Applied Soft Computing 35 (2015) 670–680

Contents lists available at ScienceDirect

Applied Soft Computing

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

A Naïve SVM-KNN based stock market trend reversal analysis for Indian benchmark indices

Rudra Kalyan Nayak[a,*], Debahuti Mishra[a,1], Amiya Kumar Rath[b,2]

[a] Department of Computer Science and Engineering, Institute of Technical Education and Research, Siksha 'O' Anusandhan University, Khandagiri, Bhubaneswar, OD 751030, India
[b] Department of Computer Science & Engineering, Veer Surendra Sai University of Technology, Burla, Sambalpur, OD 768018, India

ARTICLE INFO

Article history:
Received 7 February 2015
Received in revised form 1 June 2015
Accepted 1 June 2015
Available online 6 July 2015

Keywords:
Support Vector Machine (SVM)
K-Nearest Neighbor (KNN)
Mean Absolute Percentage Error (MAPE)
Root Mean Square Error (RMSE)
Mean Squared Forecast Error (MSFE)
Root Mean Squared Forecast Error (RMSFE)
and Mean Absolute Forecast Error (MAFE)

ABSTRACT

This paper proposes a hybridized framework of Support Vector Machine (SVM) with K-Nearest Neighbor approach for Indian stock market indices prediction. The objective of this paper is to get in-depth knowledge in the stock market in Indian Scenario with the two indices such as, Bombay Stock Exchange (BSE Sensex) and CNX Nifty using technical analysis methods and tools such as predicting closing price, volatility and momentum of the stock market for the available data. This hybrid model uses SVM with different kernel functions to predict profit or loss, and the output of SVM helps to compute best nearest neighbor from the training set to predict future of stock value in the horizon of 1 day, 1 week and 1 month. The proposed SVM and KNN based prediction model is experienced with the above mentioned distinguished stock market indices and the performance of proposed model has been computed using Mean Squared Error and also been compared with recent developed models such as FLIT2NS and CEFLANN respectively. The limitation of both of those existing models undergoes complex weight updating procedures, whereas, proposed SVM-KNN hybridized model scales relatively well to high dimensional data and the trade-off between classifier complexity and error can be controlled explicitly and have better prediction capability.

1. Introduction

In a liberalized economy, stock market is considered as an important indicator of the health of the economy. Interestingly, it seems to be most sensitive to non-economic factors like political havoc, terrorist attacks, and investors' psychology [1]. Even, the strategic planning of a company is known to cause rise or fall in the stock market. Out of three financial time series (1 day, 1 week, and 1 month) under the study, stock market has drawn the attention of researchers most. From literature it has been observed that soft computing methods such as Artificial Neural Network (ANN) [2,3], Fuzzy Set, Support Vector Machine (SVM) [4], and the evolutionary methods like Genetic Algorithm, Bacterial Foraging Optimization and, Particle Swarm Optimization have been used by researcher to predict stock market indices [5]. Bombay Stock Exchange (BSE) was predicted by ANNs [6]; however, due to large dimension of neurons, the computational complexity with such models has been increased. Thus, hybrid models based on ANN, such as Linear Wavelet Neural Network (LWNN), Functional Link Artificial Neural Network (FLANN) [7] have been developed. The same LWNN model in [8] is used to predict stock of NASDAQ and S&P CNX Nifty index. Results show that LWNN model is slightly better than ANN models. Whereas, S&P 500 indices is predicted using FLANN [9,10]. Looking at the limitations of ANN like black box technique [11], researchers prefer the hybrid techniques to develop the efficient forecasting model. A combination of methods such as fuzzy rule based system [12,13], fuzzy neural network [14], Kalman filter [15] with hybrid neuro-fuzzy architecture have been developed to predict financial time series data. A computationally efficient functional link artificial neural network (CEFLANN) [16] is the most recent method used for prediction of stock indices. The performance of CEFLANN increases when it trained using SADHS-OELM. SADHS-OELM is a hybrid learning framework called Self Adaptive Differential Harmony Search Based Optimized Extreme Learning Machine (SADHS-OELM) for single hidden layer feed forward neural network (SLFN). To optimize the fitting performance of SLFNs, extreme learning machine (ELM) along with self adaptive

* Corresponding author. Tel.: +91 9861366884; fax: +91 674 2350181.
E-mail addresses: rudrakalyannayak@gmail.com (R.K. Nayak), mishradebahuti@gmail.com (D. Mishra), amiyaamiya@rediffmail.com (A.K. Rath).
[1] Tel: +91 9337742719; fax: +91 674 2350181.
[2] Tel.: +91 09437577560; fax: +91 0663 2430204.

http://dx.doi.org/10.1016/j.asoc.2015.06.040
1568-4946/

differential harmony (SADH) search technique is used. Since, the proposed work used ELM, it has some issue such as computation of output weight that depends on input random weight and bias of the hidden nodes and choosing optimal number of hidden nodes for given problem, which is not fixed, may be different for different problem. Moreover, under practical working or testing conditions, its performance is also sensitive to noisy input which may give rise to unsatisfying accuracy [17,18]. For which author introduced Harmony Search [19–21] for optimizing the input weight, which adds extra computation cost to the model, but the choice of number of hidden neurons and activation function is left on user's choice may have a great impact on the network's usefulness. Authors in [5] proposed a structure of functional link interval type-2 fuzzy neural system (FLIT2FNS) model based on FLANN model. FLIT2FNS model consist of 5 different layers. Layer 1 and layer 2 are independent of each other. Layer 1 is simple; it only transfers the input sample to layer 4 for processing. Layer 2 acts as interval Type-2FLS. Here, each member deals with two membership functions, one lower and another upper. Nodes in layer 3 receive the membership degrees of associated rule from the nodes of layer 2. In Layer 3, two rules are generated each having two values (lower and upper). Nodes in layer 4 are called consequent nodes. The outputs obtained from layer 3 and the two local outputs of FLANN are considered as the inputs to this layer. The number of output produced by layer 4 is equal to the number of rules. Number of rules are 2. Finally, layer 5 known as defuzzification layer is used to produce output. Moreover, during training phase output of layer 5 is used by particle swarm optimization (PSO) to optimize weight and other parameter of model. This model also has certain issues in back propagation learning algorithm such as; slowness in error convergence speed and in ability to escape local optima. Thus, author used PSO algorithm to optimize the parameter of FLIT2FNS system which increases complexity and high computational cost.

To avoid the limitations of simple neural network model or any hybrid neural network this paper proposed a simple and more robust SVM and KNN algorithm that which gives better result with more accurate prediction. In this paper, a SVM [22–24] is used to classify stock data and KNN [25,26] is used to predict the stock market indices. Two Indian stock market indices such as BSE Sensex and CNX Nifty are used as the experimental data. The proposed model has been compared with the recent models such as FLIT2FNS [5] and SADHS-OELM [16]. The objective of this paper is to get in-depth knowledge in the stock market in Indian scenario with the two above mentioned stock indices using technical analysis methods and tools and to predict 1 day, 1 week, and 1 month ahead trends, volatility and momentum of stock indices in advance. This paper is organized as follows: In Section 2, dataset description, technical indicators and data normalization procedures are discussed with theoretical knowledge on SVM and K-NN as preliminaries study. Section 3 explores the schematic layout of proposed SVM-KNN based stock analysis model. Empirical experiment and result analysis and comparison with other recent models are outlined in Section 4 and finally Section 5 concludes the paper.

2. Preliminaries

2.1. Dataset description

The following are the datasets used for analysis of Indian stock market trend throughout this paper for experimental purpose.

The BSE SENSEX is the oldest and fastest stock exchange in Asia was established in 1875 with a speed of 200 microseconds. On August 2014 as many as 5000 companies are on BSE, making it world top exchange in terms of listed members. The equity market capitalization of the companies listed on the BSE was US$1.6 trillion as of June 2014, and it is also one of the world top twenty stock exchanges by market capitalization [27].

The CNX Nifty is the stock market for Indian equity market. It is managed by India Index Services and Products one of the owned subsidiary of the NSE Strategic Investment Corporation Limited. CNX Nifty offers investment managers exposure on the portfolio of 22 sectors of the Indian economy. The CNX Nifty 50 Index gives 29.70%, 0.73% and nil weightage to financial services, industrial manufacturing, agricultural sector, respectively [28].

2.2. Theoretical framework of stock market trend, momentum and volatility analysis

The stock market plays an imperative role in important Indian economy. With the remarkable explosion in the stocks from the past few months, created hype in the stock exchange in the country. As we know at different points of time, stock market goes through different phases. Sometimes, the market is bearish and sometimes it is bullish. So, it is very important to predict the stock market trend analysis. Many researchers and experts predict with the help of the direction of the market. If they notice that the overall market is rising up, then the stock prices are ready to grow. On the hand, if there is a bearish market, then the price of the stock will reduce. In order to determine whether the market is going through a bullish trend or bearish trend, we need to know whether the market is having more buyers or more sellers. When there are more sellers than buyers, then the market is going through a bearish trend. In order to know this, we have to keep a close watch on the stock price and also on the volume of the stock. We put a closer view on BSE Sensex, NSE, CNX Nifty etc., so that we can understand movement of stock market functions and can predict the future trend. In this paper, we focus not only on predicting the closing price but also the volatility of the stock market as well as momentum of the stocks. This section outlines the theoretical framework of the proposed hybridized stock market analysis method.

Step I – Time Series Dataset Generation

Let U = Angular bracket $\langle U_o, U_l, U_h, U_c\rangle$ represent the dataset, where column U_o, U_l, U_h and U_c represents *stock opening*, *stock lower*, *stock higher* and *stock closing* values respectively for any day u_i as shown in (1). Here $u_{if} \in \Re^n$ for $f = \langle o, l, h, c\rangle$.

$$U = \begin{array}{c} \\ u_1 \\ u_2 \\ \\ u_n \end{array} \begin{array}{c} \begin{array}{cccc} U_o & U_l & U_h & U_c \end{array} \\ \begin{pmatrix} u_{1o} & u_{1l} & u_{1h} & u_{1c} \\ u_{2o} & u_{2l} & u_{2h} & u_{2c} \\ \vdots & \vdots & \vdots & \vdots \\ u_{no} & u_{nl} & u_{nh} & u_{nc} \end{pmatrix} \end{array} \quad (1)$$

Considering (1) as the given dataset, most of the prediction tool will try to predict data for different time series such as; u_{n+1}, u_{n+7} and u_{n+30} as 1 day, 1 week and 1 month, respectively, in advance. For predicting with respect to those thee time horizon, classifiers need to be trained with the input in the form of (u_i, y_i). Where y_i is the ith class label for any input feature u_i. A major issue related to any stock market dataset is that it does not contain class label. Hence, in order to classify we introduce a new attribute Δc_s as change in S. Where S can be closing price or volatility or momentum can be formulated as (2).

$$\Delta c_s^i = U_{is} - U_{i+1sc} \quad \forall i = 1 \ldots n-1 \quad (2)$$

where Δc_s^i is the comparison between ith and ith + 1 closing values or volatility or momentum. If $\Delta c_s^i < 0$, it represents *loss* otherwise

profit. Hence, two class labels such as *profit* or *loss* can be assigned to for any feature that can be evaluated by using (3).

$$y_i = \begin{cases} 1 & \text{if } \Delta c \le 0 \text{ indicate profit,} \\ -1 & \text{otherwise, indicate loss} \end{cases} \quad (3)$$

Step II – Use of Technical Indicators for Smoothening Data Points

A technical indicator is a series of data points that are derived by applying a formula to the price data of a security. Price data includes any combination of the open, high, low or close over a period of time. Some indicators may use only the closing prices, while others incorporate volume and open interest into their formulas. The price data is entered into the formula and a data point is produced. For example, the average of 3 closing prices is one data point ((28,700 + 27,670 + 27,560)/3 = 27,976.66). However, one data point does not offer much information for which a series of data points over a period of time is required to create valid reference points to enable analysis. By creating a time series of data points, a comparison can be made between present and past levels. For analysis purposes, technical indicators are usually used for smoothening the data points and shown in a graphical form above or below a security's price chart. In literature many technical indicators can be found such as moving averages, price adjustments, volatility indicators, momentum indicators, trend indicators, market trend indicators, special indicators etc. [29–33]. For more smoothening the dataset and prediction few technical indicators such as simple moving average (SMA), exponential moving average (EMA), triangular moving average (TMA) and modified moving averages (MMA), Williams %R, true strength index (TSI), relative strength index (RSI), average true range (ATR), and volatility ratio (VR) are used throughout the paper and discussed below.

(a) SMA is used to smoothes the data by replacing each element with the average of the neighboring elements defined within the window. This process is equivalent to low pass filtering and is mathematically derived as (4).

$$\mathrm{MA}_i = \frac{1}{2N+1}(\mathrm{MA}(i+N) + \mathrm{MA}(i+N-1) + \cdots + \mathrm{MA}(i-N)) \quad (4)$$

where MA_i is the smoothed value for the ith data point, N is the number of neighboring data points on either side of MA_i, and $2N+1$ is the span.

EMA also known as an exponentially weighted moving average. This can also be expressed in technical analysis terms as follows, showing how the EMA steps towards the latest datum point, but only by a proportion of the difference (each time) and can be computed using (5).

$$\mathrm{EMA}_i = \mathrm{EMA}_{i-1} + \alpha X(u_{ic} - \mathrm{EMA}_{i-1}) \quad (5)$$

(b) TMA guaranties double-smooth data. It is evaluated using (6) where, it first calculates the first simple moving average with window width of cell $(N+1)/2$, it calculates a second simple moving average on the first moving average with the same window size.

$$\mathrm{TMA} = \frac{(\mathrm{MA}_1 + \cdots + \mathrm{MA}_n)}{n} \quad (6)$$

(c) MMA is also known as Running Moving Average (RMA) can be formulated as (7). Consider the argument N to be the lag of the simple moving average. The first modified moving average is calculated like a simple moving average. Subsequent values are calculated by adding the new price and subtracting the last average from the resulting sum.

$$\mathrm{MMA}_i = \frac{(N-1)_x \mathrm{MMA}_{i-1} + u_{ic}}{N} \quad (7)$$

(d) Williams %*R*, or just %*R* is a technical analysis oscillator showing the current closing price in relation to the high and low of the past N days (for a given N). It was developed by a publisher and promoter of trading materials, Larry Williams. Its purpose is to tell whether a stock or commodity market is trading near the high or the low, or somewhere in between, of its recent trading range and can be computed using (8).

$$\%R = \frac{\mathrm{high}_{N\ \mathrm{days}} - \mathrm{close}_{\mathrm{today}}}{\mathrm{high}_{N\ \mathrm{days}} - \mathrm{low}_{N\ \mathrm{days}}} \times -100 \quad (8)$$

The oscillator is on a negative scale, from −100 (lowest) up to 0 (highest), obverse of the more common 0 to 100 scale found in many technical analysis oscillators. A value of −100 means the close today was the lowest low of the past N days, and 0 means today's close was the highest high of the past N days.

TSI is a double smoothed indicator; meaning that a moving average applied to the data (daily momentum in this case) is smoothed again by a second moving average. The calculation for TSI uses exponential moving averages and can be formulated using (9).

$$\mathrm{TSI}(c_0, r, s) = 100 \times \frac{\mathrm{EMA}(\mathrm{EMA}(m, r), s)}{\mathrm{EMA}(\mathrm{EMA}(|m|, r), s)} \quad (9)$$

where, c_0 = today's closing price, $m = c_0 - c_1$ = momentum (difference between today's and yesterday's close), EMA (m, n) = exponential moving average of m over n periods, r = EMA smoothing period for momentum, typically 25 and s = EMA smoothing period for smoothed momentum, typically 13.

(e) RSI is classified as a momentum oscillator, measuring the velocity and magnitude of directional price movements. The RSI is most typically used on a 14 day time frame, measured on a scale from 0 to 100, with high and low levels marked at 70 and 30, respectively. The relative strength factor is then converted to a relative strength index between 0 and 100 and computed using (10).

$$\mathrm{RSI} = 100 - \frac{100}{1 + \frac{\mathrm{EMA}(U,n)}{\mathrm{EMA}(D,n)}} \quad (10)$$

where U is upward changes and D is downward changes is calculated for the first n series.

(f) ATR is a technical analysis volatility indicator originally developed by J. Welles Wilder, Jr. for commodities. The average true range is an N-day exponential moving average of the true range values. Wilder recommended a 14-period smoothing. The range of a day's trading is simply high–low. The true range extends it to yesterday's closing price if it was outside of today's range.

$$\mathrm{true\ \ range} = \max\left[(\mathrm{high},\ \mathrm{low}),\ \mathrm{abs}(\mathrm{high} - \mathrm{close}_{\mathrm{prev}}),\ \mathrm{abs}(\mathrm{low} - \mathrm{close}_{\mathrm{prev}})\right] \quad (11)$$

The ATR at the moment of time t is calculated using (12):

$$\mathrm{ATR}_t = \frac{\mathrm{ATR}_{t-1} X (n-1) + \mathrm{TR}_t}{n} \quad (12)$$

The idea of ranges is that they show the commitment or enthusiasm of traders. Large or increasing ranges suggest traders prepared to continue to bid up or sell down a stock through the course of the day. Decreasing range suggests declining interest.

(g) VR is technical analysis indicator used to identify price ranges and breakouts and can be calculated using (13):

$$\mathrm{VR} = \frac{\mathrm{True\ \ Range}}{\mathrm{True\ \ Range}_{\mathrm{for\ \ last\ \ } n \mathrm{\ \ periods}}} \quad (13)$$

Calculation of VR is based on true range. For n parameters is usually used value of 14.

Step III – Time Series Dataset Regeneration and Normalization

After applying technical indicators, now the dataset U can be reformulated as (14)

$$U = \begin{array}{c} \\ u_1 \\ u_2 \\ \\ \\ u_n \end{array} \begin{array}{c} \begin{array}{ccccccccc} U_o & U_l & U_h & U_c & \Delta c_s & TI_1 & TI_2 & TI_5 & y \end{array} \\ \begin{pmatrix} u_{1o} & u_{1l} & u_{1h} & u_{1c} & \Delta c_s^1 & TI_{11} & TI_{12} & TI_{15} & y_1 \\ u_{2o} & u_{2l} & u_{2h} & u_{2c} & \Delta c_c^2 & TI_{21} & TI_{22} & TI_{25} & y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_{no} & u_{nl} & u_{nh} & u_{nc} & \Delta c_s^n & TI_{n1} & TI_{n2} & TI_{n5} & y_n \end{pmatrix} \end{array} \quad (14)$$

where TI_i are the technical indicators as discussed in preceding section. All the time series datasets are first scaled between 0 and 1 using *min-max normalization* (15).

$$\tilde{u}^{ij} = \frac{u_{ij} - u_{\min j}}{u_{\max j} - u_{\min j}} \quad (15)$$

where u_{ij} is the jth attribute value of ith feature u_i, and $u_{\min j}$ and $u_{\max j}$ are the jth minimum and maximum value of the datasets respectively, and $\tilde{u}^i$ is the scaled price of ith day.

Step IV – Performance Evaluation

Closing Price: For measuring the performance of the model for predicting the closing the Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE) are used, which are defined in (16) and (17).

$$\mathrm{MAPE} = \frac{1}{T}\sum_{i=1}^{T}\left|\frac{d_i - \hat{d}_i}{d_i}\right| \times 100 \quad (16)$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(d_i - \hat{d}_i)^2} \quad (17)$$

where T is the total number of testing data, d_i *and* $\hat{d}_i$ is the desired and predicted outputs, respectively.

Volatility: Comparison of volatility forecasts is conducted for (1 day, 1 week, 1 month) ahead horizon in terms of Mean Squared Forecast Error (MSFE), Root Mean Squared Forecast Error (RMSFE) and Mean Absolute Forecast Error (MAFE) defined as (18)–(20).

$$\mathrm{MSFE} = \frac{1}{N}\sum_{i=1}^{N}(\vartheta_i - \widehat{\vartheta}_i)^2 \quad (18)$$

$$\mathrm{RMSFE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\vartheta_i - \widehat{\vartheta}_i)^2} \quad (19)$$

$$\mathrm{MAFE} = \frac{1}{N}\sum_{i=1}^{N}|\vartheta_i - \widehat{\vartheta}_i| \quad (20)$$

where ϑ_i is the realized volatility and $\widehat{\vartheta}_i$ is the predicted volatility.

Momentum: Comparison for momentum is made on (1 day, 1 week, 1 month) ahead horizon is done in terms of MSFE, RMSFE and MAFE defined in (21)–(23).

$$\mathrm{MSFE} = \frac{1}{N}\sum_{i=1}^{N}(\mathfrak{M}i - \widehat{\mathfrak{M}i})^2 \quad (21)$$

$$\mathrm{RMSFE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathfrak{M}i - \widehat{\mathfrak{M}i})^2} \quad (22)$$

$$\mathrm{MAFE} = \frac{1}{N}\sum_{i=1}^{N}|\mathfrak{M}i - \widehat{\mathfrak{M}i}| \quad (23)$$

where $\mathfrak{M}i$ is the realized momentum and $\widehat{\mathfrak{M}i}$ is the predicted momentum.

2.3. SVM and its parameter discussion

SVM is a supervised learning algorithm used for data classification [22–24] which classifies data in large datasets by identifying separating surface. Separating surface can be linear or non-linear separating surface in the input space of a dataset. Given a training set with label pairs (u_i, y_i), $i = 1, \ldots, n$ where $u_i \in \Re^n$ and, $y \in \{\text{profit, loss}\}^i$, SVM requires solution of the following optimization problem (24).

$$\min_{w,b,\xi}\frac{1}{2}w^T w + C\sum_{i=1}^{n}\xi_i$$

$$\text{Subject to:} \quad y_i(w^T\Phi(u_i)) + b \geq 1 - \xi_i, \xi_i \geq 0. \quad (24)$$

Where decision function is given by (25).

$$p = sgn\left(\sum_{i=1}^{n} y_i\alpha_i K(x_i, x) + \rho\right). \quad (25)$$

Training vector u_i is mapped into higher dimensional space by the *kernel function* ϕ. The separating surface depends on a subset of the original data known as a set of *support vectors*. $C > 0$ is the *penalty parameter* of the error term. SVM constructs a hyper plane or set of hyper planes known as *functional margin* in a high dimensional space, which separates the data into number of classes. Many times dataset is not linearly separable, for which finding a linear solution in two dimensions becomes impossible. To overcome this challenge dataset can be cultivated into multi-dimension using kernel functions, $k(u_i, u_j) \equiv \Phi(u_i)^T\Phi(u_j)$. SVM is trained using different kernels as given in (26)–(28).

(a) Linear kernel: $(u_i, u_j) = u_i^T u_j,$ (26)

(b) Polynomial kernel: $k(u_i, u_j) = (\gamma u_i^T u_j + r)^d, \gamma > 0,$ and (27)

(c) Radial Basis kernel (RBF):

$$k(u_i, u_j) = \exp(-\gamma\|u_i - u_j\|^2),\ \gamma > 0. \quad (28)$$

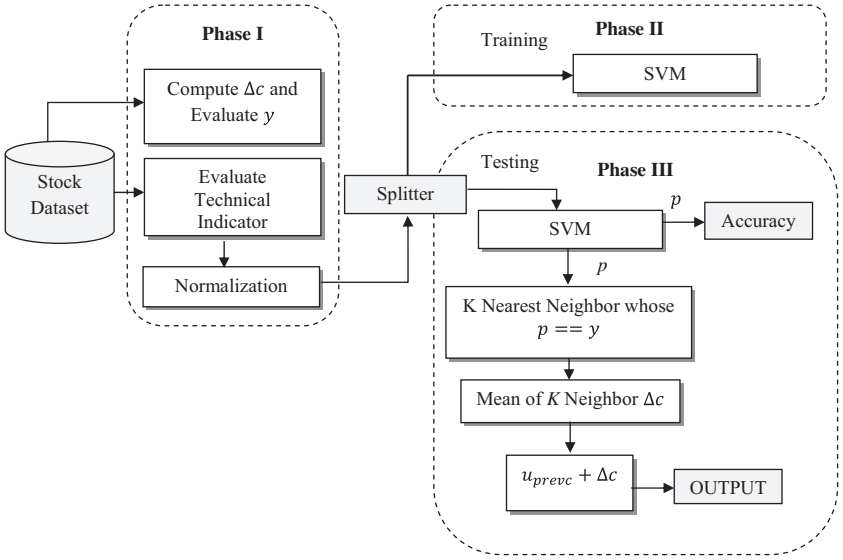

Fig. 1. Schematic layout of proposed SVM-KNN stock analysis model.

Here, C, γ, r, and d are kernel parameters which are initialized depending on the dataset. The choice of best kernel parameter is affected by the size of training set. SVM can both be linear (predicting linearly separable dataset) and non linear (predicting not linearly separable dataset). The major strengths of SVM are the training which is relatively easy as it does not have any local optimal, unlike the neural network. It scales relatively well to high dimensional data and the trade-off between classifier complexity and error can be controlled explicitly. The weakness includes the need for a good kernel function. Looking into the discontinuities among the dataset we have used exponential radial function. The choice of appropriate kernel function can differ with respect to choice of dataset. The datasets with various combination of parameters $\{C, \gamma\}$ has been implemented, in which the parameter C is chosen from $\{2–5, 2–4, \ldots, 25\}$ and γ from $\{2–15, 2–14, \ldots, 2–1\}$.

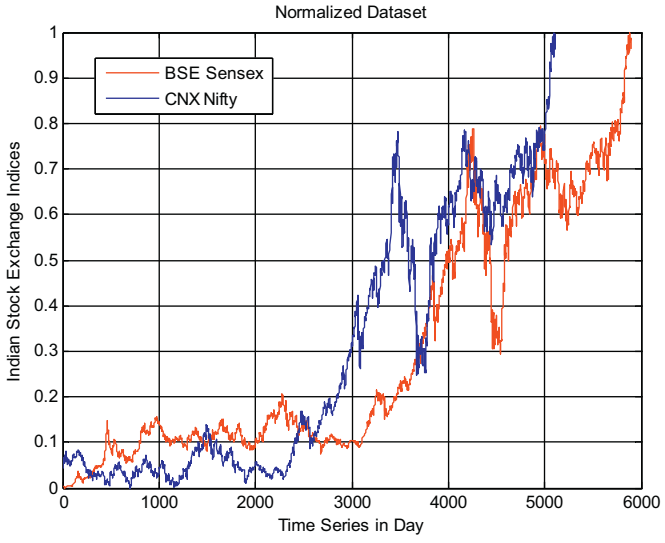

Fig. 2. Normalized stock indices – BSE Sensex and CNX Nifty.

2.4. K-Nearest Neighbor search (KNN)

KNN makes predictions by identifying the k nearest neighbors closest to the testing data. Thus, for KNN, the metric for measuring the distance between the query point and cases from the examples sample is required. One of the most popular choices to measure this distance is known as Euclidean. Other measures include Euclidean squared, City-block, and Chebyshev as given in (29).

$$D(x,p) = \begin{cases} \sqrt{(x-p)^2} & \text{Euclidean} \\ (x-p)^2 & \text{Euclidean squared} \\ \text{Abs}(x-p) & \text{Cityblock} \\ \text{Max}(|x-p|) & \text{Chebyshev} \end{cases} \tag{29}$$

where, x and p are the query point and a case from the example sample, respectively. Distance D is then sorted in increasing order and first k elements are picked as the neighbors. The most frequent class of this neighbor is predicted as the class label of x.

3. Schematic layout of proposed SVM-KNN stock analysis model

The proposed stock market analysis model is shown in Fig. 1. This model works in three phases. In phase1, first change in closing price or volatility or momentum is evaluated using (2) and class label is identified using (3). Later the dimension of dataset is increased by adding technical indicators by using (4)–(13) depending upon the requirement. After application of technical indicators data sets are regenerated using (14) and the data points are normalized and mapped within range [0–1] using (15). In phase2, dataset is divided into two parts, one part is used for training and other for testing. The model is trained using SVM classifier. In phase3, testing data is classified using SVM. Output of SVM is either +1 or −1 indicated as p, which represents *profit* or *loss*. This value of p is compared with class label y to compute accuracy of the model. Again from previous data, k nearest neighbors is searched whose class label is equal to p. Mean of those k nearest neighbor is evaluated and stored as Δc. Result is then added with previous closing value, volatility

Table 1
Description of data samples and data range.

Datasets	Total samples	Data range	Training samples	Testing samples
BSE Sensex	5894	01-01-1990 to 11-08-2014	500	200
CNX Nifty	4686	01-01-1994 to 12-08-2014	1000	400

Table 2
Description of parameters used for SVM and KNN.

Dataset	Parameter value/horizon prediction					
	1 day		1 week		1 month	
	SVM$\{C, \gamma\}$	k for KNN	SVM$\{C, \gamma\}$	k for KNN	SVM$\{C, \gamma\}$	k for KNN
BSE Sensex	$\{2^{-2}, 2^{-10}\}$	10	$\{2^{-2}, 2^{-12}\}$	10	$\{2^{-3}, 2^{-2}\}$	10
CNX Nifty	$\{2^{-2}, 2^{-10}\}$	10	$\{2^{-2}, 2^{-12}\}$	10	$\{2^{-3}, 2^{-2}\}$	10

or momentum as the predicted value in advance. The algorithm of SVM-KNN for analysis of the stock market is given as follows:

Algorithm: *Stock Market Analysis Model*
Input: $\boldsymbol{U}, \boldsymbol{w}, \boldsymbol{c}, \boldsymbol{\lambda}, \boldsymbol{k}$
Where $\boldsymbol{U}$ consist of opening, high, low and closing indices of any stock, and $\boldsymbol{w}$ is the size of data need to be trained from $\boldsymbol{U}$. Parameter c, λ, k is used for tuning SVM and KNN
Output: ***Predicted_closing_value***
Step 1: Evaluate Technical Indicator (TI) from Stock indices U using equation (4 to 13).
Step 2: Unite U and TI as one matrix as shown in (14)
$U \leftarrow [U, TI]$
Step 3: Evaluate y and Δc
Step 3: compute Ũ using (15) and initialize M ← 0
Step 4: *for* $iter \leftarrow 1 : |U| - w$
$Training\_set \leftarrow U(iter : w + iter - 1, :)$;
$Testing\_data \leftarrow U(w + iter, :)$;
$Compute\,[w, \rho] \leftarrow Train\_SVM(Training\_set, c, \lambda)$
$p \leftarrow Test\_SVM(Testing\_data, w, \rho)$
If $p == y$ *then Increment M*
$\mathcal{F} \leftarrow k\,nearest\,neighbor\_search(training_{set}, p)$
$\bar{\Delta c} \leftarrow \frac{1}{|\mathcal{F}|} \sum \Delta c_{\mathcal{F}}$
$Predicted\_closing\_value(iter) \leftarrow (last\_closing\_value) + \bar{\Delta c}$

$error(iter) \leftarrow actual\_closing\_value(iter) - Predicted\_closing\_value(iter)$
end of for
Step 8: Plot Actual vs Predicted result
Step 9: *evaluate accuracy of SVM* $\leftarrow M/(|testing\_set|) * 100$;

4. Empirical experiment and result analysis

Two significant stock market datasets as discussed in Section 2 such as; BSE Sensex and CNX Nifty are used for experiment. The model is used to predict stock indices for 1 day, 1 week and 1 month in advance. Inputs are normalized between 0 and +1 using (15) and the normalized value of dataset is shown in Fig. 2 and is divided into two parts: one for training and another for testing in the ratio as given in Table 1. Table 1 depicts the dataset used and their dimensions. BSE Sensex has 5894 number of samples from Jan 1990 to August 2014. Similarly from Jan 1994 till August 2014, total number of sample in CNX Nifty found to be 4686. For our model, last 700 samples has been taken from BSE Sensex, out of which 500 samples are used for training and last 200 samples are used for testing. Similarly last 1400 samples have been taken from CNX Nifty where 1000 samples are used for training and last 400 samples are used for testing. The 1 day, 1 week and 1 month stock closing price or volatility or momentum of BSE Sensex and CNX Nifty datasets are shown in Figs. 3 and 8 respectively. Few important technical indicators discussed in Section 2.2 are taken into consideration along with the daily closing prices or volatility or momentum of the respective stocks and computed using (4)–(14). Latter the MAPE and RMSE is evaluated to measure the performance of the trained models and the performance of volatility and momentum of stock data is measured using MSFE, RMSFE, and MAFE.

The performance metrics for the above mentioned two datasets for closing price, volatility of the market and the momentum has been obtained by evaluating executions of the algorithm on the same training and testing sets. Target vs. predicted values and error convergence speed is used to present empirical results. Parameter for SVM classifier (C, γ) are taken as $[\{2^{-2}, 2^{-10}\}, \{2^{-2}, 2^{-12}\}, \{2^{-3}, 2^{-2}\}]$ and $[\{2^{-2}, 2^{-10}\}, \{2^{-2}, 2^{-12}\}, \{2^{-3}, 2^{-2}\}]$ for BSE Sensex and CNX Nifty for 1 day, 1 week and 1 month respectively, whereas, the k value KNN for dataset is taken 10 for both datasets as well as for all three time instances as shown in Table 2. Figs. 4 and 9 show the closing indices, volatility and momentum of each dataset after applying the appropriate technical indicators. The technical indicators such as, SMA, EMA, TMA, MMA, Williams %R, TSI, RSI, ATR and VR are applied for smoothing the closing price, volatility as well as momentum. Figs. 5 and 10 show the comparison of target vs. predicted stock closing prices during testing of two datasets for 1 day, 1 week and 1 month in advance respectively. Figs. 6 and 11 depict the volatility prediction of both the datasets, whereas momentum prediction of both datasets for different horizon is shown in. Table 3 depicts accuracy achieved by SVM classifier for predicting two class label *profit* or *loss* for the two datasets; it can be observed that SVM is performing 91–96% accurately while predicting the *profit* or *loss*. Table 4 shows the comparison of MAPE during testing for proposed model with other two recent model FLIT2FNS [5] and CEFLANN [16] with two datasets. In Table 4, data for the algorithm FLIT2FNS and CEFLANN has been taken from their paper. In CEFLANN author has taken three time instances such as 1 Day, 5 Days and 10 Days, therefore the same is shown in Table 4. In case of 1 day ahead prediction the average MAPE for all the datasets stands at 0.165. Similarly, for 1 week prediction the average MAPE turns to be 0.375 and for 1 month average MAPE is 0.54.i

Table 5 depicts the performance comparison of SVM-KNN hybridized model for 1 day ahead volatility prediction with CEFLANN, whereas; Table 6 shows the performance measure of SVM-KNN for 1 week and 1 month volatility prediction for both the datasets. Table 7 shows the performance comparison of SVM-KNN hybridized model for 1 day, 1 week and 1 month ahead momentum prediction. Wilcoxon Signed Rank test has been performed to

Table 3
Performance analysis of SVM in percentage (%) for predicting *Profit* or *Loss* for 1 day, one week and 1 month ahead.

Datasets	Time horizon		
	1 day	1 week	1 month
BSE Sensex	95	96.66	94.73
CNX Nifty	94.3	93.33	91.66

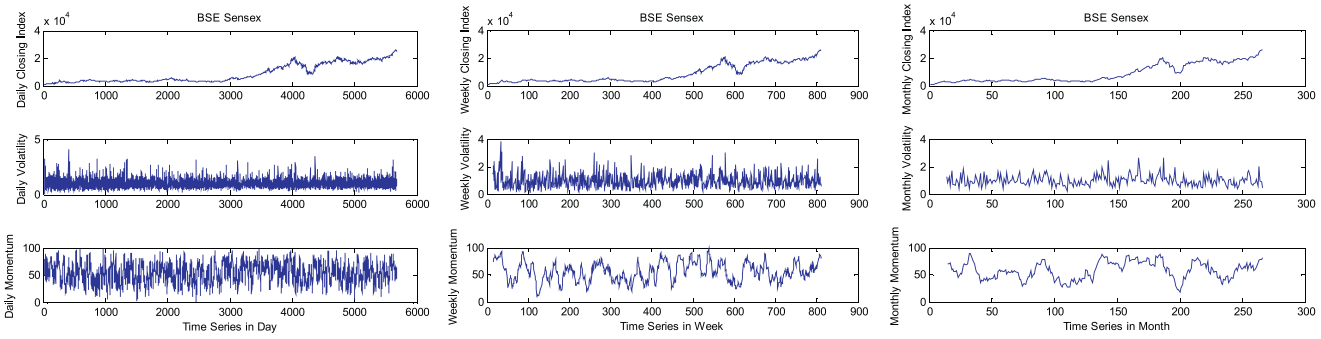

Fig. 3. Opening price, volatility and momentum of BSE Sensex for 1 day, 1 week and 1 month ahead.

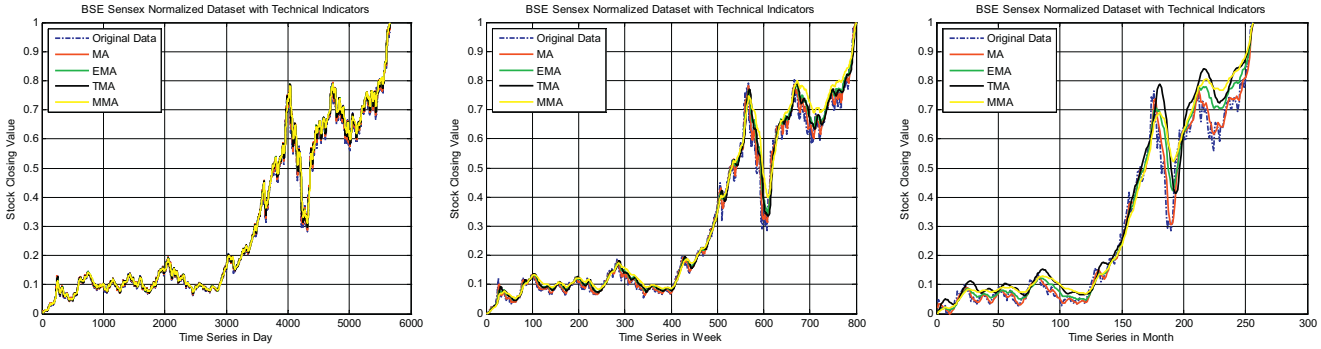

Fig. 4. Technical indicators of BSE Sensex for 1 day, 1 week and 1 month ahead.

show whether the difference between the performance metrics of proposed model are statistically meaningful or not and the critical level in the test is taken as 0.05. $h = 1$ indicates a rejection of the null hypothesis at the 5% significance level and $h = 0$ indicates a failure to reject the null hypothesis at the 5% significance level. The p and h values of the statistics are specified in Table 8. Fig. 13 depicts the error convergence speed of two datasets for all three time instances.

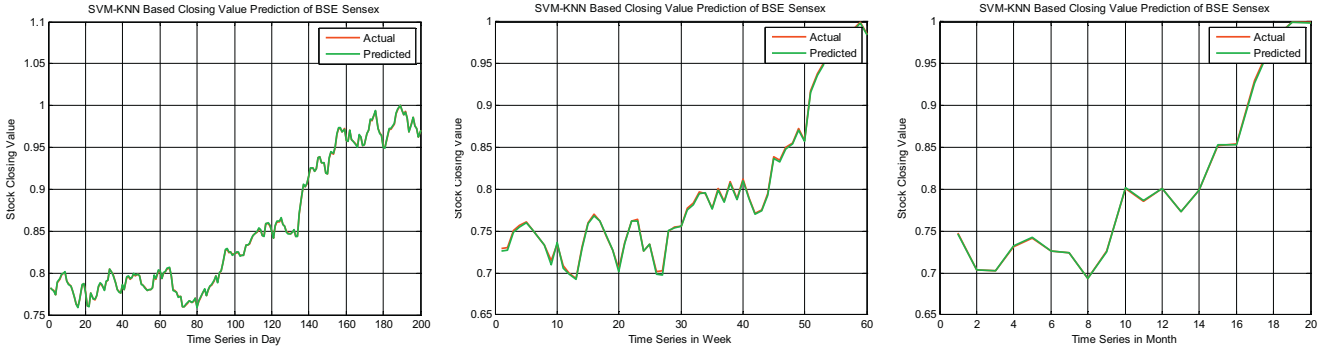

Fig. 5. Closing value prediction of BSE Sensex for 1 day, 1 week and 1 month ahead.

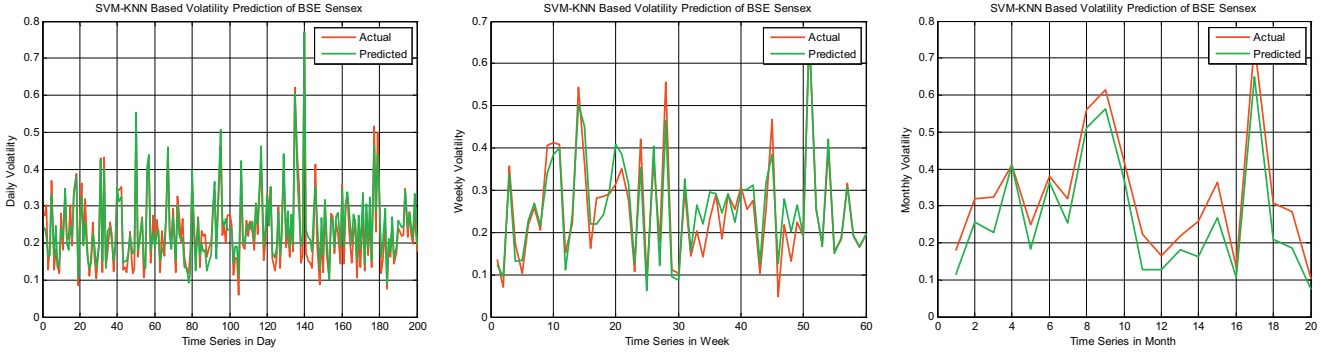

Fig. 6. Volatility prediction of BSE Sensex for 1 day, 1 week and 1 month ahead.

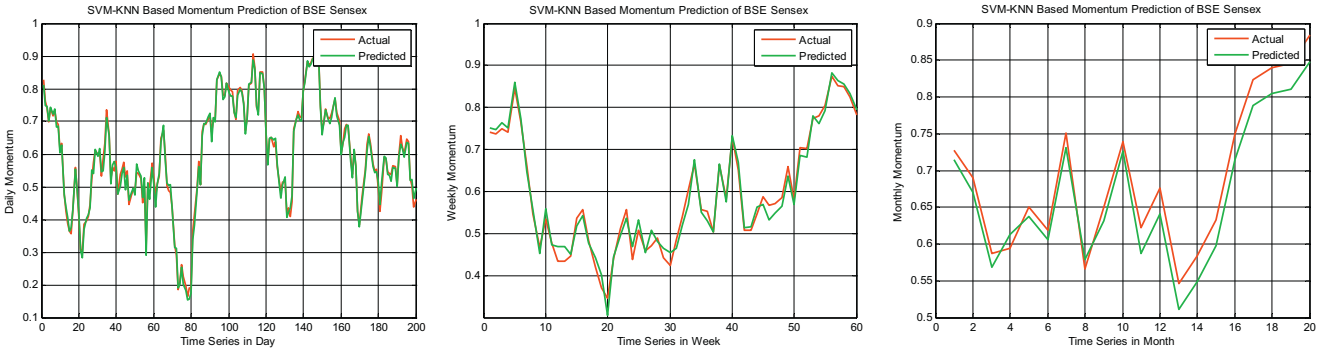

Fig. 7. Momentum prediction of BSE Sensex for 1 day, 1 week and 1 month ahead.

Table 4
Performance comparision using MAPE of SVM-KNN hybridized model for 1 day, 1 week and 1 month ahead with FLIT2FNS and CEFLANN.

Datasets	FLIT2FNS [5] Time horizon			CEFLANN [16] Time horizon			SVM-KNN method Time horizon		
	1 day	1 week	1 month	1 day	5 days	10 days	1 day	1 week	1 month
BSE Sensex	0.23	0.43	0.61	0.56	1.3	1.8	0.0650	0.1870	0.1123
CNX Nifty	NA	NA	NA	0.64	1.5	2.0	0.0915	0.2234	0.58

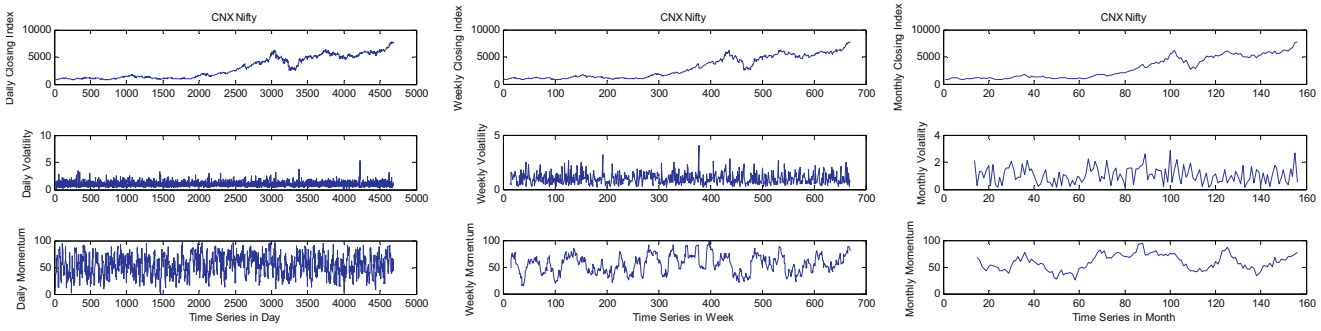

Fig. 8. Opening price, volatility and momentum of CNX Nifty for 1 day, 1 week and 1 month ahead.

Table 5
Performance comparision of SVM-KNN hybridized model for 1 day ahead volatility prediction with CEFLANN.

Datasets	CEFLANN [16]			SVM-KNN method		
	MSFE	RMSFE	MAFE	MSFE	RMSFE	MAFE
BSE Sensex	0.09	0.30	0.22	0.0012	0.0349	0.0299
CNX Nifty	0.16	0.40	0.28	0.0012	0.0347	0.0304

This paper can be summarized as follows:

Comparison of proposed SVM-KNN model with CEFLANN and FLIT2FNS model shows the higher forecasting performance. Looking to Table 4, clearly shows proposed model is 5 to 15 times more accurate then FLIT2FNS model and Table 5 shows that proposed method is 10 to 15 times better then CEFLANN model in respect to 1 day ahead prediction.

Table 6
Performance comparision of SVM-KNN hybridized model for 1 week and 1 month ahead volatility prediction.

Datasets	Time horizon	SVM-KNN Method		
		MSFE	RMSFE	MAFE
BSE Sensex	1 week	0.0019	0.0431	0.0340
	1 month	0.0046	0.0681	0.0614
CNX Nifty	1 week	3.8052e−004	0.0195	0.0147
	1 month	0.0038	0.0617	0.0524

Proposed method does not required many parameter and iteration to train and test, thus reduces complexity and computational cost, whereas CEFLANN required harmonic search for parameter tuning and FLIT2FNS needs additional algorithm for weight optimization.

Table 7
Performance comparision of SVM-KNN hybridized model for 1 day, 1 week and 1 month ahead momentum prediction.

Datasets	SVM-KNN Method Momentum			
	Time Horizon	MSFE	RMSFE	MAFE
BSE Sensex	1 day	2.0434e−004	0.0143	0.0110
	1 week	3.5714e−004	0.0189	0.0161
	1 month	7.5487e−004	0.0275	0.0256
CNX Nifty	1 day	5.6803e−004	0.0238	0.0193
	1 week	4.0927e−004	0.0202	0.0175
	1 month	0.0016	0.0396	0.0387

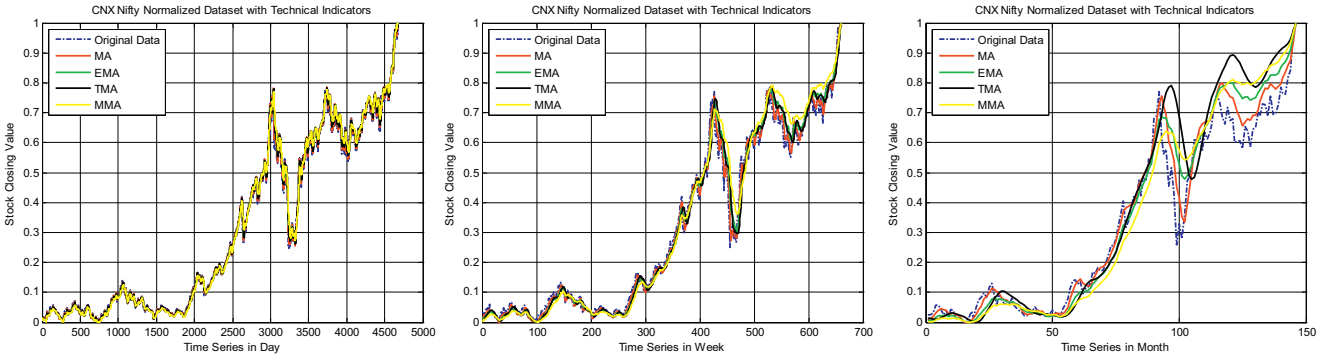

Fig. 9. Technical Indicators of CNX Nifty for 1 day, 1 week and 1 month ahead.

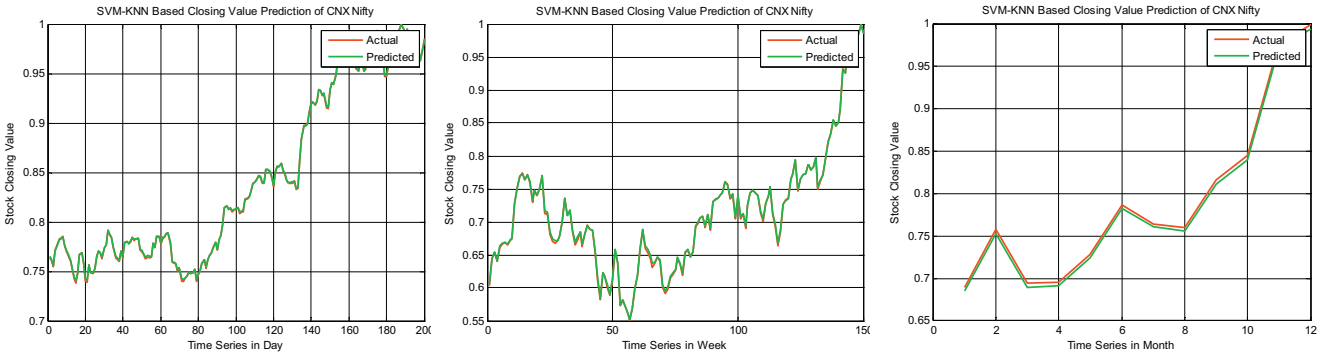

Fig. 10. Closing value prediction of CNX Nifty for 1 day, 1 week and 1 month ahead.

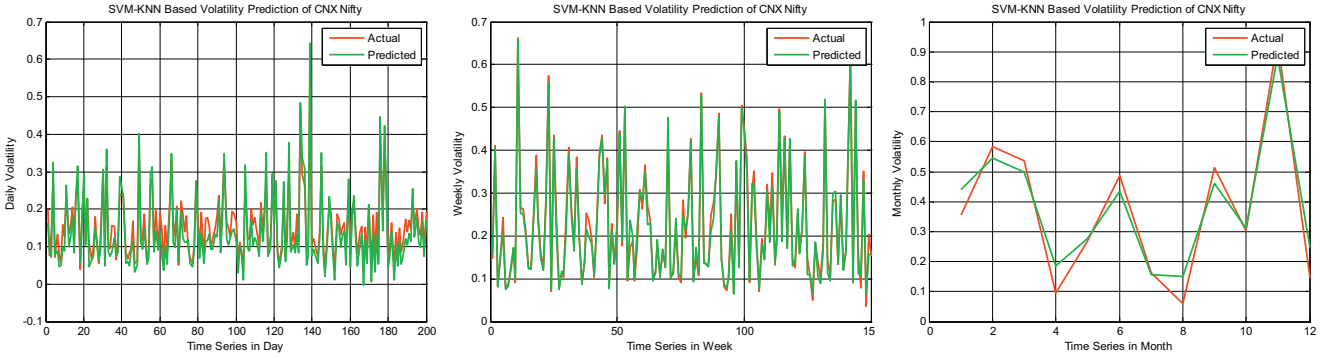

Fig. 11. Volatility prediction of CNX Nifty for 1 day, 1 week and 1 month ahead.

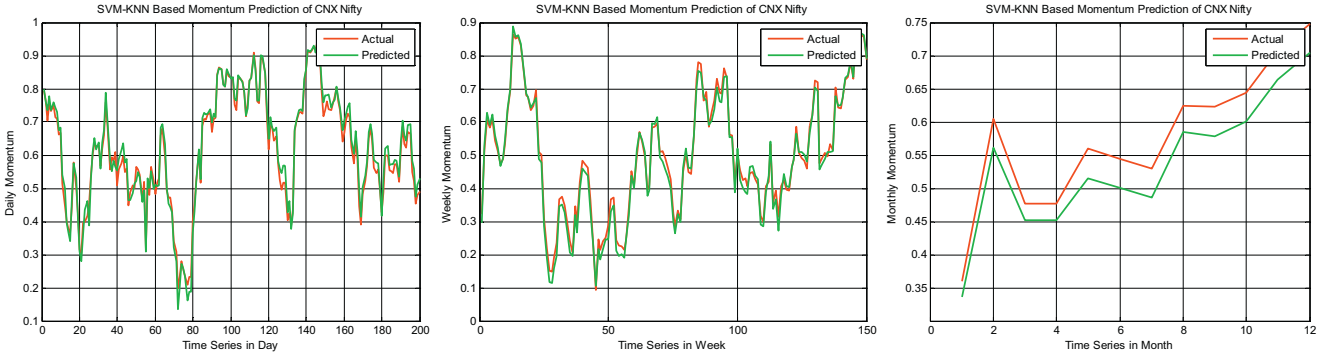

Fig. 12. Momentum prediction of CNX Nifty for 1 day, 1 week and 1 month ahead.

Table 8
p, *h*, *zval* and *signedrank* value of Wilcoxon signed rank test.

Datasets	Time horizon	*p*, *h*, *zval* & *signedrank*	Actual closing/predicted	Actual volatility/predicted	Actual momentum/predicted
BSE Sensex	1 day	*p*	1.5637e−011	1.6059e−004	8.2754e−005
		h	1	1	1
		zval	−6.7419	−3.7741	−3.9363
		signedrank	4663	6957	6824
	1 week	*p*	2.6707e−009	0.1490	0.8888
		h	1	0	0
		zval	−5.9507	−1.4432	−0.1399
		signedrank	108	719	896
	1 month	*p*	0.2503	9.9939e−005	2.0212e−004
		h	0	1	1
		zval	−1.1496	−3.8907	−3.7163
		signedrank	75	1	7
CNX Nifty	1 day	*p*	7.3085e−008	1.5599e−013	4.9662e−009
		h	1	1	1
		zval	−5.3834	−7.3820	−5.8483
		signedrank	5708	4001	5257
	1 week	*p*	1.9673e−006	0.0022	4.1475e−008
		h	1	1	1
		zval	−4.7568	−3.0634	−5.4845
		signedrank	3147	4030	2740
	1 month	*p*	4.8828e−004	0.5566	4.8828e−004
		h	1	0	1
		zval	–	–	–
		signedrank	0	31	0

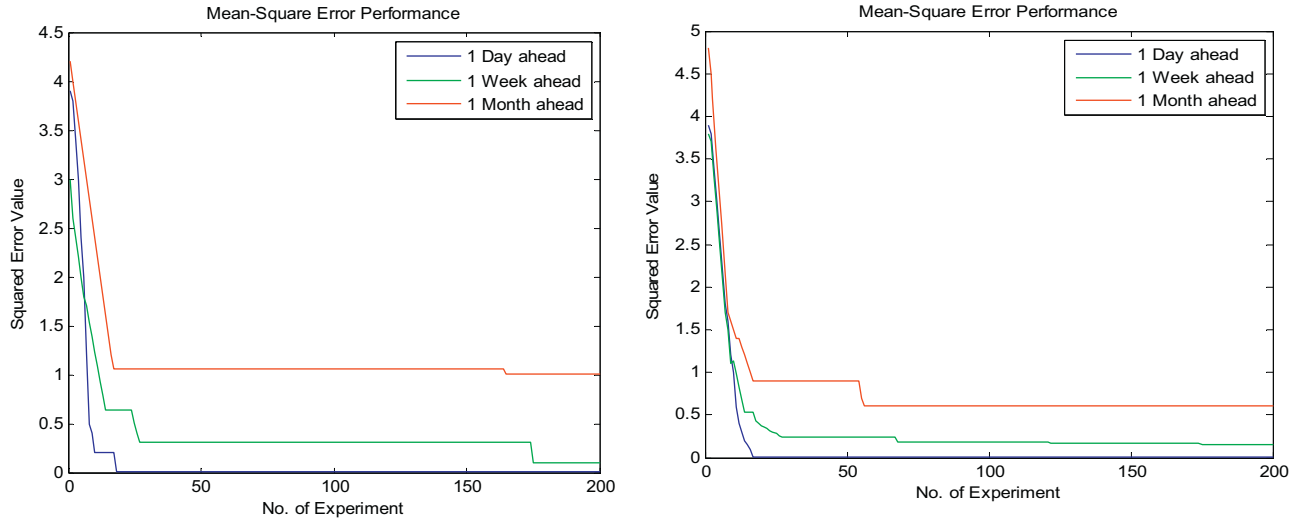

Fig. 13. Convergence speed of BSE SENSEX and CNX Nifty using Mean-Squared Error for 1day, 1 week and one month ahead.

Proposed model fully depends upon the classification result of SVM denoted as *p*, and then KNN algorithm compute the *k* neighbor which has similar class label *p*. Next day indices are predicted by simply adding last day indices with the mean of those *k*-neighbors's Δc_s. This process filters unwanted data to participate for prediction, thus reduces error by high margin.

Proof can be easily realized over the predicted graph shown in Figs. 4–7 and 9–12, where the actual and predicted curve are almost hard to distinguish.

5. Conclusion

An integrated model comprising SVM and KNN system has been proposed to predict two stock market indices such as, BSE Sensex and CNX Nifty for l day, l week, and l month in advance. This model has been compared with FLIT2FNS and CEFLANN. From the results, it is clearly established that the model not only predicts *profit* or *loss* but also guides the user to analyze the stock data with respect to its closing price, volatility, and momentum stock indices in advance. The simulation results clearly depicts that the SVM-KNN based stock analysis model, where training with SVM and analyzes with KNN offers significant improvement over other two compared models by overcoming the problem of choosing too many parameters of ANN and fuzzy based model and also the prediction capability has been improved for Indian stock market datasets.

References

[1] T.H. Abdoh, H. Jouhare, The investigation of efficiency of stock price index of T.S.E., J. Financ. Res. l3 (11–l2) (1996).
[2] K.S. Ravichandran, P. Thirunavukarasu, R. Nallaswamy, R. Babu, Estimation on return on investment in share market through ANN, J. Theor. Appl. Inf. Technol. 3 (2007) 44–54.
[3] V. Kodogiannis, A. Lolis, Forecasting financial time series neural network and fuzzy system based technique, Neural Comput. Appl. 11 (2002) 90–92.
[4] L. Yu, S. Wang, K.L. Keung, Mining Stock Market Tendency Using GA-Based Support Vector Machine, Springer Verlag, Berlin/Heidelberg, 2005, pp. 336–345, LNCS 3828.

[5] S. Chakravarty a, P.K. Dash, A PSO based integrated functional link net and interval type-2 fuzzy logic system for predicting stock market indices, Appl. Soft Comput. 12 (2012) 931–941.
[6] G. Dutta, P. Jha, A.K. Laha, N. Mohan, Artificial neural network models for forecasting stock price index in the Bombay stock exchange, J. Emerg. Market Financ. 5 (2006) 3.
[7] R. Chodhury, K. Garg, A hybrid machine learning system for stock market forecasting, in: Proceeding of World Academy of Science, Engineering and Technology, vol. 29, 2008, ISSN l307-6884.
[8] Y. Chen, X. Dong, Y. Zhao, Stock index modeling using EDA based local linear wavelet neural network, in: IEEE Proceedings of International Conference on Neural Networks and Brain, vol. 3, 2005, pp. l646–l650.
[9] B. Majhi, H. Shalabi, M. Fathi, FLANN based forecasting of S&P 500 index, Inf. Technol. J. Asian Netw. Sci. Inf. 4 (3) (2005) 289–292.
[10] R. Majhi, G. Panda, G. Sahoo, Development and performance evaluation of FLANN based model for forecasting stock market, Expert Syst. Appl. 36 (2009) 6800–6808.
[11] H. Fu-Yuan, Integration of an improved particle swarm optimization algorithm and fuzzy neural network for Shanghai stock market prediction, in: Workshop on Power Electronics and Intelligent Transportation System, IEEE, 2008, pp. 242–247, 978-07695-3342.
[12] P.-C. Chang, C.-Y. Fan, S.-H. Chen, Financial time series data forecasting by wavelet and TSK fuzzy rule based system, Institute of Electrical and Electronics Engineers, Fourth International Conference on Fuzzy Systems and Knowledge Discovery, IEEE (2007), 0-7695-2874.
[13] C.-H. Cheng, T.-L. Chen, H.-J. Teoh, Multiple-period modified fuzzy time series for forecasting TAIEX, in: Institute of Electrical and Electronics Engineers, Fourth International Conference on Fuzzy systems and Knowledge Discovery, IEEE, 2007, 0-7695-2874.
[14] L. Yu, Y.Q. Zhang, Evolutionary fuzzy neural networks for hybrid financial prediction, Inst. Electr. Electron. Eng. Trans. Syst. Man Cybern. 35 (2) (2005) 244–249.
[15] S. Chokri, Neuro-fuzzy network based on extended Kalman filtering for financial time series, in: Proceeding of World Academy of Science, Engineering and Technology, vol. 15, 2006, ISSN 1307-6884.
[16] R. Dash, P.K. Dash, R. Bisoi, A self adaptive differential harmony search based optimized extreme learning machine for financial time series prediction, Swarm Evol. Comput. 19 (2014) 25–42.
[17] K. Javed, R. Gouriveau, N. Zerhouni, SW-ELM: a summation wavelet extreme learning machine algorithm with a priori parameter initialization, Neurocomputing 123 (2014) 299–307.
[18] M. Luo, K. Zhang, A hybrid approach combining extreme learning machine and sparse representation for image classification, Eng. Appl. Artif. Intell. 27 (2014) 228–235.
[19] K.S. Lee, Z.W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, Comput. Methods Appl. Mech. Eng. 194 (2005) 3902–3933.
[20] R. Diao, Q. Shen, Feature selection with harmony search, IEEE Trans. Syst. Man Cybern. B: Cybern. 42 (6) (2012) 1509–1523.
[21] B. Wu, C. Qian, W. Ni, S. Fan, Hybrid harmony search and artificial bee colony algorithm for global optimization problems, Comput. Math. Appl. 64 (2012) 2621–2634.
[22] V. Cortes Cortes, Vapnik, Support-vector networks, Machine Learning 20 (3) (1995) 273.
[23] C.-W. Hsu, C.-J. Lin, A comparison of methods for multiclass support vector machines, in: IEEE Transactions on Neural Networks, 2002.
[24] Press, H. William, Teukolsky, A. Saul, Vetterling, T. William, B.P. Flannery, Section 16.5. Support vector machines, in: Numerical Recipes: The Art of Scientific Computing, 3rd ed., Cambridge University Press, New York, 2007.
[25] T. Bailey, A. Jain, A note on distance-weighted k-nearest neighbor rules, IEEE Trans. Syst. Man Cybern. 8 (1978) 311–313.
[26] S. Bermejo, J. Cabestany, Adaptive soft k-nearest-neighbour classifiers, Pattern Recognit. 33 (2000) 1999–2005.
[27] http://www.world-exchanges.org/statistics/monthly-reports
[28] Sectoral weightage of CNX Nifty 50 Index. http://www.nseindia.com/content/indices/ind_cnx_nifty.pdf
[29] B. Magda, Fayek, Multi-objective optimization of technical stock market indicators using gas, Int. J. Comput. Appl. 68 (20) (2013), 0975-8887.
[30] http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:williams_r
[31] C. Michael, Thomsett, The options trading body of knowledge: the definitive source for information, 268, FT Press, 2009, pp. 2009.
[32] Blau. William, True strength index, Tech. Anal. Stocks Commod. (traders.com) 11 (1) (1991) 438–446.
[33] J.J. Murphy, The Visual Investor: How to Spot Market Trends, 2nd ed., John Wiley and Sons, 2009, pp. 100.