to identify the hidden pattern based on market trends and momentum. Subsequently, prior to the training, an external estimator assigns a certain weight to all the features. Initially, the LRC model is trained with all the generalized features, followed by assigning rank (feature importance value) to each of

Table 1 List of technical indicators and mathematical formula [21]

Technical indicators	Formula
Simple moving average	$\text{SMA} = \frac{(\text{Interval Close Price}_n + \text{Interval Close Price}_{n-1} + \cdots + \text{Interval Close Price}_i)}{n}$
Stochastic oscillator	$\%K = \frac{(\text{current\_close} - \text{Min\_Low})}{(\text{Max\_High} - \text{Min\_Low})} * 100$
	$\%D = 3 - \text{period SMA of } \%K$
Intraday momentum index	$\text{IMI} = 100 * \frac{\text{Gains}}{(\text{Gains} + \text{Losses})}$
Relative strength index	$\text{RS} = \text{Average Gain}/\text{Average Loss}$
	$\text{RSI} = 100 - \frac{100}{(1+\text{RS})}$
	$\text{Typical price} = (\text{Interval high} + \text{Interval low} + \text{Interval Close})/3$
	$\text{Money Flow} = \text{Typical Price} * \text{Volume}$
Money flow index	$\text{MFR} = \frac{(\text{Positive Money Flow})}{(\text{Negative Money Flow})}$
	$\text{MFI} = 100 - \frac{100}{(1+\text{MFR})}$
Rate of Change	$\text{ROC}(5) = \frac{(\text{Interval Close Price}_i - \text{Interval Close Price}_{i-5})}{\text{Interval Close Price}_{i-5}} * 100$
Commodity channel index	$\text{Typical price} = (\text{Interval high} + \text{Interval low} + \text{Interval Close})/3$
	$\text{CCI} = \frac{(\text{Typical Price} - n \text{ interval SMA of Typical Price})}{(0.015 * \text{Standard deviation of Typical Price})}$
Force index	$\text{RFI} = (\text{Interval Close Price}_i - \text{Interval Close Price}_{i-1}) * \text{Volume}$
	$\text{FI} = [((\text{RFI}(14) - \text{SMA}(13) * 0.1818) + \text{SMA}(14)]$
Average true range	$\text{TR} = [\max(\text{Interval High, Prev Interval Close}) - \min(\text{Interval Low, Prev Interval Close})]$
	$\text{ATR} = \text{TR}/n$
	$\text{DM} = [(\text{Interval High} + \text{Interval Low}) - (\text{prev Interval High} + \text{prev Interval Low})]/2$
Ease of movement	$\text{WR} = (\text{Volume}/100,000,000)/(\text{Interval High} - \text{Interval Low})$
	$\text{EMV} = \text{DM}/\text{WR}$
Directional movement index	$\text{DX} = \frac{(\text{Positive Directional Indicator}) - (\text{Negative Directional Indicator})}{(\text{Positive Directional Indicator}) + (\text{Negative Directional Indicator})} * 100$
Exponential moving average	$\text{EMA}_i = [(\text{Interval Close Price}_i - \text{EMA}_{(i-1)}) * 0.1818] + \text{EMA}_{(i-1)}$

the features based on maximum accuracy achieved by the classifier. Then, the least important features with a maximum rank number are discarded from the feature set and a new feature subset is generated. Again the new LRC model has been developed with these new feature subset, followed by allocating the rank based on classifier accuracy if the previous accuracy is enhanced. Also, if the newly developed model accuracy does not improve then the eliminated feature is included back to the feature set and the second least important feature is being discarded from the feature set. With this feature subset the new model is developed, accuracy measured, rank assigned, feature discarded, new feature subset generated and model developed. This process is continuing until there is no feature to eliminate from the feature set. Finally, the feature set that provides the maximum accuracy is said to be an optimistic feature set which is used as an input vector to train the RNN–LSTM network.

Hyper-Parameter Tuning RNN–LSTM Model

This hybrid mechanism of the RNN–LSTM network integrating with optimization technique is the backbone of our research work, since it resolves the limitation of processing with trial and error based heuristic estimation of architectural hyper-parameter. This mechanism helps us suggesting a systematic tactic to develop an optimal deep learning model by automatic generation of optimized RNN–LSTM network. Here, we are exploring the five most sensitive hyper-parameters of this network that are having a maximum impact to develop a precise model for intra-day stock market forecasting. The list of five hyper-parameters is time lag, number of hidden layers, number of hidden neurons, batch size, and epochs.

The intention of tuning these hyper-parameters are first, the time lag: the appropriate time lag (time steps) help the model to capture conditional dependencies between successive periods by discarding the outdated data. In addition, many lagged values increase the dimensionality problem, difficult to train, and overfit the model. Second, the number of hidden layers: a topology with too few or too many number of layers allows the model to overfit, learns the training data which in response fails with test data to generalize the new unseen data. Third, the number of hidden neurons: a network with very few neurons fails to detect the signal in a complicated dataset, causes the model to underfit. In addition, the network with too many neurons increases the information processing capacity and the limited amount of information in the training set fails to train all the neurons in the hidden layer, causes the model to overfit and sufficient training data increases the training time to the point that is impossible to train the network. Forth, the batch size: a network with too large or too low batch size decreases the stochasticity of the gradient descent causes to show a negative effect on the forecasting model during training. Finally, the number of epochs: the disadvantage of too many epochs in the topology is that it does not support early stop which causes the model to overfit the training data and helps the model to memorizes the data rather than learning.

Optimal RNN–LSTM Model Selection Using Optimization Techniques

The preceding module of data preparation allows us to find an optimal feature set which is used as an input vector to develop an optimal RNN–LSTM network by tuning the hyper-parameters. Subsequently, the next level processing starts with reshaping the input vector data by performing normalization, splitting the train-test data along with the target value.

Table 2 Characteristics of different optimization techniques

Attributes	PSO	GA	FPA	ABC
Initial population for convergene	Mild sensitive	Very sensitive	Not sensitive	Sensitive
Convergence	Faster convergence	Slower convergence	Mild convergence	Slow convergence
Exploration and exploitation over search space	Does not maintain balance, it may converge at local minima	Does not maintain balance, prematuraly converge into loca optima	Switch probability maintain balance, helps to converge globally and locally	Does not maintain balance, poor local seraching ability
Time consuming	Less time consuming	More time consuming	Moderate time consuming	More time consuming

In this phase of implementation, we are performing level-k optimization w.r.t objective function such that it minimizes the error rate, maximizes the accuracy value and automatically searches for the optimal RNN–LSTM network using optimization techniques. There are several successful optimization techniques proposed in the past that provide a wide diversity of feasible solutions for solving a complex problem such as artificial bee colony (ABC), genetic algorithm (GA), flower pollination algorithm (FPA), and particle swarm optimization (PSO). The characteristics of these optimization techniques with problem-solving ability have been described in Table 2. In addition, our research work relies upon two challenges such as the generation of the random initial population, proper exploration, and exploitation over search space for the convergence of an optimum solution. Therefore, we have experimented with the proposed framework with two newly developed nature-inspired meta-heuristic optimization techniques namely, flower pollination algorithm (FPA) and particle swarm optimization (PSO). The advantage of opting for these two algorithms is as follows: PSO has achieved great success in the computational domain to search for the optimum solution. This algorithm allows faster convergence on the optimum solution with the ability to move all the particles through global search and end with local search in the final generation. FPA is a robust and powerful technique to investigate the problem with the key parameters p (switch probability) which makes the algorithm simpler to implement and faster to reach the optimum solution. This key parameter p (switch probability) controls the degree of explorations and exploitations by sustaining the balance between the local pollination and global pollination carried out in each generation. Thus, the algorithm is more likely to escape locally optimal points and yields a globally optimum solution.

Optimal RNN–LSTM model selection using FPA In run-through, aggregation of this mechanism and randomized global pollination using levy flight distribution led to the foundation of a remarkable optimization system, which rapidly explores the given objective function scenery to acquire to its optimum domain, concurrently avoiding the process from getting trapped with local limits.

Figure 4 shows the workflow of the FPA mechanism to automatically search for an optimized RNN–LSTM model that gives an optimal solution. Here, prior to the starting point, we need to set the preliminary insight parametric value of FPA such as population size (n), switching probability (p), scaling factor for step size (γ), uniform distribution (E denote ϵ), levy-flight based step size ($L(\beta)$). Population size (n) is the number of individual solutions to an optimization problem w.r.t an objective function that is produced at each level of optimization. More likely an ideal population size of $n = 25$ is opted by the researchers since it produces an optimal outcome and an increase in the value of n hardly shows any change in the obtained optimal solution rather than only increasing the time complexity. Switch probability (p) controls the degrees of exploration and exploitation and helps to preserve the balance between local and global pollination. It is experienced from the results that $p = 0.8$ is an optimistic value that offers a minimum value of fitness function. As if the value of p increases, it led to explores the search space globally and escape the locally optimal points which more likely compromises the overall search performance and degrades the solution. Other parameters based on experimental analysis

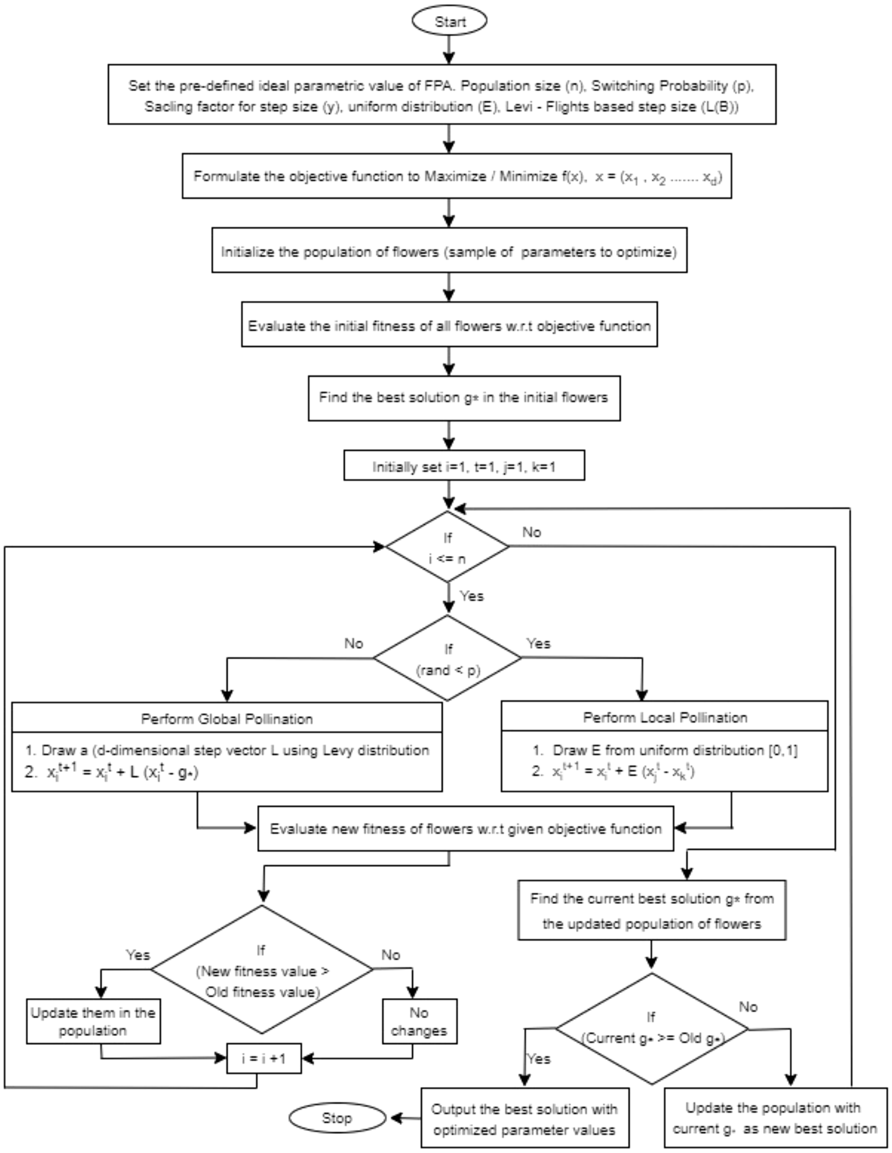

Start
Set the pre-defined ideal parametric value of FPA. Population size (n), Switching Probability (p), Sacling factor for step size (y), uniform distribution (E), Levi - Flights based step size (L(B))
Formulate the objective function to Maximize / Minimize f(x), x = (x1 , x2 ...... xd)
Initialize the population of flowers (sample of parameters to optimize)
Evaluate the initial fitness of all flowers w.r.t objective function
Find the best solution g* in the initial flowers
Initially set i=1, t=1, j=1, k=1
If i <= n
No
Yes
If (rand < p)
No
Yes
Perform Global Pollination
1. Draw a (d-dimensional step vector L using Levy distribution
2. xi t+1 = xi t + L (xi t - g*)
Perform Local Pollination
1. Draw E from uniform distribution [0,1]
2. xi t+1 = xi t + E (xj t - xk t)
Evaluate new fitness of flowers w.r.t given objective function
If (New fitness value > Old fitness value)
Yes
No
Update them in the population
No changes
i = i +1
Find the current best solution g* from the updated population of flowers
If (Current g* >= Old g*)
Yes
No
Output the best solution with optimized parameter values
Update the population with current g* as new best solution
Stop

Fig. 4 Workflow of FPA to tune hyper-parameters and discover optimized RNN–LSTM network

has been chosen as scaling factors (γ) for step size = 0.1, uniform distribution (ϵ) uses randi function to generate integer value with range 0 to 1, levy-flight based step size (strength of pollination) $L(\beta) = 0.5$.

The formulation of an objective function is to minimize the error rate value $f(x), x = (x_1, x_2 \ldots x_d)$ where x_1, $x_2 \ldots x_d$ are d dimensional hyper-parameter values to be optimized. Thereafter, initialize the population of flowers $f(x_i)$, where

$i = 1, 2 \dots n$ (n sample data that is initially generated using random function with below mentioned specified range of hyper-parameter values as combination set to perform level k optimization as shown in Table 4). As the former studies suggest the best min-max range for these hyper-parameter value are timelag = [1, 30], hidden layers = [1, 8], hidden neurons = [10, 150], batch size = [1, 30] and epochs = [50, 500].

Subsequently, with these initial n sample data, we are developing the n-LSTM model as shown in Fig. 1 in level 0 optimization processing. Then, the initial fitness value of each solution (LSTM model) $f(x_i)$ is being evaluated by calculating its corresponding objective function value (error rate) as shown in Table 3. Among these n-LSTM model developed in level zero optimization, the best solution g^* is recognized. Thereafter, we perform level zero optimization of hyper-parameters by finding local and global minima with these sample data as shown in Fig. 4. Initially set the values of $i = 1$, $t = 1$, $j = 1$ and $k = 1$. If the condition ($i \leq n$) holds true and if (rand $< p$) holds true then perform local pollination, if (rand $< p$) holds false then perform global pollination where rand is the probabilistic value of the random solution and p is switch probability = 0.8. The probabilistic value for random solution $f(x_i)$ is computed as the ratio of obtained fitness value (say $f(x_i)$) to the best solution g^*.

To perform local pollination process, the new solution is generated by random number ϵ, ϵ in [0, 1] from uniform distribution as follows:

$$X_i^t + 1 = x_i^t + \epsilon(x_j^t - x_k^t) \tag{7}$$

where x_i^t, x_j^t, x_k^t are solution from different lowers of same plant.

To perform a global pollination process, the new solution is generated by step vector L using levy distribution as follows:

$$X_i^t + 1 = x_i^t + L(x_i^t - g^*) \tag{8}$$

where L is levy flight, $L > 0$, s >> $s_0 > 0$ and calculated as follows:

$$L \sim \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}} \tag{9}$$

Subsequently performing either global pollination or local pollination as per conditions satisfied, evaluate the fitness value of newly generated solution w.r.t to objective function error rate. If the condition (new fitness value > old fitness value) holds true then update them in the population along with changed hyper-parameter values, else no changes are required in the population and their hyper-parameter values. Increment the count of i by one and overthrow the flow control to the first condition if ($i \leq n$), repeat this processing until this condition holds false. In addition, if the condition ($i \leq n$) holds false then we can say that level zero optimization has been completed by updating all the n developed LSTM model with their new hyper-parametric value as per the condition of if (new fitness value > old fitness value) holds true or false. Finally, again the current best g^* is recognized and checked with the condition (current g^* < old g^*) to stop the optimization process. If this condition

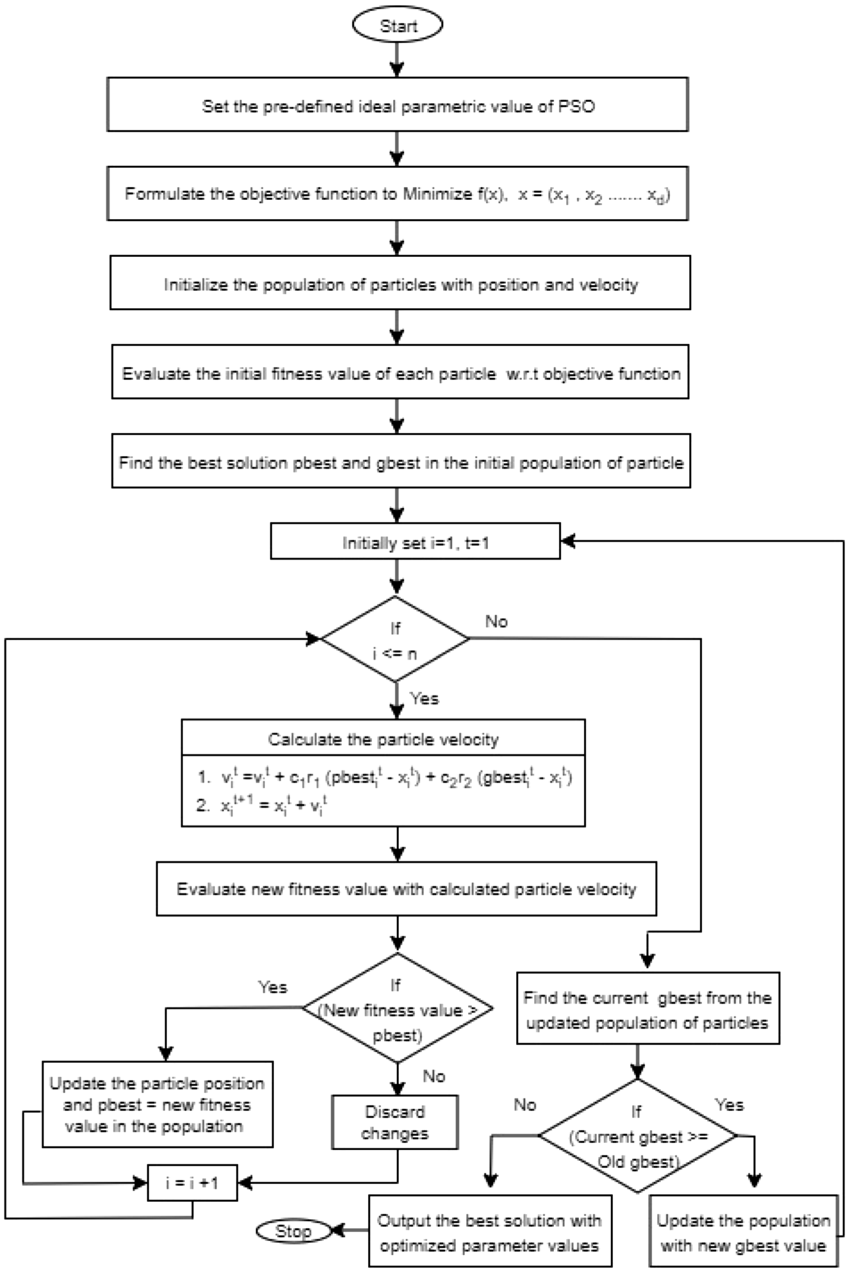

Start
Set the pre-defined ideal parametric value of PSO
Formulate the objective function to Minimize f(x), x = (x_1 , x_2 .... x_d)
Initialize the population of particles with position and velocity
Evaluate the initial fitness value of each particle w.r.t objective function
Find the best solution pbest and gbest in the initial population of particle
Initially set i=1, t=1
If i <= n
No
Yes
Calculate the particle velocity
1. $v_i^t = v_i^t + c_1 r_1 (pbest_i^t - x_i^t) + c_2 r_2 (gbest_i^t - x_i^t)$
2. $x_i^{t+1} = x_i^t + v_i^t$
Evaluate new fitness value with calculated particle velocity
If (New fitness value > pbest)
Yes
No
Update the particle position and pbest = new fitness value in the population
Discard changes
i = i +1
Find the current gbest from the updated population of particles
If (Current gbest >= Old gbest)
No
Yes
Output the best solution with optimized parameter values
Update the population with new gbest value
Stop

Fig. 5 Workflow of PSO to tune hyper-parameters and discover optimized RNN–LSTM network

holds true, then the output with the current best g^* is selected and suggested as an optimal solution with an optimized RNN–LSTM network among the n generated model. Otherwise, if the condition holds false, then we set the value of $i = 1$, update the population old g^* with current g^* as the new best solution and pass the control to the first condition check of $(i \le n)$ by performing the next level 1 optimization process with the newly updated population in a similar adopted fashion. This process continues to level k optimization until the check condition (current $g^* <$ old g^*) holds true. At the finishing level, this systematic method led us with an automatic generation of an optimized RNN–LSTM network for model building that can guarantee an optimal solution.

Optimal RNN–LSTM model selection using PSO: The workflow of the PSO mechanism which assistance us to automatically search for the best solution with an optimized RNN–LSTM model as shown in Fig. 5. Here, prior to starting point, we need to set the pre-defined ideal parametric value of PSO based on experimental analysis. It is experienced from the results, suggested in the former studies that the population size (n) =25, c_1 and c_2 are learning factor generally $c_1 = c_2 = 2$, r_1 and r_2 are random numbers between (0, 1) are an appropriate choice since it offers most optimistic value of fitness function.

In a similar fashion as discussed above, we are formulating an objective function, Initializing the population of particles with hyper-parameter values as position and velocity equals zero, developing the n-LSTM model in level-0 optimization processing, computing the initial fitness value of each particle $f(x_i)$. Thereafter, among these n-LSTM model as developed in level zero optimization, the best solution pbest and gbest are recognized. Here, pbest is defined as the best solution attained by individual particle so far based on fitness function, and gbest is known as the global best solution attained by any particle in the population, tracked by particle swarm optimizer. Later, we perform level-0 optimization of hyper-parameters by finding local and global minima with these sample data as shown in Fig. 5. Here, rather than using the concept of switch probability, we are determining the new particle position based on the computational value of particle velocity that is calculated using the formula as follows:

$$v_i^t + 1 = v_i^t + c_1 r_1(\text{pbest}_i^t - x_i^t) + c_2 r_2(\text{gbest}_i^t - x_i^t) \tag{10}$$

$$x_i^t + 1 = x_i^t + v_i^t + 1 \tag{11}$$

Further, again with this calculated new particle position, we evaluate the new fitness value which is mapped with the given conditions as shown in the figure that performs useful decision making for the updation of population hyper parametric values, pbest and gbest values at each level of optimization. Subsequently, followed by a similar iteration process as discussed in the FPA mechanism for each level of optimization as per the given conditions of the PSO algorithm. At the finishing level, this systematic method led us with an automatic generation of an optimized RNN–LSTM network for model building that can guarantee an optimal solution.

Optimized RNN–LSTM Network Model Evaluation

The most commonly used metrics to evaluate the performance of the forecasting model in the finance domain are error rate, precision, recall, and F1-score [21]. The error rate is the evaluation between the actual outcome and predicted outcome which is calculated by subtracting one to the value computed as a ratio of the sum of false positive and false negative with the sum of true positive, true negative, false positive, and false negative known as accuracy. Precision is the ratio of true positive with the sum of a true positive and false positive. The recall is the ratio of true positive with the sum of a true positive and false negative. F1-score is the hybrid measure of both precision and recall, known as an alternative to accuracy measure. The evaluation metrics are calculated as follows:

$$\text{Error rate} = 1 - \frac{(\text{FP} + \text{FN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})} \tag{12}$$

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \tag{13}$$

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \tag{14}$$

$$\text{F1-score} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})} \tag{15}$$

where true positive (TP) occurs when both actual outcome and predicted outcome are the same, true negative (FN) occurs when both differ such as the actual outcome is yes but the predicted outcome is no, false positive (FP) occurs when both differ such as actual outcome is no but the predicted outcome is yes, false negative (FN) occurs when both differ such as actual outcome is yes but predicted outcome is no.

Experimental Results and Analysis

All the investigations were performed on Intel® Core™ i5-7200U, CPU@2.50 GHz, Memory (RAM) 8GB processor. We opted for Python language along with Keras library as a front end and TensorFlow as a backend on our application domain problem since it provides the best implementation and simulation platform.

In this section, the experimental results of our proposed approach have been shown using RFE-LRC for an optimistic feature selection set along with their performance measure. Further, the enhanced results achieved during the optimization of the RNN–LSTM network have also been evaluated. Hence, the summary of the hyper-parametric values updation based on minimum fitness value (error rate) at each iterative level, say level-0 optimization enhanced in level-1, followed by level-2, ends with level-*k* optimization to achieve optimized RNN–LSTM network has been shown using two different optimization techniques named as FPA and PSO. In addition, the results say the plot of adverse effect perceived in error rate during level-k optimization with an inappropriate time lag and hidden layers, a plot of enhanced optimization curve after each level of optimization to generate optimized RNN–LSTM network has been exposed. Also, the plot of the optimized RNN–LSTM network learning curve based on model behavior (train and test data) with the tuned hyper-parametric value obtained after level-k optimization helps us to illustrates that an optimal deep learning model with an optimal solution has been shaped.

This analysis is concluded from the results of six companies namely, Sirius Minerals PLC (LON: SXX), ICICI Bank Ltd (NSE: ICICIBANK), International Business Machine Corp (NYSE: IBM), Solar Industries India Ltd (BOM:532725), Amazon.com (NASDAQ: AMZN) from six distinct stock exchange named as London, National, New York, Bombay, Nasdaq, and Toronto respectively [21]. In addition, the summary table of the 18 listed companies with optimized hyper-parametric values of RNN–LSTM network along with performance measure after the execution of level-k optimization.

Prior to the optimization process, an optimistic feature selection set has been explored using RFE-LRC to improve the predictor accuracy by eliminating noisy features and identifying non-linear correlation among themselves as shown in Table 3. The key advantage of choosing RFE-LRC has been described in "Data preparation". As the given Table 3 clearly shows the results obtained by the baseline feature engineering module with an optimistic feature selection set based on the classifier performance measure. Herewith an example, say company ticker LON: SXX, these five features set = $\{f_6, f_7, f_8, f_{12}, f_{13}\}$ has been considered as optimistic features set that has been assigned rank one, because this combination attains the maximum accuracy error rate = 0.32, precision = 0.69, recall = 0.68 and $F1$-Score = 0.68. Similarly, the results of other individual companies have experimented. The overall results express that though we succeed to produce an optimistic feature set still it fails to develop the precise model. The evaluation metric average values, such as error rate = 0.30, precision =0.71, recall= 0.71 and F1-score = 0.71 claims that the forecasting model deficient to provide us promising result.

Table 3 List of optimistic feature selection set along with evaluation metrics based on RFE-LRC

Company tickers	Optimistic feature selection set	Error rate	Precision	Recall	F1-score
LON:SXX	$\{f_6, f_7, f_8, f_{12}, f_{13}\}$	0.32	0.69	0.68	0.68
NSE:ICICIBANK	$\{f_5, f_6, f_7, f_8, f_9, f_{12}, f_{13}\}$	0.30	0.72	0.71	0.71
NYSE:IBM	$\{f_3, f_5, f_6, f_7, f_{10}, f_{12}, f_{13}\}$	0.29	0.71	0.73	0.72
BOM:532725	$\{f_2, f_4, f_5, f_9, f_{11}, f_{13}\}$	0.29	0.71	0.70	0.71
NASDAQ:AMZN	$\{f_3, f_4, f_8, f_{10}, f_{12}, f_{13}\}$	0.30	0.70	0.70	0.70
TSE:ABX	$\{f_2, f_3, f_5, f_6, f_7, f_{10}, f_{11}, f_{13}\}$	0.28	0.71	0.73	0.71
Average	–	0.30	0.71	0.71	0.71

Table 4 Iterative updation of the hyper-parametric values with level-0 optimization enhanced in level-*k* optimization using FPA

Initial population with *n* LSTM models in level-0 optimization of hyper-parameters							Population with *n* LSTM models in level-1 Optimization of hyper-parameters						
TL	HL	HN	BS	Epochs	Error rate	SP	TL	HL	HN	BS	Epochs	Error rate	SP
5	2	42	4	303	0.29	0.758	6	2	41	8	277	0.273	0.807
25	8	70	21	244	0.40	0.549	17	5	53	17	247	0.394	0.558
26	4	22	16	64	0.39	0.564	17	3	31	14	157	0.381	0.589
29	7	65	20	440	0.42	0.523	19	5	53	16	345	0.394	0.558
25	1	99	15	186	0.37	0.594	17	1	70	14	218	0.366	0.605
24	4	85	4	212	0.30	0.733	16	3	63	8	231	0.285	0.778
8	**2**	**40**	**12**	**250**	**0.22**	**1.0**	**8**	**2**	**35**	**13**	**256**	**0.218**	**1.0**
1	7	97	7	266	0.45	0.488	4	5	69	9	258	0.441	0.495
24	1	77	8	441	0.36	0.611	16	1	59	10	346	0.345	0.641
15	8	80	8	226	0.47	0.468	12	5	60	10	238	0.436	0.506
8	4	68	10	61	0.38	0.578	8	3	54	11	155	0.371	0.589
14	2	33	21	420	0.26	0.846	13	2	23	21	339	0.258	0.872
28	5	25	24	220	0.42	0.523	18	4	32	18	235	0.374	0.589
29	7	74	27	393	0.48	0.458	19	5	57	20	322	0.452	0.484
7	5	46	19	305	0.42	0.523	7	4	43	16	278	0.412	0.531
28	7	85	28	67	0.51	0.431	18	5	63	20	158	0.425	0.519
16	2	61	14	390	0.27	0.814	15	3	62	13	356	0.267	0.838
6	6	80	29	409	0.32	0.687	7	4	60	21	330	0.315	0.703
25	6	21	15	389	0.41	0.536	17	4	30	14	320	0.382	0.573
17	2	30	17	480	0.31	0.709	13	2	35	15	365	0.243	0.908
13	6	72	24	65	0.43	0.511	11	4	56	18	157	0.384	0.573
16	1	49	23	484	0.36	0.611	12	1	45	18	367	0.354	0.622
20	7	92	6	136	0.41	0.536	14	5	66	9	193	0.391	0.558
17	4	11	25	152	0.32	0.687	13	3	25	19	201	0.315	0.703
18	4	61	17	226	0.28	0.785	13	3	21	15	238	0.294	0.751
						–	–						
						–	–						
Population with *n* LSTM models in level-($k-1$) optimization of hyper-parameters							Population with *n* LSTM models in level-*k* optimization of hyper-parameters						
TL	HL	HN	BS	Epochs	Error rate	SP	TL	HL	HN	BS	Epochs	Error rate	SP
9	2	25	9	180	0.154	0.915	10	2	24	9	186	0.149	0.946
8	3	30	15	210	0.174	0.824	9	3	29	15	204	0.167	0.844
13	2	15	11	162	0.163	0.865	12	2	16	11	168	0.159	0.886
13	1	29	14	151	0.175	0.805	12	1	28	14	157	0.175	0.805
8	2	15	13	193	0.158	0.892	9	2	16	13	187	0.157	0.898
9	2	15	12	198	0.154	0.915	10	2	16	12	192	0.153	0.921
11	2	21	10	176	0.143	0.986	**10**	**2**	**20**	**10**	**182**	**0.141**	**1.0**
13	2	29	13	164	0.161	0.875	12	2	20	13	170	0.156	0.903
11	3	18	11	151	0.173	0.815	10	3	19	11	157	0.174	0.810

Table 4 (continued)

Population with *n* LSTM models in level-($k-1$) optimization of hyper-parameters							Population with *n* LSTM models in level-*k* optimization of hyper-parameters						
TL	HL	HN	BS	Epochs	Error rate	SP	TL	HL	HN	BS	Epochs	Error rate	SP
13	2	18	9	196	0.159	0.886	12	2	19	9	190	0.160	0.881
10	2	28	11	169	0.144	0.979	11	2	27	11	175	0.144	0.979
14	3	30	10	187	0.174	0.810	13	3	29	10	181	0.174	0.810
14	2	28	9	197	0.156	0.903	13	2	27	9	191	0.151	0.933
12	2	26	10	173	0.153	0.921	11	2	25	10	179	0.147	0.959
13	2	18	10	178	0.151	0.933	12	2	19	10	184	0.150	0.940
14	2	26	12	183	0.146	0.965	13	2	25	12	189	0.145	0.972
12	2	16	8	196	0.150	0.940	11	2	17	8	190	0.148	0.952
14	3	27	9	195	0.175	0.805	13	3	26	9	189	0.173	0.815
13	3	15	13	195	0.173	0.815	12	3	16	13	189	0.174	0.810
14	2	22	14	160	0.153	0.921	13	2	21	14	166	0.150	0.940
10	**2**	**21**	**10**	**182**	**0.141**	**1.0**	11	2	22	10	188	0.143	0.986
11	1	22	12	182	0.175	0.805	10	1	21	12	188	0.175	0.805
13	2	20	9	167	0.165	0.854	12	2	21	9	173	0.161	0.875
9	2	20	9	200	0.147	0.959	10	2	21	9	194	0.144	0.979
14	3	19	11	177	0.174	0.810	13	3	20	11	183	0.173	0.815

Bold indicates to determine the best hyperparameters combination in particular level of iteration

TL time lag, *HL* no. of hidden layers, *HN* no. of hidden neurons, *BS* batch size and SP denotes switch probability

To enhance the success of these optimistic feature set, we have incorporated the RNN–LSTM network to develop the precise model which has the key advantage of sequence learning time series prediction with time lag factors that helps to boost the performances. The open challenge with the RNN–LSTM network is to explore the hyper-parameters which plays a vital role to improve the forecasting ability. Hence the new mechanism for the RNN–LSTM network with an optimization technique has been introduced in our proposed framework to search for an optimal solution, also the effective results of applying optimization concepts illustrate great success as shown in the below section.

RNN–LSTM Model Using FPA

The results with an iterative optimization processing of the hyper-parametric values updation at each level based on objective function have been shown in Table 4.

Here, the summary replicates the experimental exploration of mentioned hyper-parameter values along with their error-rate and switch probability value (SP) at level-0, level-1, level $k-1$, and level-k over LON: SXX stock data. Initially, the population is the size (n = 25), so at each level of iteration, the 25-LSTM model is developed as shown in Table 4.

The level-0 results show the threat of having an inappropriate hyper-parametric value to the RNN–LSTM network, the intended model does a lot of compromises which in turn fails to attain an optimum solution (minimum error rate which implies enhanced accuracy). This compromises led the foundation of an increased error rate that has been reflected because of imprecise time lag which affects the sequence learning, unnecessarily required a large number of epochs to train the model, more than the sufficient number of hidden neurons in hidden layers allows the model to overfit.

At level-0, the LSTM model with minimum error rate = 0.22 has been recognized as the best solution that is used to compute the switch probability (SP). This SP value of a particular LSTM model is responsible to find the new maxima or minima values of their hyper-parametric value. The further level-1 population is obtained after performing level-0 optimization such that the old hyper-parametric values are updated with new maxima or minima value. Similarly, the iterative process continues until the successive level attaining minimum error rate must be equal. As shown in level-($k-1$) and level k, the minimum error rate achieved in both iteration remains unchanged so the iteration will stop by providing an optimal solution. Hence, the optimal solution with an optimized hyper-parametric value are TL = 10, HL = 2, HN = 24, BS = 9, Epochs = 186 and Error rate = 0.141.

In addition, the level-k optimization results show the impact of the optimized RNN–LSTM network, the intended model attains an optimum solution by reducing an error rate by an average 0.07 (approx. by 35%), avoids unnecessary training of the model with the sufficient number of epochs, hidden neurons reduced as per network capacity to learn more precisely with an appropriate time lag which allows the model to avoid overfitting.

The comparative results of the level-0 optimization of hyper-parameters with level-k optimization of hyper-parameters for the RNN–LSTM network are shown in Table 5. Here, the summary reveals the experimental results over six stock companies by selecting the best network among the n-LSTM model at each level-0 and

Table 5 Summary of the hyper-parametric values based on fitness value with level-0 optimization enhanced in level-k optimization for optimized RNN–LSTM network using FPA

Company tickers	Minimum fitness value obtained among n-LSTM models in level-0 optimization of hyper-parameters						Minimum fitness value obtained among n-LSTM models in level-k optimization of hyper-parameters					
	TL	HL	HN	BS	Epochs	Error rate	TL	HL	HN	BS	Epochs	Error rate
LON: SXX	8	2	40	12	250	0.22	10	2	20	10	180	0.14
NSE: ICICIBANK	12	2	25	15	245	0.21	11	2	20	10	170	0.13
NYSE: IBM	20	3	50	15	240	0.20	14	2	25	12	175	0.14
BOM: 532725	15	2	25	10	230	0.22	9	2	25	10	165	0.14
NASDAQ: AMZN	14	2	30	12	240	0.19	11	2	20	12	160	0.13
TSE: ABX	14	3	50	12	235	0.20	12	2	25	11	155	0.15

TL time lag, *HL* no. of hidden layers, *HN* no. of hidden neurons and BS denote batch size

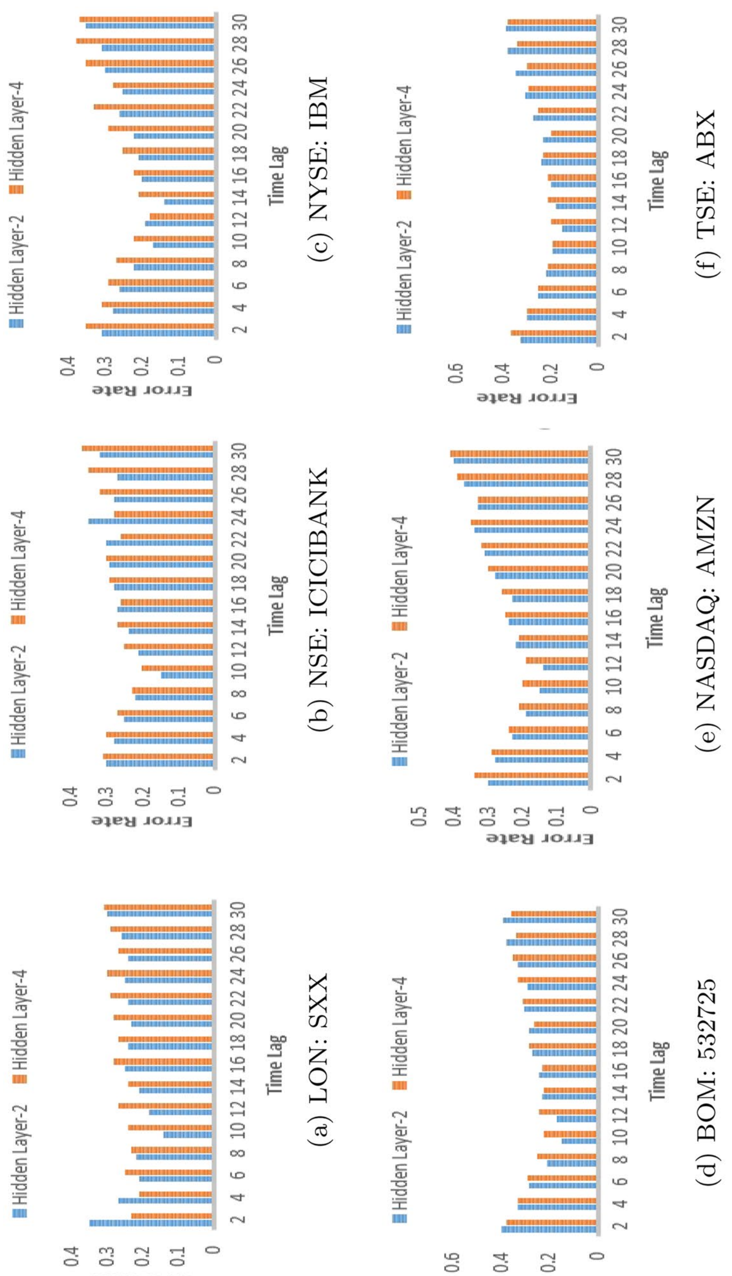


Fig. 6 Plot of adverse effect perceived in error rate during level-k optimization with inappropriate time lag and hidden layers using FPA

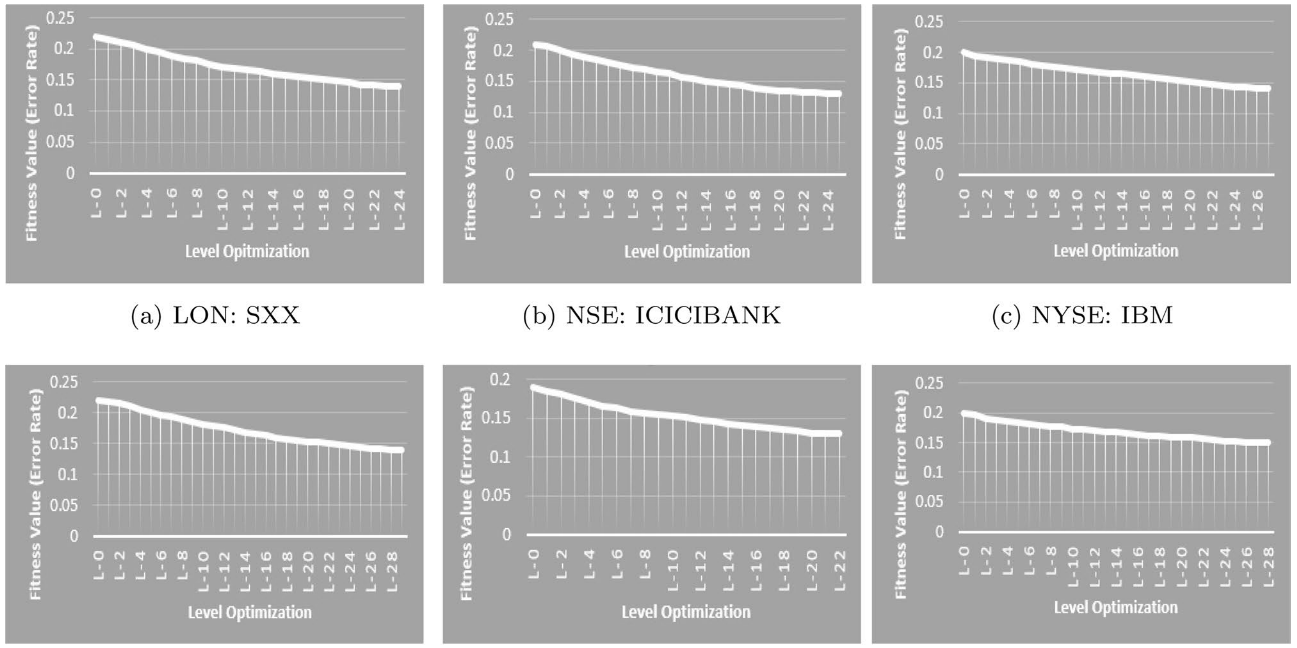
level-*k* optimization based on attaining minimum fitness value (error rate) along with their hyper-parametric values. The results clearly reflect the effectiveness of the assembling optimization system for the searching mechanism of an optimized RNN–LSTM network.

Furthermore, the intra-day stock market forecasting model significantly requires more precise data by opting for proper time lag that discards the outliers data, contains sufficient information along with the hidden layer processing capacity to carry out sequential learning. The conclusive claim made through the point of inappropriate time lag and hidden layers has been seen in the plot of adverse effect perceived in error rate during the accomplishment of level-*k* optimization as shown in Fig. 6. The graph plotted shows the relationship between time lag versus hidden layers versus error rate with the time lag values range between 2 and 30 and the number of hidden layers is 2 and 4. The results clearly convey that opting marginal less or more values of the time lag shows a big impact on the error rate for both the cases of hidden layers. Also, it is investigated that for almost all the stock data, network with much more processing capacity (say hidden layer-4) and the limited amount of significant information required by the forecasting model reached the worst error rate. With the suitable time lag obtained in most of the dataset as seen in the graph that lies between the range of 10–14. It also fails to reach an optimum solution (error rate) with simply increasing the hidden layers from say 2–4. Thus, we can conclude that both these hyperparameters are very sensitive and set plenty of impacts to enhance the forecasting ability.

The graph plotted in Fig. 7 shows the result of an enhanced optimization curve after each level of optimization to generate an optimized RNN–LSTM network. It also explores the convergence of fitness value (error rate) versus the number of levels of optimization to reach an optimal solution for all six examples. It is perceived from Fig. 7 that this mechanism led on the way to a slower convergence rate with an average number of 25 levels of optimization but yields more improved fitness value by reducing the error rate with an average of 0.07 (approx. by 35%).

The graph plotted in Fig. 8 shows the learning curve of the suggested optimized RNN–LSTM network with the tuned hyper-parametric value obtained after level-*k* optimization. Here, the figure plots a two-line curve with the categorical-crossentropy loss over epochs for the train and test dataset. From the plot, it is observed that the training process converged well with a sufficient number of epochs, validation loss is very smooth. The testing process discloses that the model avoids both the cases of either overfitting or underfitting which means that precise learning has been conducted instead of memorizing the dataset.

The results of the summary of the 18 listed companies with optimized hyper-parametric values of the RNN–LSTM network along with performance measures after the execution of level-*k* optimization are shown in Table 6 which strongly present the effectiveness of our proposed framework. Several key advantages of an optimized RNN–LSTM network has been observed based on summary Table 6, it led to the foundation of remarkable forecasting model by finding the optimum global minima with an average enhanced error rate = 0.14 (say approx. error rate minimized from 0.21 to 0.14 by 35%), precision = 0.87, recall = 0.86 and F1-score = 0.86 (say approx. 6% increment). Tuning these sensitive hyper-parameters allow us to figure



(a) LON: SXX (b) NSE: ICICIBANK (c) NYSE: IBM

(d) BOM: 532725 (e) NASDAQ: AMZN (f) TSE: ABX

Fig. 7 Plot of adverse effect perceived in error rate during level-*k* optimization with inappropriate time lag and hidden layers using FPA

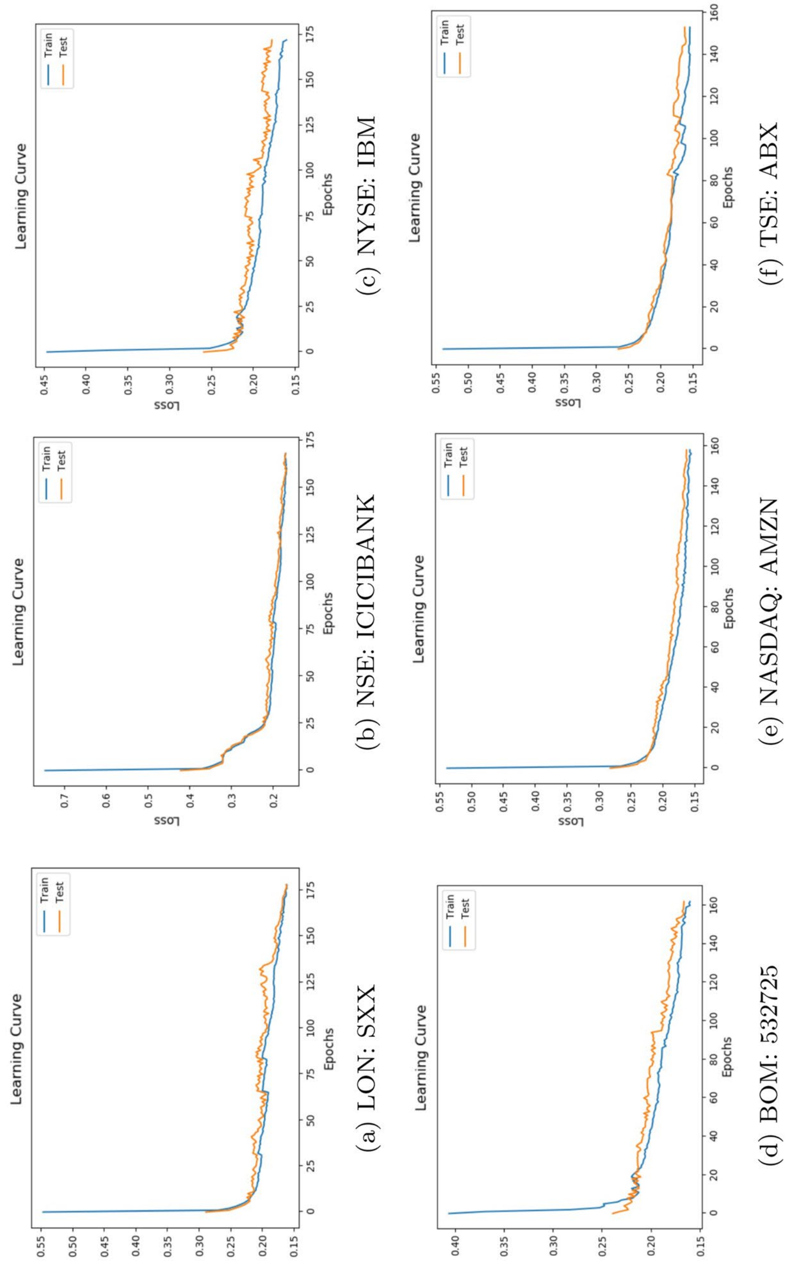


Fig. 8 Plot of optimized RNN–LSTM network learning curve with tuned hyper-parametric value obtained after level-*k* optimization using FPA

Table 6 Summary of the 18 listed companies with optimized hyper-parametric values of RNN–LSTM network along with performance measure after execution of level-*k* optimization using FPA

Company tickers	TL	HL	HN	BS	Epochs	Error rate	Precision	Recall	F1-score
LON: SXX	10	2	20	10	180	0.14	0.89	0.87	0.87
LON: UKOG	10	2	20	10	170	0.13	0.88	0.88	0.87
LON: ADM	13	2	25	12	165	0.14	0.87	0.86	0.86
NSE: HDFCBANK	12	1	30	11	185	0.13	0.87	0.87	0.87
NSE: ICICIBANK	11	2	20	10	170	0.13	0.86	0.87	0.87
NSE: INFY	14	2	25	12	180	0.14	0.88	0.86	0.86
NYSE: IBM	14	2	25	12	175	0.14	0.86	0.85	0.85
NYSE: HMC	11	2	20	10	180	0.15	0.88	0.85	0.85
NYSE: TWTR	10	1	20	11	170	0.14	0.86	0.86	0.86
BOM: 532725	9	2	25	10	165	0.14	0.85	0.88	0.87
BOM: 890144	10	2	20	13	170	0.13	0.87	0.89	0.88
BOM: 532461	12	1	25	12	180	0.14	0.88	0.86	0.85
NASDAQ: AAPL	12	1	30	12	175	0.15	0.86	0.85	0.85
NASDAQ: AMMA	13	2	25	10	165	0.13	0.86	0.87	0.86
NASDAQ: AMZN	11	2	20	12	160	0.13	0.88	0.87	0.87
TSE: ABX	12	2	20	11	170	0.15	0.85	0.84	0.84
TSE: KRN	11	1	30	10	180	0.14	0.86	0.86	0.86
TSE: PSK	10	2	25	10	185	0.13	0.88	0.87	0.87
Average	–	–	–	–	–	0.14	0.87	0.86	0.86

TL time lag, *HL* no. of hidden layers, *HN* no. of hidden neurons and BS denote batch size

out the optimal time lag and the number of hidden layers with the appropriate number is hidden neurons to the network since Fig. 6 clearly shows the adverse effect of marginal error in time lag or hidden layers. In addition, it helps us to minimize the training time by reducing the number of epochs by approx. 30–40%, convincingly developing a more precise forecasting model.

RNN–LSTM Model Using PSO

With, the similar initial population of LON: SXX stock data, the iterative optimization processing results using PSO has been shown in Table 7. Yet again, it resolves the challenge of tuning hyper-parametric values based on the particle velocity computation to find new particle positions. But it is very difficult to control the particle velocity exploitation during searching for maxima or minima. As, after completion of level-0 optimization, the error rate obtained in level -1 optimization with the newly updated population is reduced but the solution with hyper-parametric value fails to escape locally optimal points rather than searching for globally optimal points. Subsequently, the level-*k* iteration provides a faster convergence rather than providing a promising solution (minimum error rate). The optimized hyper-parametric value obtained are TL = 12, HL = 2, HN = 25,

Table 7 Iterative updation of the hyper-parametric values with level-0 optimization enhanced in level-k optimization using PSO

Initial population with n LSTM models in level-0 optimization of hyper-parameters						Population with n LSTM models in level-1 optimization of hyper-parameters					
TL	HL	HN	BS	Epochs	Error rate	TL	HL	HN	BS	Epochs	Error rate
5	2	42	4	303	0.29	5	2	42	5	293	0.281
25	8	70	21	244	0.40	22	3	64	20	245	0.397
26	4	22	16	64	0.39	23	2	25	16	101	0.412
29	7	65	20	440	0.42	25	1	60	19	402	0.397
25	1	99	15	186	0.37	22	2	88	15	198	0.374
24	4	85	4	212	0.30	21	2	76	5	219	0.297
8	**2**	**40**	**12**	**250**	**0.22**	8	2	40	12	250	0.22
1	7	97	7	266	0.45	2	2	86	8	263	0.445
24	1	77	8	441	0.36	21	3	70	8	403	0.355
15	8	80	8	226	0.47	14	2	72	8	230	0.444
8	4	68	10	61	0.38	8	2	63	10	98	0.379
14	2	33	21	420	0.26	**13**	**3**	**34**	**20**	**386**	**0.218**
28	5	25	24	220	0.42	24	2	28	22	226	0.389
29	7	74	27	393	0.48	25	2	68	24	365	0.431
7	5	46	19	305	0.42	7	2	45	18	294	0.421
28	7	85	28	67	0.51	24	2	76	25	103	0.492
16	2	61	14	390	0.27	15	2	57	14	362	0.228
6	6	80	29	409	0.32	6	3	72	26	378	0.34
25	6	21	15	389	0.41	22	3	24	15	362	0.391
17	2	30	17	480	0.31	16	2	32	16	434	0.291
13	6	72	24	65	0.43	12	2	66	22	102	0.371
16	1	49	23	484	0.36	15	1	48	21	438	0.356
20	7	92	6	136	0.41	18	2	82	7	158	0.381
17	4	11	25	152	0.32	16	2	16	23	171	0.351
18	4	61	17	226	0.28	16	3	57	16	230	0.305
Population with n LSTM models in level-$(k-1)$ optimization of hyper-parameters						Population with n LSTM models in level-k optimization of hyper-parameters					
TL	HL	HN	BS	Epochs	Error rate	TL	HL	HN	BS	Epochs	Error rate
10	2	17	12	153	0.165	10	2	21	12	156	0.161
15	2	30	14	175	0.169	14	2	32	14	174	0.164
11	1	30	8	162	0.174	11	1	32	8	163	0.173
14	3	15	15	158	0.176	13	3	20	15	160	0.176
12	**2**	**25**	**12**	**155**	**0.16**	**12**	**2**	**25**	**12**	**155**	**0.160**
14	3	21	14	173	0.172	13	3	24	16	154	0.171
8	2	35	160	151	0.169	8	2	36	16	154	0.168
8	3	22	15	165	0.177	8	3	25	15	166	0.176
11	2	22	11	174	0.165	11	2	25	11	174	0.164
15	2	15	14	167	0.171	14	2	20	14	167	0.171
9	1	35	12	153	0.174	9	1	36	12	156	0.173

Table 7 (continued)

Population with *n* LSTM models in level-($k-1$) optimization of hyper-parameters						Population with *n* LSTM models in level-*k* optimization of hyper-parameters					
TL	HL	HN	BS	Epochs	Error rate	TL	HL	HN	BS	Epochs	Error rate
13	3	31	14	166	0.173	12	3	32	14	166	0.171
11	2	24	15	169	0.164	11	2	27	15	169	0.162
14	3	16	15	157	0.174	13	3	20	15	159	0.173
14	2	20	13	165	0.166	13	2	24	13	166	0.162
10	3	27	13	178	0.172	10	3	29	13	177	0.171
8	2	16	12	172	0.162	8	2	20	12	172	0.161
14	3	20	10	166	0.172	13	3	24	10	166	0.171
11	1	21	16	167	0.173	11	1	24	16	167	0.172
11	2	31	13	161	0.161	11	2	32	13	162	0.162
13	2	23	16	169	0.163	12	2	26	16	169	0.161
8	2	31	10	166	0.162	8	2	32	10	166	0.162
11	2	16	15	161	0.165	11	2	20	15	162	0.164
11	3	18	13	150	0.171	11	3	22	13	154	0.170
13	1	29	16	153	0.172	12	1	31	16	156	0.172

Bold indicates to determine the best hyperparameters combination in particular level of iteration

Table 8 Summary of the hyper-parametric values based on fitness value with level-0 optimization enhanced in level-*k* optimization for optimized RNN–LSTM network using PSO

Company tickers	Minimum fitness value obtained among *n*-LSTM models in level-0 optimization of hyper-parameters						Minimum fitness value obtained among *n*-LSTM models in level-*k* optimization of hyper-parameters					
	TL	HL	HN	BS	Epochs	Error rate	TL	HL	HN	BS	Epochs	Error rate
LON: SXX	8	2	40	12	250	0.22	12	2	25	12	155	0.16
NSE: ICICIBANK	12	2	25	15	245	0.21	14	2	20	11	150	0.15
NYSE: IBM	20	3	50	15	240	0.20	11	1	25	10	160	0.14
BOM: 532725	15	2	25	10	230	0.22	11	2	25	13	150	0.16
NASDAQ: AMZN	14	2	30	12	240	0.19	13	2	30	10	140	0.15
TSE: ABX	14	3	50	12	235	0.20	14	1	27	14	145	0.17

TL time lag, *HL* no. of hidden layers, *HN* no. of hidden neurons and BS denote batch size

BS = 12, Epochs = 155 and error rate = 0.16. In addition, the supremacy of level-*k* optimization over level-0 has been shown in Table 8 that clearly reveals the facts of the model such as, it reaches the most favorable solution by reducing an error rate by an average 0.05 (approx. 25%).

Furthermore, the results as shown in Fig. 9, convey the similar consequences by opting marginal less or more values of the time lag which shows a gigantic increase in the error rate for both the cases of hidden layers. Yet again, the graph plotted

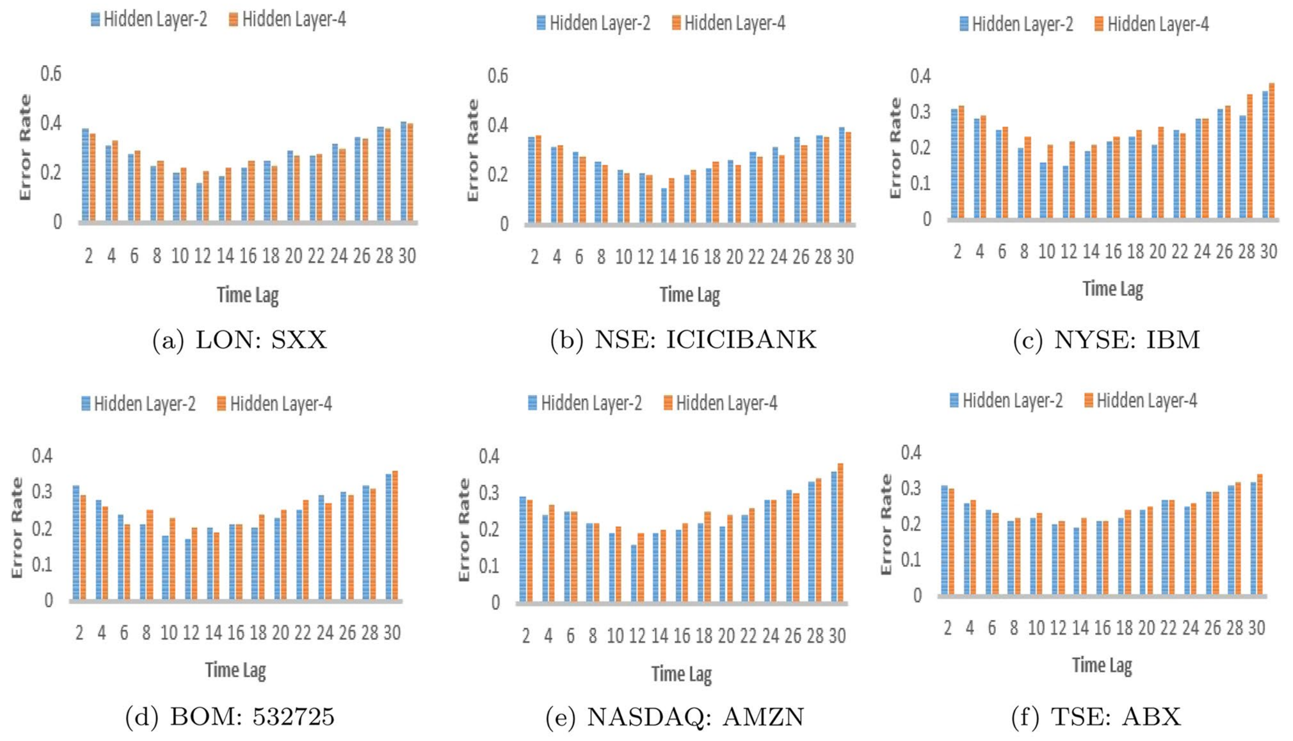
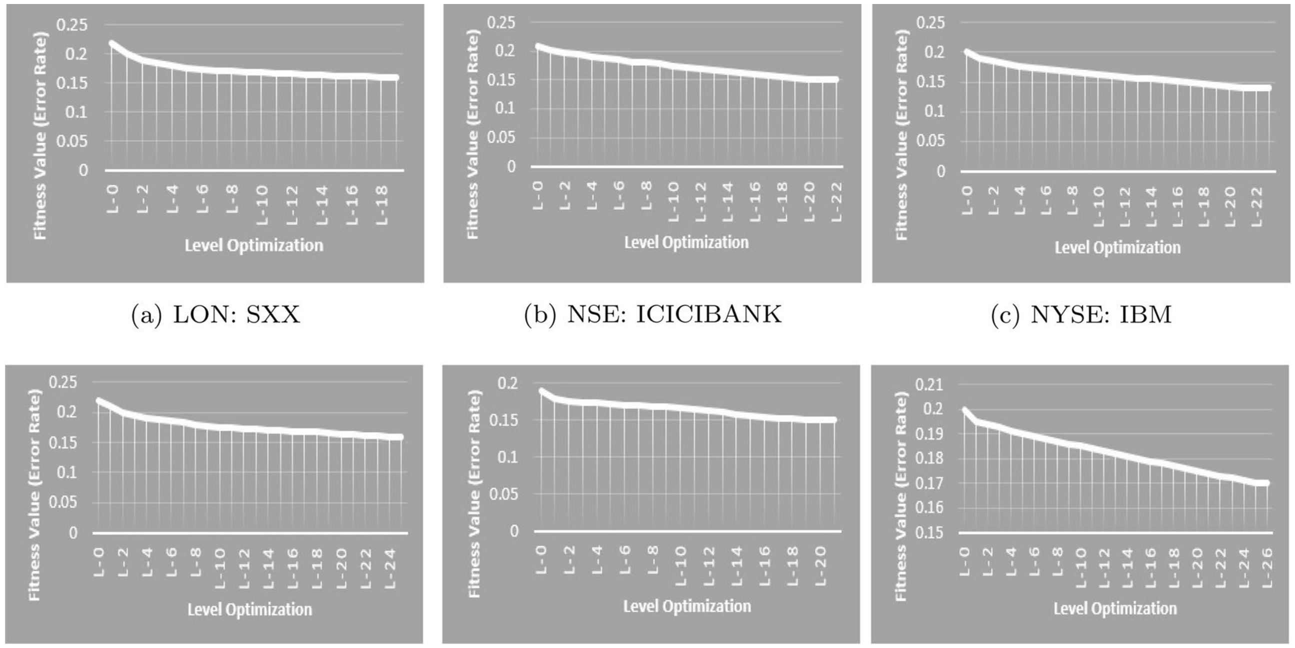


Fig. 9 Plot of adverse effect perceived in error rate during level-k optimization with an inappropriate time lag and hidden layers using PSO



(a) LON: SXX (b) NSE: ICICIBANK (c) NYSE: IBM

(d) BOM: 532725 (e) NASDAQ: AMZN (f) TSE: ABX

Fig. 10 Plot of adverse effect perceived in error rate during level-*k* optimization with inappropriate time lag and hidden layers using PSO

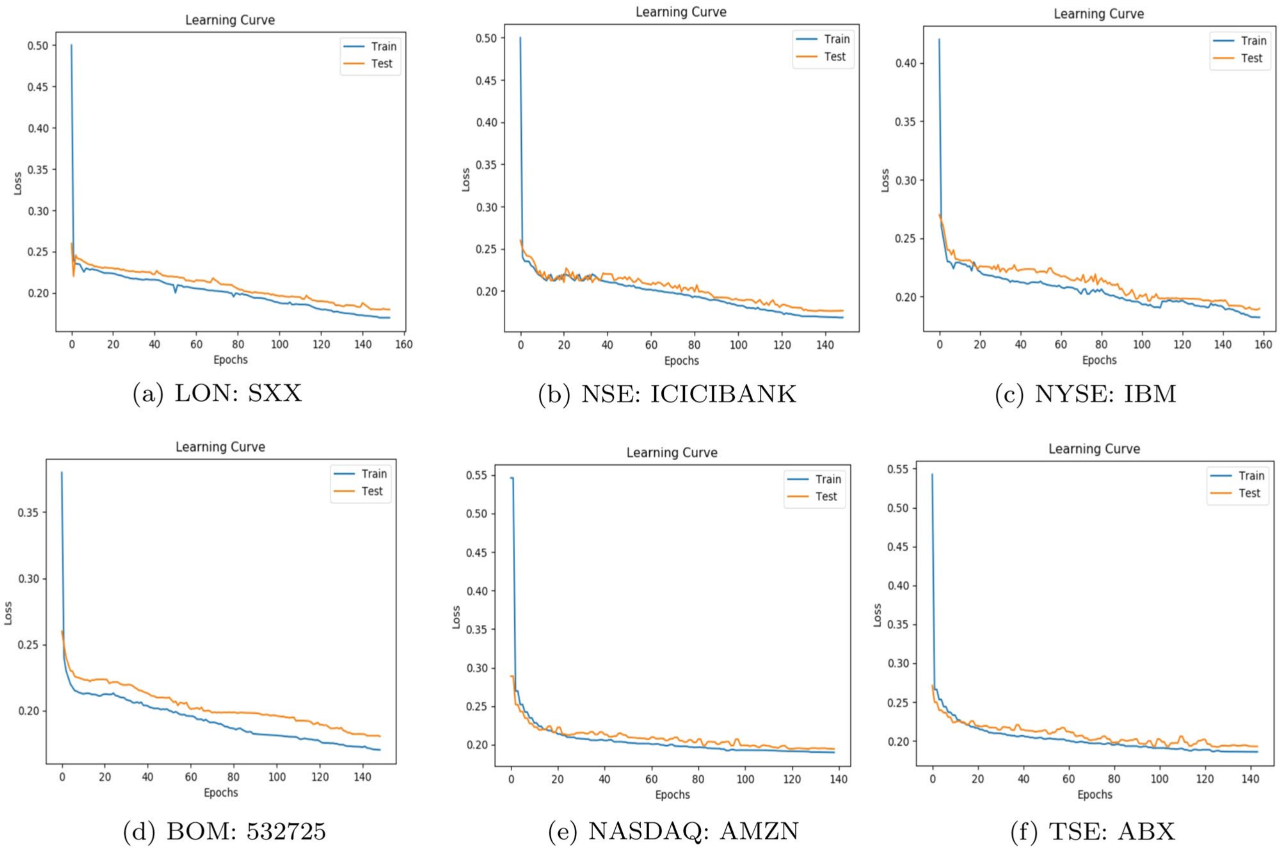


Fig. 11 Plot of optimized RNN–LSTM network learning curve with tuned hyper-parametric value obtained after level-*k* optimization using PSO

Table 9 Summary of the 18 listed companies with optimized hyper-parametric values of RNN–LSTM network along with performance measure after execution of level-*k* optimization using PSO

Company tickers	TL	HL	HN	BS	Epochs	Error rate	Precision	Recall	F1-score
LON: SXX	12	3	25	12	155	0.16	0.86	0.84	0.85
LON: UKOG	14	2	30	12	145	0.14	0.87	0.86	0.86
LON: ADM	14	2	35	10	150	0.14	0.84	0.86	0.85
NSE: HDFCBANK	13	2	30	11	155	0.15	0.84	0.84	0.84
NSE: ICICIBANK	14	2	20	11	150	0.15	0.86	0.85	0.85
NSE: INFY	13	2	30	10	155	0.14	0.85	0.86	0.85
NYSE: IBM	11	1	25	10	160	0.14	0.86	0.85	0.86
NYSE: HMC	13	2	25	12	140	0.15	0.85	0.85	0.85
NYSE: TWTR	10	2	20	10	145	0.16	0.85	0.83	0.84
BOM: 532725	11	2	25	13	150	0.16	0.85	0.82	0.83
BOM: 890144	12	2	20	11	155	0.15	0.86	0.84	0.85
BOM: 532461	12	1	25	12	145	0.14	0.85	0.86	0.85
NASDAQ: AAPL	11	2	30	10	160	0.15	0.84	0.85	0.84
NASDAQ: AMMA	11	2	35	12	150	0.14	0.86	0.86	0.86
NASDAQ: AMZN	13	2	30	10	140	0.15	0.84	0.87	0.85
TSE: ABX	14	1	25	14	145	0.17	0.83	0.84	0.84
TSE: KRN	12	2	30	13	155	0.16	0.86	0.82	0.84
TSE: PSK	10	2	35	11	140	0.15	0.85	0.85	0.85
Average	–	–	–	–	–	0.15	0.85	0.85	0.85

TL time lag, *HL* no. of hidden layers, *HN* no. of hidden neurons and BS denote batch size

clearly suggest that increasing the number of the hidden layer with more than sufficient processing capacity increases the error rate.

From the graph plotted in Fig. 10, it is perceived that this mechanism led on the way to a bit higher convergence rate with an average number of 22 levels of optimization as compared with the FPA algorithm (25 levels of optimization). But this mechanism fails to find an optimum solution by minimizing the error rate with an average of 0.05 (approx. 25%) which is around 10% less obtained by the FPA optimization approach.

From the graph plotted in Fig. 11, it is observed that the training process converged well with a very less number of epochs required to train and validation loss is also very smooth as compared with the FPA algorithm. The testing process discloses similar facts such as the model avoids both the cases of either overfitting or underfitting.

The success of PSO algorithm based on the summary of 18 listed companies as shown in Table 9 has also been remarkable by minimizing the average error rate = 0.15 (say approx. error rate minimized from 0.20 to 0.15 by 25%), precision = 0.85, recall = 0.85 and F1-score = 0.85 (say approx. 4% increment). In addition, it helps us to minimize the training time by reducing the number of epochs by approx. 35–45%, convincingly developing a more precise forecasting model.

Another comparative report analysis on time and resource consumption using parallel computing have been explored. The investigation of our proposed search methods (FPA, PSO) with existing search methods named as Genetic algorithm (GA), Local Beam search (LBS) with several initial population sizes ($n = 25, 30, 35, 40$) has been conducted. In addition, since RNN–LSTM deep learning is notorious for time and resource consumption due to a large number of parameters are learned in the training phase. Therefore, in our stated approach initially, we have worked on the optimal feature selection phase to reduce the input feature dimensionality challenge. It helps to minimize the time complexity and resource consumption of the RNN–LSTM network during the training phase by more than halve and attained the minimized error rate.

As, the investigated results clearly suggest that population size ($n = 25$) is sufficient to search for an optimistic model using FPA, PSO, GA, and LBS search methods. Also, with increased population size (n) we have achieved a similar error rate. But, it adversely affects the time complexity and resource utilization by showing an increment of approx. 30–35%, 60–65% and almost twice for respective population size ($n = 30, 35, 40$, etc.). Furthermore, with an ideal value of $n = 25$, FPA search method attain minimized error rate = 0.14, time consumption = 251.25 min, resource consumption = 2332 MB. Similarly, PSO obtained values are 0.15, 221.1 min, and 1805 MB; GA obtained values are 0.17, 451.75 min, and 3945 MB; LBS obtained values are 0.19, 259.64 min, and 3113 MB. (These above data of time and resource consumption has been computed using python library package: timeit; resource.RUSAGE_SELF).

The comparative results of our proposed model over a similar sphere forecasting model are shown in Table 10. Trending deep learning, the LSTM network mostly preferred these days and considered a benchmark model, since it shows great success with the advantages of sequential time series prediction. In addition, SVM, LRC in past decades achieved success, claimed as the finest machine learning algorithm [8]. The benchmark dataset or the dataset which has been used by several comparative papers is not publically available so we have evaluated the benchmark model will the similar dataset which has been used in our study.

Here, Zhou et al. [24] predicted the price for CSI-300 based on the two different strategies generative adversarial network (GAN-FD) and long short-term memory (LSTM-FD) by achieving an avg acc of 66% and 64%, respectively. Arevalo et al. [25] adopted a deep neural network (DNN) for the forecasting of next time-series price movement by attaining an avg acc = 65%. Kumar and Haider 2019 [21] in his research work use the approach based on the different benchmark models without optimization process such as LSTM without optimal features selection (OFS) and reaches the avg error rate = 0.29 and avg acc = 70%, Logistic regression classifier (LRC) with OFS reaches avg error rate = 0.30 and avg acc = 69%, LRC-LSTM with OFS shows better performance measure with an avg error rate = 0.20 and avg acc = 81%, Decision Tree (DT)-LSTM with OFS reaches an avg error rate = 0.23 and avg acc = 78%, Support Vector Machine (SVM)-LSTM with OFS reaches an avg error rate = 0.22 and avg acc = 79%. Several approaches attain excellent results but still fails to reach an optimum solution. However, the limitation of these above studies as

Table 10 Comparative results with existing work

Authors	Stock exchange	Algorithm	OFS	OPT	Performance measure				Accuracy measure		
					Error rate	Precision	Recall	F1-score	Max	Min	Avg
Zhou et al. 2018 [24]	CSI-300	GAN-FD	No	No	0.34	–	–	–	69	53	66
	Index	LSTM-FD	No	No	0.36	–	–	–	65	53	64
Arevalo et al. 2016 [25]	US Apple	DNN	Yes	No	0.35	–	–	–	66	62	65
	Lon, Nse,	LSTM	No	No	0.29	0.71	0.70	0.70	81	55	70
Kumar and Haider 2019 [21]	Nyse, Bom,	LRC	Yes	No	0.30	0.69	0.70	0.69	72	66	69
	Nasdaq, Tse	LRC-LSTM	Yes	No	0.20	0.81	0.81	0.81	84	79	81
		DT-LSTM	Yes	No	0.23	0.78	0.78	0.77	82	74	78
		SVM-LSTM	Yes	No	0.22	0.78	0.80	0.79	82	77	79
Chung and Shin 2018 [2]	KOSPI	GA-LSTM	No	Yes	0.17	–	–	–	–	–	–
	Lon, Nse,	**LRC with**	**Yes**	**Yes**	**0.14**	**0.87**	**0.86**	**0.86**	**88**	**84**	**86**
Our Proposed work	Nyse, Bom,	**FPA-LSTM**									
	Nasdaq, Tse	LRC with PSO-LSTM	Yes	Yes	0.15	0.85	0.85	0.85	86	83	85

Bold indicates to determine the most powerful methodology with best acheived results

OFS optimal feature selection, *OPT* denote optimization shown in Table 10 which degrades the performance measure are primarily due to the optimization problem of a network architecture of the model.

Recently, Chung and Shin [2] in his study functioned with the similar optimization problem of the LSTM network by tuning two hyper-parameters named as time lag and hidden layers using a Genetic Algorithm. With the GA-LSTM approach, although the model achieved enhanced performance with higher accuracy but still have drawbacks like the convergence rate is very slow, a large number of iterations are required to reach the best topology, search mechanism with few more sensitive hyper-parameter needs to be optimized which can improve the performance measure. Our proposed hybrid mechanism LRC with FPA-LSTM and LRC with PSO-LSTM overcome the above-mentioned drawbacks of GA-LSTM such as the searching mechanism with the five most sensitive hyper-parameters to be optimized which in turns shows much-enhanced performance with higher accuracy, rapid convergence rate with a reduced number of iterations to reach the optimum solution. In addition, among these two proposed approach, LRC with FPA-LSTM with the advantage of switch probability as a key factor attained most enhanced average values of minimized error rate = 0.14, precision = 0.87, recall = 0.86, F1-score = 0.86 and the maximum acc = 88%, minimum acc = 84%, avg acc = 86% has been achieved by any of the listed companies. In addition, LRC with PSO-LSTM achieved excellent results with avg minimized error rate = 0.15, precision = 0.85, recall = 0.85, F1-score = 0.85 and max acc = 86%, min acc = 83%, avg acc = 85%. Thus, we can claim the superiority of both the proposed hybrid mechanism which led us to the remarkable optimization system with much more enhanced performance measure. Though PSO shows the rapid convergence rate, less number of level of optimization, and less number of epochs but still FPA yields more improved accuracy.

We have also tested statistically using paired *t* test to claim the statement whether the improvement of our proposed model outperforms the existing (benchmark) model significantly. A paired *t* test statistic is used to compare the means and standard error of the mean (SEM) of two paired set samples, computed as follows. Here, Zhou et al. [24] predicted the price for CSI-300 based on the two different strategies generative adversarial network (GAN-FD) and long short-term memory (LSTM-FD) by achieving an avg acc of 66% and 64%, respectively. Arevalo et al. [25] adopted a deep neural network (DNN) for the forecasting of the next time-series price movement by attaining an avg acc = 65%. Kumar and Haider [21] in his research work use the approach based on the different benchmark models without optimization process such as LSTM without optimal features selection (OFS) and reaches the avg error rate = 0.29 and avg acc = 70%, Logistic regression classifier (LRC) with OFS reaches avg error rate = 0.30 and avg acc = 69%, LRC-LSTM with OFS shows better performance measure with an avg error rate = 0.20 and avg acc = 81%, Decision Tree (DT)-LSTM with OFS reaches an avg error rate = 0.23 and avg acc = 78%, Support Vector Machine (SVM)-LSTM with OFS reaches an avg error rate = 0.22 and avg acc = 79%. Several approaches attain excellent results but still fails to reach an optimum solution. However, the limitation of these above studies as shown in Table 10 which degrades the performance measure are primarily due to the optimization problem of a network architecture of the model.

Recently, Chung and Shin [2] in his study functioned with the similar optimization problem of the LSTM network by tuning two hyper-parameters named as time lag and hidden layers using a Genetic Algorithm. With the GA-LSTM approach, although the model achieved enhanced performance with higher accuracy but still have drawbacks like the convergence rate is very slow, a large number of iterations are required to reach the best topology, search mechanism with few more sensitive hyper-parameter needs to be optimized which can improve the performance measure. Our proposed hybrid mechanism LRC with FPA-LSTM and LRC with PSO-LSTM overcome the above-mentioned drawbacks of GA-LSTM such as the searching mechanism with the five most sensitive hyper-parameters to be optimized which in turns shows much-enhanced performance with higher accuracy, rapid convergence rate with the reduced number of iterations to reach the optimum solution. In addition, among these two proposed approach, LRC with FPA-LSTM with the advantage of switch probability as a key factor attained most enhanced average values of minimized error rate = 0.14, precision = 0.87, recall = 0.86, F1-score = 0.86 and the maximum acc = 88%, minimum acc = 84%, avg acc = 86% has been achieved by any of the listed companies. In addition, LRC with PSO-LSTM achieved excellent results with avg minimized error rate = 0.15, precision = 0.85, recall = 0.85, F1-score = 0.85 and max acc = 86%, min acc = 83%, avg acc = 85%. Thus, we can claim the superiority of both the proposed hybrid mechanism which led us to the remarkable optimization system with much more enhanced performance measure. Though PSO shows the rapid convergence rate, less number of level of optimization, and less number of epochs but still FPA yields more improved accuracy.

We have also tested statistically using paired *t* test to claim the statement whether the improvement of our proposed model outperforms the existing (benchmark) model significantly. A paired *t* test statistic is used to compare the means and standard error of the mean (SEM) of two paired set samples, computed as follows.

$$\text{Paired } t\text{test } (t) = \frac{m}{SD/\sqrt{n}} \tag{16}$$

$$\text{Standard Error of the Mean (SEM)} = \frac{SD}{\sqrt{n}}, \tag{17}$$

where m = mean, SD = standard deviation and n is size of sample.

We use the paired *t* test to evaluate the superiority of the comparable model by calculating the level of significance (p value) based on error rate as measuring variable. The mathematical representation to evaluate the null hypothesis and alternate hypothesis are defined below:

$$H_0 : \mu_{(\text{existing model}-\text{proposed model})} = 0 \tag{18}$$

$$H_a : \mu_{(\text{existing model}-\text{proposed model})} > 0 \tag{19}$$

The null hypothesis (H_0) claims that if a significant difference is zero then the proposed model forecast is less accurate than other existing models whereas the

Table 11 Paired t test result for model comparison with null hypothese that the proposed model forecast are less accurate than other methods

Sample data	Comparable models						
	Existing model	Proposed model	Mean	SD	SEM	t	p
Lon, Nse, Nyse, Bom, Nasdaq, Tse	LRC-LSTM	LRC with LSTM-FPA	0.0683	0.0132	0.0054	12.5930	0.000134***
	LRC-LSTM	LRC with LSTM-PSO	0.055	0.0176	0.0071	7.6519	0.000303***
KOSPI	GA-LSTM	LRC with LSTM-FPA	0.0425	0.0117	0.0047	8.877	0.000201***
	GA-LSTM	LRC with LSTM-PSO	0.0342	0.0105	0.0047	7.9806	0.006338***
Lon, Nse, Nyse, Bom, Nasdaq, Tse	GA-LSTM	LRC with LSTM-FPA	0.0357	0.0104	0.0042	8.3564	0.000151***
	GA-LSTM	LRC with LSTM-PSO	0.0233	0.015	0.0061	3.701	0.000249***

***Significant at the level 1%

alternate hypothesis (H_a) claims the proposed model is superior if a significant difference is greater than zero. The paired t test statistics compute mean, SD, SEM, t values, and p values for several pairing comparable models as shown in Table 11. As these evaluated results are statistically significant at the level of 1% [significance level alpha (α) = 0.1]. Also, since all the obtained p value is greater than α value ($p > \alpha$) that means the null hypothesis is rejected and the alternate hypothesis is accepted. This result verifies the statistical improvement of our proposed model that outperforms the existing model, shows superior forecasting capability. In addition, results reflect less risk indication with very low p values, and error values are consistent with very low standard deviation, SEM values.

Conclusion and Future Work

Accurate forecasting of the stock market price to capitalize over the threat of the financial loss or gains during an investment plan significantly need to assess the most volatile data which are very adaptive in nature, prerequisites to adjust dynamically to identify market behavior. Consequently, several mechanisms in the past have been proposed to develop a precise model and to predict financial time series value. In addition, many studies have claimed the LSTM network as a benchmark model, since it shows excellent achievements due to the key advantage of the memory cell that stores data to perform sequential learning. Despite their success, still, the open challenge for the scientist is to resolve the optimal learning architectural factors of this network. In this research, we built and optimized the RNN–LSTM network by exploring new ways of solving optimization problems using nature-inspired meta-heuristics algorithms. We have developed two hybrid mechanisms, LRC with FPA-LSTM and LRC with PSO-LSTM to select the best computational optimized

RNN–LSTM network for intraday stock market prediction. The key to our success, rely upon the model building process by carrying out investigation over two important aspects named as a baseline feature engineering for important predictor variable selection using wrapper method (RFE-LRC) and tuning the most sensitive hyper-parameters (architectural factors) of the RNN–LSTM network.

The experimental results of 18 listed companies using proposed hybrid mechanism LRC with FPA-LSTM and LRC with PSO-LSTM over existing literature has been shown in Table 10 which clearly reflects that it outperforms the existing model. These techniques have a key feature of global searching which has eliminated drawbacks of existing model based on several metrics such as rapid convergence rate, minimized number of iterations to reach an optimum solution, minimized training time by reducing the number of epochs (approx. 30–45%), validation loss is very smooth and most importantly enhanced performance with higher accuracy. The results strongly present the effectiveness of our proposed framework which led to the foundation of a remarkable forecasting model by finding the optimal network that minimized the avg error rate by approx. 35%, and maximize the accuracy precision, recall, and F1-score with an increment of approx. 6%. In addition, it is perceived that although PSO shows the rapid convergence rate, less number of level of optimization and less number of epochs but still FPA dominated over PSO algorithm due to the key feature of switch probability by achieving an avg minimized error rate = 0.14 which is 10% less, precision = 0.87, recall = 0.86 and F1-score = 0.86 which are bit more than PSO.

Further in the future, we will enhance the model by focusing on the various other hyper-parameters in the LSTM network such as optimizer, activation function, etc. In addition, a knowledge-based system for intraday stock market prediction will be developed to provide a better investment plan based on the skilled trader's strategies.

Compliance with Ethical Standards

Conflict of interest The authors of the paper have no conflict of interest with any companies or institutions.

Human and animal rights statement This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., Lew, M.S.: Deep learning for visual understanding: a review. Neurocomputing **187**, 27–48 (2016)
2. Chung, H., Shin, K.-S.: Genetic algorithm-optimized long short-term memory network for stock market prediction. Sustainability **10**(10), 3765 (2018)
3. Baykasoğlu, A., Özbakır, L., Tapkan, P.: Artificial bee colony algorithm and its application to generalized assignment problem. In: Swarm intelligence, focus on ant and particle swarm optimization. IntechOpen 1, (2007)
4. Madasu, S.D., Kumar, M.S., Singh, A.K.: A lower pollination algorithm based automatic generation control of interconnected power system. Ain Shams Eng. J. **9**(4), 1215–1224 (2016)

5. Tay, F.E., Cao, L.: Application of support vector machines in financial time series forecasting. Omega **29**(4), 309–317 (2001)
6. Franses, P.H., Van Dijk, D.: Forecasting stock market volatility using (non-linear) garch models. J. Forecast. **15**(3), 229–235 (1996)
7. Wei, L.-Y., Cheng, C.-H.: A hybrid recurrent neural networks model based on synthesis features to forecast the Taiwan stock market. Int. J. Innov. Comput. Inf. Control **8**(8), 5559–5571 (2012)
8. Atsalakis, G.S., Valavanis, K.P.: Surveying stock market forecasting techniques-part II: soft computing methods. Expert Syst. Appl. **36**(3), 5932–5941 (2009)
9. Kim, K.-J., Han, I.: Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. Expert Syst. Appl. **19**(2), 125–132 (2000)
10. Yu, H., Chen, R., Zhang, G.: A SVM stock selection model within PCA. Procedia Comput. Sci. **31**, 406–412 (2014)
11. Gheyas, I.A., Smith, L.S.: A novel neural network ensemble architecture for time series forecasting. Neurocomputing **74**(18), 3855–3864 (2011)
12. Patel, J., Shah, S., Thakkar, P., Kotecha, K.: Predicting stock market index using fusion of machine learning techniques. Expert Syst. Appl. **42**(4), 2162–2172 (2015)
13. Ballings, M., Van den Poel, D., Hespeels, N., Gryp, R.: Evaluating multiple classifiers for stock price direction prediction. Expert Syst. Appl. **42**(20), 7046–7056 (2015)
14. Lee, J., Jang, D., Park, S.: Deep learning-based corporate performance prediction model considering technical capability. Sustainability **9**(6), 899 (2017)
15. Ding, X., Zhang, Y., Liu, T., Duan, J.: Deep learning for event-driven stock prediction. In: Twenty-fourth international joint conference on artiicial intelligence 2327–2333, (2015)
16. Yoshihara, A., Fujikawa, K., Seki, K., Uehara, K.: Predicting stock market trends by recurrent deep neural networks. In: Paciic rim international conference on artiicial intelligence, pp. 759–769. Springer (2014)
17. Sezer, O.B., Ozbayoglu, M., Dogdu, E.: A deep neural-network based stock trading system based on evolutionary optimized technical analysis parameters. Procedia Comput. Sci. **114**, 473–480 (2017)
18. Fischer, T., Krauss, C.: Deep learning with long short-term memory networks for financial market predictions. Eur. J. Oper. Res. **270**(2), 654–669 (2018)
19. Hsieh, T.-J., Hsiao, H.-F., Yeh, W.-C.: Forecasting stock markets using wavelet transforms and recurrent neural networks: an integrated system based on artificial bee colony algorithm. Appl. Soft Comput. **11**(2), 2510–2525 (2011)
20. Rather, A.M., Agarwal, A., Sastry, V.: Recurrent neural network and a hybrid model for prediction of stock returns. Expert Syst. Appl. **42**(6), 3234–3241 (2015)
21. Kumar, K., Haider, M.T.U.: Blended computation of machine learning with the recurrent neural network for intra-day stock market movement prediction using a multi-level classifier. Int. J. Comput. Appl. 1–17 (2019)
22. Tsai, C.-F., Hsiao, Y.-C.: Combining multiple feature selection methods for stock prediction: union, intersection, and multi-intersection approaches. Decis. Support Syst. **50**(1), 258–269 (2010)
23. Achelis, S.B.: Technical Analysis from A to Z. McGraw Hill, New York (2001)
24. Zhou, X., Pan, Z., Hu, G., Tang, S., Zhao, C.: Stock market prediction on high-frequency data using generative adversarial nets. Math. Probl. Eng. **2018**, 1–11 (2018)
25. Arévalo, A., Ni no, J., Hernández, G., Sandoval, J.: High-frequency trading strategy based on deep neural networks. In: International conference on intelligent computing, pp. 424–436. Springer (2016)