Cluster Comput
https://doi.org/10.1007/s10586-017-1553-5

Map-Reduce framework based cluster architecture for academic student's performance prediction using cumulative dragonfly based neural network

M. R. M. VeeraManickam[1] · M. Mohanapriya[1] · Bishwajeet K. Pandey[2] · Sushma Akhade[3] · S. A. Kale[3] · Reshma Patil[4] · M. Vigneshwar[1]

Received: 2 November 2017 / Revised: 13 December 2017 / Accepted: 14 December 2017

Abstract The major aim of the education institute is to provide the high-quality education to students. The way to attain the high quality in the education system is to determine the knowledge from the educational data and learn the attributes which influence the performance of the students. The extracted knowledge is used to predict the academic performance of the students. This paper presents the student performance prediction model by proposing the Map-reduce architecture based cumulative dragonfly based neural network (CDF-NN). The CDF-NN is proposed by training the neural network by the cumulative dragonfly algorithm (DA). Initially, the marks of the students from semester 1 to semester 7 are collected from different colleges. In the training phase, the features are selected from the student's information and the intermediate data is generated by the mapper. Then, the intermediate data is provided to the reducer function which is built with the CDF-NN to provide the estimated marks of the students in a forthcoming semester. The proposed method is compared with the existing methods, such as Dragonfly- NN and Back prorogation algorithm for the evaluation metrics, MSE and RMSE. The proposed prediction model obtains the MSE of 16.944 and RMSE of 4.665.

✉ M. R. M. VeeraManickam
manic.veeramrm@gmail.com

[1] Department of Computer Science Engineering, Karpagam Academy of Higher Education (Deemed to be University), Coimbatore, India

[2] Gyancity Research Lab, Gurgaon, India

[3] KJ College of Engineering and Management Research, Pune, India

[4] Trinity College of Engineering and Research, Pune, India

Keywords Educational data mining · Map-Reduce framework · Cluster · Neural network · Dragonfly algorithm · Smart E-learning

1 Introduction

Assessment of educational data is one of the emerging and encouraging fields that consents the progressing of methods, which explore the data that originates from e-learning daises. The data attained by the application of educational data mining (EDM) [1–4] can be utilized for several things, like granting suggestions to academic planners especially in higher education institutes for the enhancement of their decision-making process, proceeds corresponding actions for underperforming students [5] [6] [7]. Since the EDM is more beneficiary to learners, i.e., students, it has arisen as a state of art research area in recent years [1–4]. The student performance prediction model is considered as a most significant application provided by EDM. Performance prediction model depends on various factors [8], and some of the factors are deliberated for EDM. Recently, the researchers are paying great attention in researching the e-learning behavior analytics, prediction analytics, and so on [4,9,10]. The prediction of student performance through e-learning behavior analytics can aid developers to assess e-learning system efficiently, can progress system availability and develop the function of system function by which the behavior of the system can easily visualize [11] and also makes way for future development. This prediction model helps the teachers to know the trend of student behavior, to enhance the curriculum as well as the quality of teaching.

One of the most renowned prediction models is Artificial Neural Network (ANN) based prediction model [12–14]. Further, NN-based (neural network) [15], performance pre-

Published online: 07 February 2018

diction or estimating models are specified as the demographic background inputs to the model. This model increases the accuracy of prediction when compared to other regression-based prediction model. The enhancement in accuracy of NN-based modeling purely depends on the training algorithm. Training algorithm in NN is concerned as a vital role since they grant the correct output for given input, which is based on the weight of neuron. Hence, the selection of optimal weight for NN training is considered as the major optimization problem [16] [17]. So, a number of optimization algorithms have been revised for training ANN, like genetic algorithm (GA), cuckoo search (CS), firefly (FF) [18,19], etc. However, the inadequateness of accuracy is still reflected as a major barrier, which needs to have an efficient optimization algorithm for providing fast convergence rate, less complexity, prompt solution achievement, and so on.

Recently, numbers of data mining approaches are used for predicting the performance of the students, but they are not suitable for the huge volume of data. In order to handle the big data, one of the modern programming models is the Map-Reduce which simplify the distributed applications. Various parameters are defined to the job for configuring the Map-Reduce framework. Numbers of map and reducer jobs, location, and format of the input are specified by the user. In Hadoop Map-Reduce framework, the input data is divided into a number of pieces. Every input data is provided to the map function. In map function, the input data is processed, and the intermediate data is generated. The data is sorted in both the map and reduce phase and kept in partitions as stated by the reducer operations. In Hadoop, the HashPartitioner is utilized. The combiner is specified by the user for aggregating the intermediate results to minimize the overhead in data transmission. Then, the map-reduce framework gathers all pairs with the similar key from every list and group them together, and the reduce function is provided to every group in a parallel manner [20]. For predicting the performance of the student, the Map-Reduce concept is implemented by dividing the student data and carry out the data mining operations. Then, the aggregated result is provided. The performance of the Map-Reduce is based on the student data distribution. Here, the workload distribution is based on the data partitioning algorithm [21]. The map-reduce framework increases the accuracy of the prediction model.

In this paper, the student performance prediction model is introduced by proposing the Map-Reduce cluster architecture based CDF-NN. The CDF-NN is proposed by training the neural network by DA. Initially, the marks of the students from semester 1 to semester 7 are collected from different colleges. Then, this information is provided to the Map-Reduce framework. In the training phase, the features are selected from the student's information and the intermediate data is generated. Then, the intermediate data is provided to the reducer function which is built with the CDF-NN, and the trained model is generated. At the testing phase, the information of the students is provided to the map function which is built with the trained CDF-NN and the intermediate data is generated. Then, the intermediate data is provided to the reducer function which provides the estimated marks of the students in semester eight.

The contribution of the paper is as follows:

- This paper proposes a new nature-inspired algorithm called, cumulative dragonfly (CDF) by modifying the dragonfly optimization.
- The CDF-NN is proposed by training the neural network with cumulative dragonfly algorithm.
- The Map-Reduce based cluster architecture based student performance prediction model is proposed to predict the final semester marks of the students.

This paper is organized as follows: Sect. 2 presents the literature survey. Section 3 presents the proposed Map-Reduce framework based cluster architecture for Academic Student's Performance Prediction using CDF-NN. Results and Discussion are provided in Sects. 4, and 6 concludes the paper.

2 Literature survey

Cristoba Romero et al. analyzed the many recent research works in student performance prediction model [22]. The major impartial of the mentioned prediction model was predicting the last semester presentation of a new learner via their contribution in online discussion media. A number of models had been used for 'Multiple linear regression' and classifier named SVM classifier for the prediction purpose. The results obtained have shown that the data gathered for prediction was incomplete terms. Hence the simplification of prediction way was impossible. Subsequently, the mentioned prediction model had organized on clustering models; it's complex and even more substantiated model. Kotsiantis had presented prediction model based on the regression method [2]. The main purpose of this prediction model was predicting the marks of the learner for distant learning model. The performance of developed prediction was better when compared to other conventional regression, and the outcome has wide-ranging outcomes, which has given the impression as an issue.

Behrouz Minaei-Bidgoli *et al*. [3] had introduced a prediction model on the basis of multiple classifiers. The purpose of the developed model classification was to predict the grade of students. The accuracy of prediction model attained better performance when compared to other classifier that has used on the multiple classifiers combined one. In order to enhance the prediction of the performance, they have used a genetic algorithm. The results of the model have attained

maximum accuracy. However, the complexity that blend of multiple classifiers has listed as, "Quadratic Bayesian classifier, " 'Parzen-window,' 'MLP,' and 'Decision Tree' and so on is unbearable" [3]. The attained accuracy of the multi-classifier model has minimized as it has added some additional attributes, which was chosen for prediction. Annika Wolff et al. [4] have utilized the decision tree classifier as well as SVM model in their developed prediction model. The major idea was to point the learners who were aids in the trainer participation. The developed integrated models have used a decision classifier as well as SVM for the purpose of features' prediction, which was mainly for the evaluation of scores. The impartial element chosen was based on demographic, and the evaluation was inexact that have supported in compromise. Nevertheless, the complexity of prediction was not so better for utilizing the collective practice of SVM as well as Decision tree classifiers collected.

One of the other systems, which have integrated the 'Decision tree' classifier, and NB-classifier, and, it was proposed by Camilo.E. Lopez et al. [5]. The major principle belongs with the developed prediction model was to evaluate the students who are underperforming from the initial-year deployment. The model has utilized two major classification techniques, for enhancing perceptive as well as further to that quality evaluates the predicted data. The developed model has attained the prediction accuracy that was varied around 50–57%. Additionally, Ernesto Pathros Ibarra García et al. [6] have developed a prediction model based on the classifier method. The developed prediction model was mainly for forecasting the first-year student. They have achieved 70% prediction accuracy with the aid of 'Naïve Bayes classifier.' O. C. Asogwa has attained accurate prediction with the aid of ANN as the classification of the performance of academic students [23]. To overwhelm the conventional issues that were listed have been done by introducing some prediction model. Hence the problem has focused on only one optimization algorithms DA as modified one-CDF-NN for student's marks prediction utilizing day's scholar's assessments marks as a dataset to predict student's marks, so it aids in improving forthcoming performance valuations.

Ke Wu et al [31] have proposed an approach based on a back-propagation neural network (BPNN) with an HR map (BPNN_HRM), in which a supervised model is introduced into sub-pixel land-cover change detection (SLCCD). The performance of this algorithm was evaluated by comparing this algorithm with four conventional methods. The results of the experimentation showed that this method outperformed the methods in providing a more detailed map for change detection. Seyedali Mirjalili [1] had presented a swarm intelligence optimization technique named DA. The author considered the proposal of binary and multi-objective versions of DA called binary DA (BDA) and multi-objective DA (MODA), respectively. These algorithms improved the initial random population for a given problem, converge towards the global optimum, and provide very competitive results compared to other well-known algorithms.

2.1 Research gap identified

Even though the methods progressed in literature shows many advantageous performances in predicting the performance of the student still, the requirements are broadly spread, which must be recovered by finding some effective models. The decision support system [2] needs to show the varied problem impact of students' achievement, and also it could not help the instructor in developing the homework even effectively. A genetic algorithm for data mining [3] could not classify the students' problem directly. Moreover, it needs additional Evolutionary Algorithms for the classification purpose. The proposed data mining model [5] however, minimizes the performance rate if some extra data is added as second enrollment. Further, the algorithm requires some assumption of independence property. The prediction model [6] cannot add additional information, i.e., here the training set is restricted. The additional information for prediction might improve the efficiency of predicting the students' performance. In terms of execution time, the MapReduce model [24] does not show significant breakthrough. The map-reduce framework in [21] requires long processing time.

3 Proposed Map-Reduce framework based cluster architecture for academic student's performance prediction using cumulative dragonfly based neural network

This section presents the proposed Map-Reduce framework based cluster architecture for Academic Student's Performance Prediction using CDF-NN. The aim of this work is to predict the final semester mark of the students based on the marks obtained by those students from semester 1 to semester 7. The proposed student prediction model utilizes the map-reduce framework, and the neural network is trained with the cumulative Dragonfly algorithm. In the training phase, the marks of the students are gathered from different colleges and applied to the mapper functions, mapper 1, mapper 2, and so on. Then, the mapper functions select the features and generate the intermediate data. After that, the intermediate data is applied to the reducer function. The reducer function is built with the CDF-NN. At last, the trained model is generated. In the testing phase, the marks stored in the datasets are applied to the mapper function which is built with the trained CDF-NN. Then, the intermediate data is generated which is provided to the reducer function. The reducer function provides the estimated marks of semester eight.

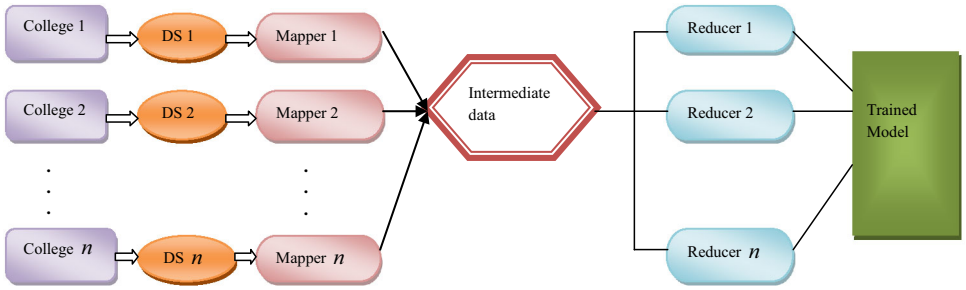

Fig. 1 Training architecture of the proposed student performance prediction model

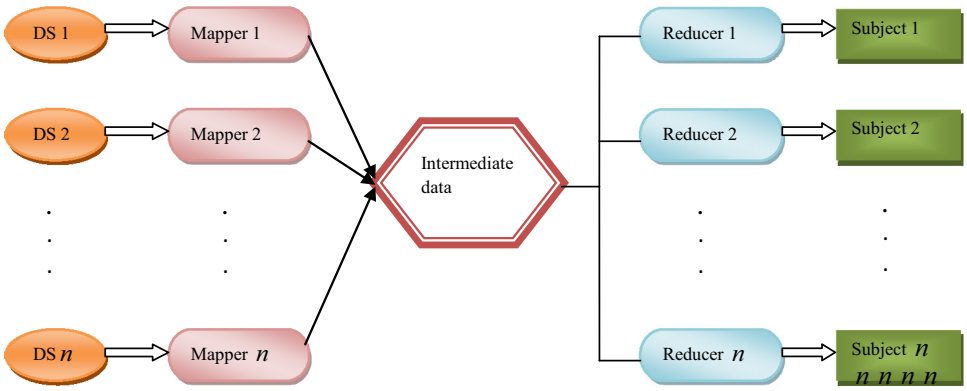

Fig. 2 Testing architecture of the proposed student performance prediction model

3.1 Map-Reduce cluster architecture for student performance prediction model

This section presents the Map-Reduce architecture for the proposed student performance prediction model. Figure 1 shows the training architecture of the proposed student performance prediction model. Initially, the marks of the students from the first semester to the seventh semester are collected. The first dataset consists of marks of the students from college 1 and the second dataset consists of marks of the students from college 2. Similarly, the dataset n consists of marks from college n. In the figure, $DS\ 1$, $DS\ 2$,..., $DS\ n$ represent the Datasets. Then, the marks are provided to the mapper functions. The marks in the dataset 1 are provided to the mapper 1, the marks in the dataset 2 are provided to the mapper 2. Similarly, the marks in the dataset n are provided to the mapper n. The mapper functions select the features of each data, and the intermediate data is generated. Then, the intermediate data is provided to the reducer function which is built with the CDF-NN. At last, the trained model is generated by the reducer function.

Figure 2 shows the testing architecture of the proposed student performance prediction model. In testing, the marks in the datasets are provided to the mapper functions which are built with the trained CDF-NN. The mapper functions produce the intermediate data which is then provided to the reducer functions. The reducer functions provide the estimated marks of the students in the semester 8. Reducer 1 produces the estimated marks of subject 1; Reducer 2 produces the estimated marks of the subject 2. Similarly, Reducer n provides the estimated marks of the subject n.

3.2 Proposed CDF-NN for student performance prediction model

Figure 3 shows the proposed student performance prediction model. Here, the input variables are the marks of students which are considered as the population of Dragonfly. The marks of the students from semester 1 to semester 7 are collected and provided to the CDF-NN. The CDF-NN is constructed by training the NN with the DA for optimally predicting the performance of the students. The input variables are processed in CDF-NN, and the marks of the students are predicted for semester eight.

The aim of the developed prediction model is as follows. The major purpose of the developed prediction model is to predict the last semester mark, i.e., the 8^{th}-semester mark of each student S based on the data collected (marks of semester1 to semester 7). Let A_i is the collected data, which is defined in Eq. (1), where n denotes the number of collected data.

$$A_i = A_1, A_2, \cdots A_n \tag{1}$$

The features are extracted from the collected data, which is based on entropy measures. Further, the evaluation of entropy value for extracted feature set is done, and the feature with less entropy value is chosen as the best feature. The best fea-

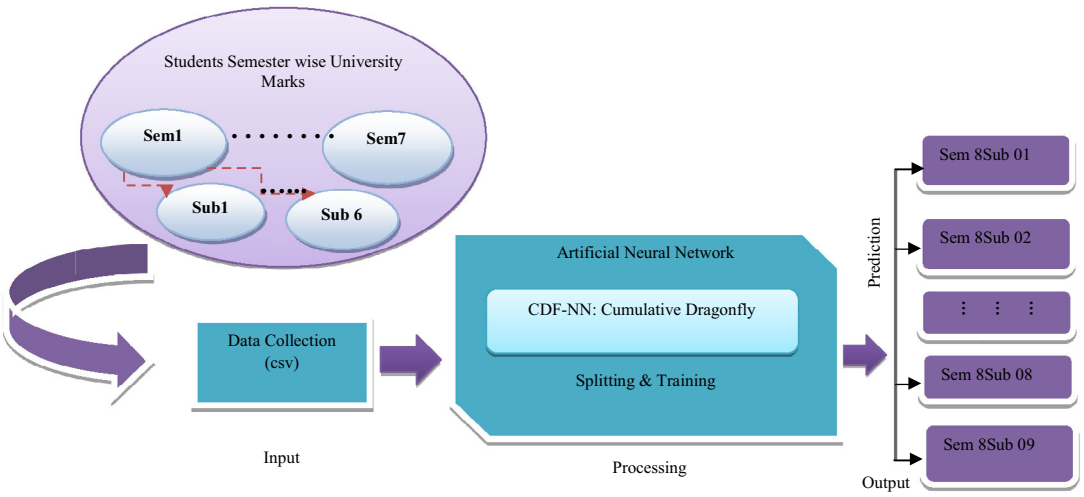

Fig. 3 Proposed model of student performance prediction model

ture for prediction is determined in Eq. (2), where m denotes the selected best feature.

$$A_f = \left\{A_{f_1}, A_{f_2}, \cdots A_{f_m}\right\} \tag{2}$$

The selected feature set is given as the input to NN for predicting the performance of the student in the final semester. Here, the optimal weight for prediction is selected using the offered CDF-NN algorithm. The solution vector that denotes the optimal weight of neuron in NN is defined as given below:

$$w = w_1, w_2, \cdots w_0 \tag{3}$$

3.2.1 Data collection

In the developed prediction model, data collection is the initial step. Data, like marks of each student from semester 1 to semester 7 are collected. The semester marks include the marks of internal assessment, marks of a laboratory experiment, attendance marks, marks of model examination conducted in college, etc. The students for data collection are selected from different departments, like computer science and engineering, electrical and electronics, electrical, civil, mechanical, and so on. From the collected data, one can get the corresponding detail of the students in terms of their sincerity and dedication in studies and other activities too, which also helps the teachers to categorize the students as well. Data connected with entire semester marks of students are gathered in CSV file format, which is passed on NN as input neurons to the developed system. From the collected data, as per the requirement of NN, they are grouped into two: Input variable and output variable. As mentioned before, input variables are the marks of students (semester 1 to semester 7). The output variables determine the performance of students. From the collected input, the performance of the student in the 8^{th} semester is predicted by which the students those who are underperforming can be further improved. Here, A_i denotes the data collected from students S.

3.2.2 Feature selection using hierarchical clustering

Usually, the assumption behind the 'feature selection' algorithms is that the worthy feature subset comprises the features that are highly related to the class labels, whereas the elements are uncorrelated with one other. This basis is quite same to data clustering, which divides the data into different clusters (or groups), and hence the fellows of a cluster similar to one other than the fellows in other cluster [8]. Let d be the dataset containing N instances. The N instances are grouped into K different clusters, where the instances in the same group are more similar to each other, and at the same time, they are dissimilar from other different groups. Good results can be achieved by considering within-cluster distance S^W and between-cluster distance S^B. Here, two kinds of clusters are formed by grouping the features: chosen cluster S^C and numerous candidate clusters. Each feature in S^C has already selected for the separation of the classes C. Each candidate cluster comprises single candidate feature f, which has still not been chosen. In this situation, f with minimum within-cluster distribution S^W with S^C and maximal among-cluster dispersion S^B with the class cluster C^C will be collected with S^C. Finally, the particular cluster S^C is the final selected subset.

Clustering distance The distance between the candidate cluster f and the selected cluster S^C is defined as the sum

of distance $R\left(f', f\right)$ among f and all feature points f' in S^C, which are defined in Eq. (4).

$$S^C(f) = \sum_{f' \in S^C} R\left(f', f\right) \tag{4}$$

Eq. (4) is the attained within-cluster distance $S^W\left(S^C\right)$ of S^C, and it could be attained accumulatively. Firstly, $S^W\left(S^C\right)$ is zero and then add $S^C(f)$ at a time when f has been merged with S^C as shown below:

$$S^W\left(S^C \cup f\right) = S^W\left(S^C\right) + S^C(f) \tag{5}$$

As per this analysis, for f, the evaluation function $E(f)$ is defined as given in Eq. (6), where $|S^C|$ denotes the number of elements in S^C, $S^B\left(C, S^C\right)$ denotes the class cluster distance of the chosen cluster and class cluster C and $J(f; C)$ denotes the mutual information of f and C.

$$E(f) = \frac{S^B\left(C, S^C\right) + J(f; C)}{|S^C| + S^W\left(S^C \cup f\right)} \tag{6}$$

Feature clustering Based on the evaluation function $E(f)$, a specific feature clustering is adopted as given in Algorithm 1. Initially, an initiation step is essential, where each f in P' is prepared as the candidate cluster, the class labels that denoted as C are reserved as the class cluster C. Next to this, for each f, evaluation of $S^B(C, f)$ takes place. The candidate cluster f with the largest $E(f)$ value is assigned to the selected cluster S^C. Then, the hierarchical clustering iteration is processed, where the candidate cluster f with the largest $E(f)$ is accumulated with S^C. The clustering process does not terminate unless the features in S^C are larger than the pre-specified threshold value.

Algorithm 1: Feature clustering

	Algorithm: Feature clustering
1	**Input:** training dataset $T' = (d, P', C)$
2	**Output:** The chosen feature subset S^C
3	**Parameters:** S^C →chosen cluster, f →candidate feature, C^C →class cluster
4	**Begin**
5	Initialize parameters $P = P'; S^C = \phi, C^C = C$
6	Evaluate the distance $S^B(C^C, f)$ for each f
7	$P = P - \{f\}; S^W = 0$
8	while
9	$\|S^C\| < \delta$ do
10	For
11	each candidate cluster $f \in P$ do
12	Calculate $E(f)$ as Eq. (6)
13	Choose f with largest $E(f)$
14	Clustering S^C with f
15	update S^W and S^B of S^C
16	End
17	Return S^C as the final selection result
18	End while
19	**End**

3.2.3 Prediction model using CDF-NN

In any ANN based prediction model, the results depend on appropriate input set. In the developed prediction model, entire A_i is denoted as inputs neurons and are given as the input to NN. In this system, the 8th-semester marks are predicted by considering the university assessment marks of semester 1 to 7, and this is for the day scholar students. This principle can only succeed by training and learning of the supervised learning model, which reduced the error occurred among the predicted and target outputs. In this prediction model, the new marks of the student are predicted using the CDF-NN through dragonflies' position vector that highly enhances each new learning cycle. The predictions result the list as SEM 8 sub1, SEM 8 sub2 and so on.

(a) Architecture of CDF-NN

In this model, ANN intends to predict the last semester marks of engineering college students. This is a basic model on biological NN and considered as one of the significant intelligent machine learning technique. Further, Feed Forward NN (FNN) [25–27] is the traditional forward method; it feeds data as the input from the front to the back manner and gets the outputs as the corresponding manner. NN has three levels of layers termed as an input layer, hidden layer, and an output layer. Each level layer is connected to every neuron of the next layers. The modest practical network, which is illustrated in Fig. 4, has 2 'input' cells (neuron) and a single output cell (neuron), where the model logic gates can be utilized. The two-layer FNN is extensively used as the most popular NN in numerous real-time applications [27] since it is well known that the learning process is most essential for any NN. FNNs trains in back-propagation model, which means NN network signifies the datasets of 'what measures must be carried out for getting expected results' [27]. This learning strategy is also termed as a supervised learning method, in contrast to this, the unsupervised learning only gives the input, and the target is unknown. The efficiency of the process is based on the error that being backpropagated, which is mentioned as the variance of the deviation among input and the output (MSE). Since the network accommodates with enough hidden layers, it models the linking at each time among the input and output layer.

The developed CDF-NN architecture for students' presentation prediction model is described in this section. In any NN model, neurons comprise series of layers. As mentioned earlier, NN is made up of three layers: (i) input layers, (ii) output layers, and (iii) hidden layers. Here, the first layer is not so logical one since it gives only the input. The proposed CDF-NN prediction model has used FNN, which recruits the output function from input to output layer. The Network architecture of developed CDF-NN model is diagrammatically represented in Fig. 4.

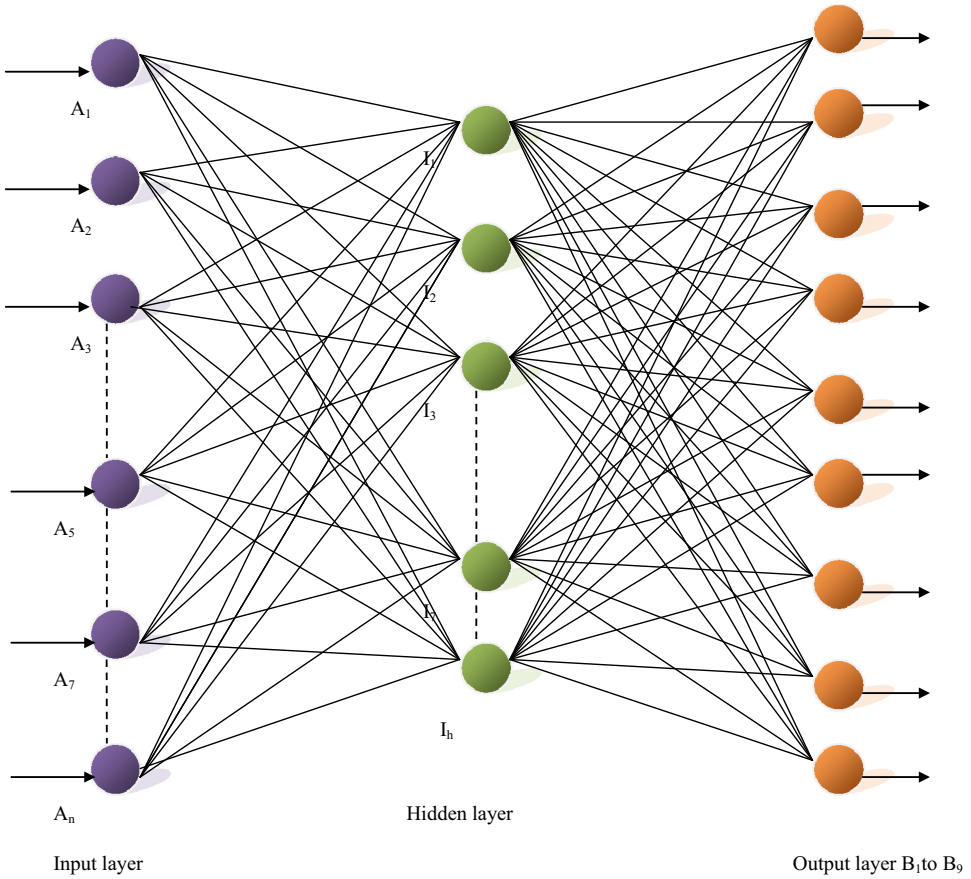

Fig. 4 Architecture of ANN. Network architecture of the proposed CDF-NN model

In Fig. 4, input vectors are represented as A_1 to A_n, which is defined in Eq. (7), hidden vectors are denoted as I_1 to I_h, and output vectors are represented as O_1 to O_9, as defined in Eq. (8) and the corresponding neurons are specified as n, h, m. The process of NN depends on the weight of neuron. Further, the set of neurons' weights are used to training NN along with their biases

$$A = A_1, A_2, \cdots A_n \tag{7}$$

$$O = O_1, O_2, \cdots O_m \tag{8}$$

where n denotes the number depicted as input features, and m refers the output count. The determination of the count of neurons, as well as the count of layers in a neural network, is the major objective, which is also mentioned in the preliminary steps. The vital influences are either in smaller or larger in correspondence with neurons and layers that predict the performance of network grants enhanced prediction outcomes with less error rate. The data moves only in forward direction, and the output of a hidden neuron can be defined as per Eq. (9), where $F(.)$ denotes the node transfer or activation function that utilized for all nodes in NN. Usually, nonlinear functions include sigmoid function are chosen as a function, node transfer function [2]. The output of hidden neuron as per sigmoid function can be given as per Eq. (10).

$$I_r = F(h_o) \tag{9}$$

$$I_r = \frac{1}{1 + (\exp(-h_o))}; o = 1, 2, \cdots h \tag{10}$$

$$h_o = \sum_{p=1}^{n} \left(X_{po} A_p\right) - \theta_o \tag{11}$$

where h refers to the number of hidden layers, X_{po} denotes the connection weight of hidden as well as the input neuron, θ bias of hidden neuron, A is the input node and n denotes the count of hidden neurons. The NN output is defined as given in Eq. (12), where I_o specifies the output of hidden neuron, X_{oq} specifies the weight among hidden and output neuron and θ_q refers to neuron's bias .

$$O_q = \frac{1}{1 + \left(\exp\left(-b_q\right)\right)} \tag{12}$$

$$b_q = \sum_{o=1}^{h} \left(X_{o,q}, I_r\right) - \theta_q; q = 1, 2, 3.....m \tag{13}$$

(b) Cumulative dragonfly algorithm: training of neural network

Solution encoding The solution encoding is diagrammatically illustrated in Fig. 5. In the Cumulative Dragonfly

Fig. 5 Representation of solution encoding

algorithm, the solution represents the weight W_i (where, $i = 1, 2, \cdots N$, N refers the number of samples) of each neuron. In the training phase, the neural network model is trained to obtain the optimal weight as the output.

Fitness function The fitness function used in the proposed Dragonfly training algorithm for NN is mainly for finding the evaluation of mean square error (MSE). The calculation of MSE is achieved using the actual and desired response of NN, which is defined in Eq. (14), where n denotes the input and m denotes the output, D' is the preferred response in NN and O specifies the actual result value of NN

$$MSE\left(X_{pq}\right) = \frac{1}{nm}\sum_{p=1}^{n}\sum_{q}^{m} D'_{pq} - O_{pq} \tag{14}$$

MSE is calculated for all solutions of DA, and the solution with the best fitness is selected as the optimal weight. Terminate criteria is aided in weight selection and updating MSE. NN is trained by reducing error function. The error function that utilized in this developed model is MSE, which is defined in Eq. (15).

$$MSE = \frac{1}{nm}\sum_{p=1}^{n}\sum_{q}^{m} D'_{pq} - O_{pq} \tag{15}$$

(c) Proposed cumulative dragonfly training algorithms

In this paper, a CDF is proposed to train the NN classifier. Dragonfly algorithm has numerous advantages. It improves the initial random population for a given problem, converges towards the global optimum, and provides very competitive results when compared to other optimization algorithms. It solves single-objective, discrete, and multi-objective problems and determines the very accurate results. In the proposed CDF, the basic aim of any group is survival. Thus, the individuals (dragonfly) in a group must be attracted toward the food source and distracted to external enemies. With the consideration of these two behaviors, five factors are concerned in location updating of individuals (dragonfly). They are (1) Control cohesion (2) Alignment (3) Separation (4) Attraction (5) Distraction. Addition to these factors, the proposed method determines a factor as (6) Cumulative memory property. This adoption of cumulative property enhances the performance of DA model in terms of an accurate optimal solution.

Step 1: Separation Separation denotes the static collision avoidance of dragonfly (individuals) from further individuals in the neighborhood. The separation of r^{th} a dragonfly, Z_r makes its neighbors, which is evaluated as per Eq. (16), where Y' refers the position of a current dragonfly, Y'_s denotes the location of s^{th} neighboring dragonfly and B' refers the count of neighboring dragonflies.

$$Z_r = \sum_{s=1}^{B'}\left(Y' - Y'_s\right) \tag{16}$$

Step 2: Alignment Alignment refers to the velocity matching of dragonfly (individuals) to other individuals (dragonflies) in the neighborhood. The alignment is evaluated as per Eq. (17), where Q'_s denotes the velocity of s^{th} neighboring dragonfly.

$$K_r = \frac{\sum_{s=1}^{B'} Q'_s}{B'} \tag{17}$$

Step 3: Cohesion Cohesion denotes the preference of individuals towards the center of the mass of neighborhood. The cohesion is assessed as given in Eq. (18)

$$V_r = \frac{\sum_{s=1}^{B'} Y'_s}{B'} - Y' \tag{18}$$

Step 4: Attraction Attraction towards food is evaluated as per Eq. (19) where Foo denotes the position of food.

$$G_r = Foo - Y' \tag{19}$$

Step 5: Distraction The distraction to the external enemy is evaluated as per Eq. (20), where ene shows the enemy's position.

$$X_r = ene + Y' \tag{20}$$

Step 6 Cumulative memory property The cumulative memory property B_i is defined in Eq. (21). The added property enhances the performance in terms of better convergence rate. Moreover, the property enhances the accuracy in attaining an optimal solution. The behavior of dragonflies associated with the combinations of mentioned six patterns. Two vectors namely $\Delta Y'$ step and position Y' is concerned here for the purpose of updating the dragonflies' position. The step vector $\Delta Y'$ is equivalent to velocity vector in Particle Swarm Optimization (PSO) algorithm [28–30] since the dragonfly is based on the concept of PSO algorithm. The step vector $\Delta Y'$ represents the direction movement of dragonflies, which is determined in Eq. (21), where q' denotes the separation weight, P_r refers the separation of r^{th} dragonfly, t' refers the alignment weight, K_r denotes the alignment of r^{th} dragonfly, v' denotes the cohesion weight, V_r refers the cohesion

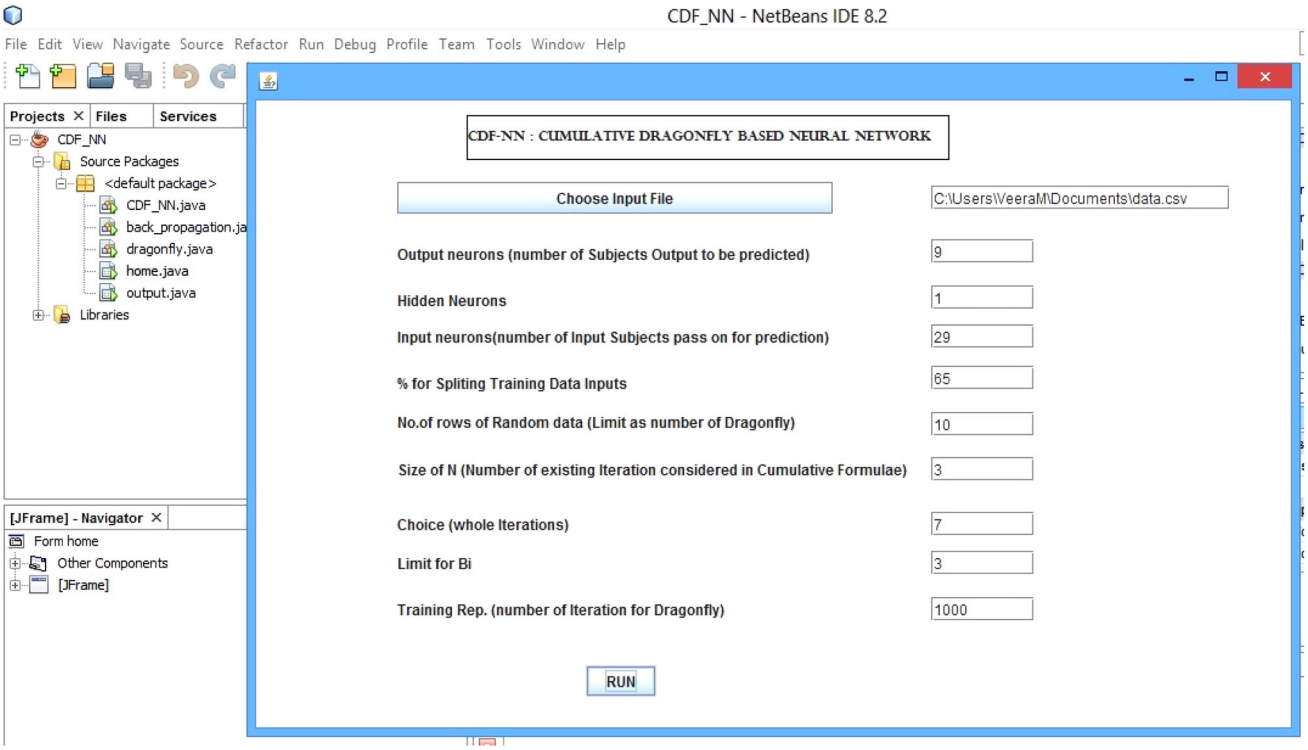

Fig. 6 Screenshot of CDF-NN implementation

of r^{th} dragonfly, u' is the food factor, G_r denotes the food source of r^{th} dragonfly, z' refers the enemy factor, X_r refers the position of enemy of r^{th} dragonfly, b denotes the constant, δdetermines the inertia weight and l refers the iteration counter.

$$\Delta Y'_{l+1} = \left(q'Z_r + t'K_r + v'V_r + u'G_r + z'X_r + bB_i\right) + \delta.\Delta Y'_l \tag{21}$$

where $B_i = \frac{(Y_l + Y_{l-1} + Y_{l-2})}{3}$

Various exploitative, as well as explorative behaviors of the dragonfly, can be determined while optimization using the above factors. The position vector is calculated as given in Eq. (22), where l refers the current iteration.

$$Y'_{l+1} = Y'_l + \Delta Y'_{l+1} \tag{22}$$

The dragonflies are assigned with great alignment and less cohesion while exploring the search space, and less alignment and great cohesion while exploiting the search space. In case if there are no neighborhood solutions, the dragonfly makes a random walk (levy flight), by the randomness, exploration and stochastic behavior are enhanced highly. In such case, the dragonflies' position gets updated with the aid of Eq. (23), where l denotes the current iteration, d denotes the dimensions of position vectors, y_1 and y_2 denotes the two random numbers in [0,1], $\wp$ refers to a constant value. As the iteration increases, the position, as well as the steps of every dragonfly, get updated using Eqs. (21, 22, and 23) respectively. For Y' and $\Delta Y'$ update, the neighbor of every dragonfly is defined by evaluating the Euclidean distance among entire dragonflies and choosing B' of them. The position update progresses iteratively till the last criterion gets satisfied.

$$Y'_{l+1} = Y'_l + levy\,(d) \times Y'_l \tag{23}$$

$$levy\,(d) = 0.01 \times \frac{y_1 \times \Phi}{|y_2|^{\frac{1}{d}}} \tag{24}$$

$$\Phi = \left(\frac{\Psi\,(1+\wp) \times \sin\left(\frac{\pi\wp}{2}\right)}{\psi\left(\frac{1+\wp}{2}\right) \times \wp \times 2^{\left(\frac{\wp-1}{2}\right)}}\right) \tag{25}$$

$$\psi\,(x) = (x-1)! \tag{26}$$

The pseudo code of proposed DA is illustrated in Algorithm 2. Here the input is the weightW_i. Very first the population $Y'_s : s = 1, \cdots B'$ gets initialized. Then the step vector$\Delta Y'$ is initialized. In the case of unsatisfied condition, some evaluations must be done as given in Fig. 6. If the individual (dragonfly) has one neighboring dragonfly, the velocity and position vector gets updated. Then at the end, the new position is viewed. The algorithm results

in the optimal weight W^*, which is the attained new position.

Algorithm 2: Cumulative DA

	Algorithm: Cumulative DA
1	**Input:** Weight W_i, where $i = 1,2,\ldots N$, N indicates the number of samples.
2	**Output:** Optimized weight W^*
3	**Parameters:** $W_i \rightarrow$weight, $W^* \rightarrow$optimized weight, $Y'_s \rightarrow$population, $\Delta Y' \rightarrow$step vector, $q', t', v', u', z' \rightarrow$separation weight, alignment weight, cohesion weight, food factor, enemy factor, $Z_r, K_r, V_r, G_r, X_r \rightarrow$ Separation, alignment, cohesion, the food source of r^{th} dragonfly, the position of the enemy of r^{th} dragonfly, $b \rightarrow$constant.
4	**Begin**
5	Initialize the input population $Y'_s : s = 1,\ldots..B'$
6	Initialize the step vector $\Delta Y'$
7	**while**
8	the end condition is not satisfied
9	Evaluate the objective value of each dragonfly
10	Food source update and enemy update
11	Update q', t', v', u', z' and the cumulative memory property constant b
12	Evaluate the primitive behavior of all Z_r, K_r, V_r, G_r, X_r using Eq. (17)to Eq.(21)
13	Neighboring radius update
14	**If**
15	an individual (dragonfly) has one neighboring dragonfly
16	Velocity vector value update using Eq. (22)
17	position vector value update using Eq. (23)
18	**Else**
19	position vector value update using Eq. (24)
20	**End if**
21	View the new position
22	**End while**
23	**End**

4 Results and discussion

The results of developed students' performance prediction model are discussed and compared to different features.

4.1 Simulation configuration

4.1.1 Experimental setup

The proposed CDF-NN model is implemented using Java as a software platform. Moreover, the conventional methods, like Dragonfly, Back prorogation, and Cumulative Dragonfly is implemented in java with FNN learning pattern and prediction. The Map-Reduce framework is constructed with ten mapper functions and ten reducer functions.

4.1.2 Metrics employed

Average error (MSE) If $\hat{V}'$ is a vector of $\hat{n}$ prediction, and V' is the vector of observed values in terms of the inputs to the function, then MSE can be evaluated as defined below

$$MSE = \frac{1}{\hat{n}} \sum_{t'=1}^{\hat{n}} \left(\hat{V}'_{t'} - V'_{t'} \right)^2 \tag{27}$$

RMSE The RMSE of predicted values $\hat{p}_t$ for t time of a p_t regression's dependent variable is evaluated for d different predictions as

$$RMSE = \sqrt{\frac{\sum_{t=1}^{d} \left(\hat{p}_t - p_t \right)^2}{d}} \tag{28}$$

4.1.3 Performance and comparative methods

The performance of CDF-NN is measured using certain valuations like MSE and root mean square error (RMSE) results by comparing it with learning ANN algorithms like existing dragonfly-NN [1], back prorogation optimization [31] as training algorithm.

5 Experimental results

Metric concerned for generating best results is on the basis of minimum error produced while learning and predicting the output in NN with MSE and RMSE measured values. Input data is passed on to the prediction model then all variables parameters values are assigned for every prediction performance model, such as output neurons (as the number of subject's marks to be predicted), hidden neurons (required number of hidden neurons), input neurons (as the number of input subjects marks for prediction), percentage values (percentages of training data taken for learning), limit value (number of dragonfly), N size (number of existing iteration considered in cumulative formulae), Choice (as whole iterations limit), Bi—Limit value (from which iteration the cumulative formulae to be considered using previous learning cycle), Training reps (number of iteration for dragonfly for training and learning).

This resultant screenshot Fig. 7 is implementation outcome of developed CDF with back prorogation and actual Dragonfly algorithms. *CDF achieved least MSE results of 16.007% which best-fit outcome among all 24 different experimental outcomes as shown in four varied SETs of the graph with trend line—in that CDF-MSE best was 16.33%.* This 16.00% of CDF-MSE is attained by taken inputs values (as per Fig. 6) are 1hidden layers, 65% training data, 1000 training representation of iterations, with maximum of 10 dragonflies, ordinary input neurons size as 29 and predicted output neurons as 9.

5.1 Performance evaluation

This section explains the performance analysis of proposed CDF-NN over the conventional methods, like backpropagation with NN and dragonfly with NN respectively. The investigation is done by varying the training data set to 60, 70, 80, and 90%.

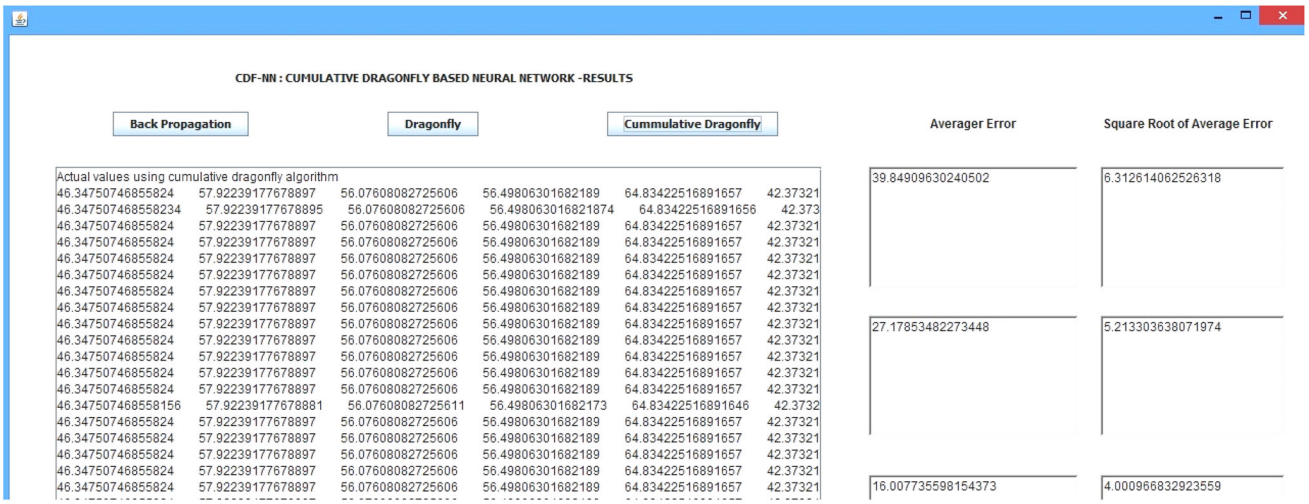

Fig. 7 CDF-NN application screenshot: MSE and RMSE:

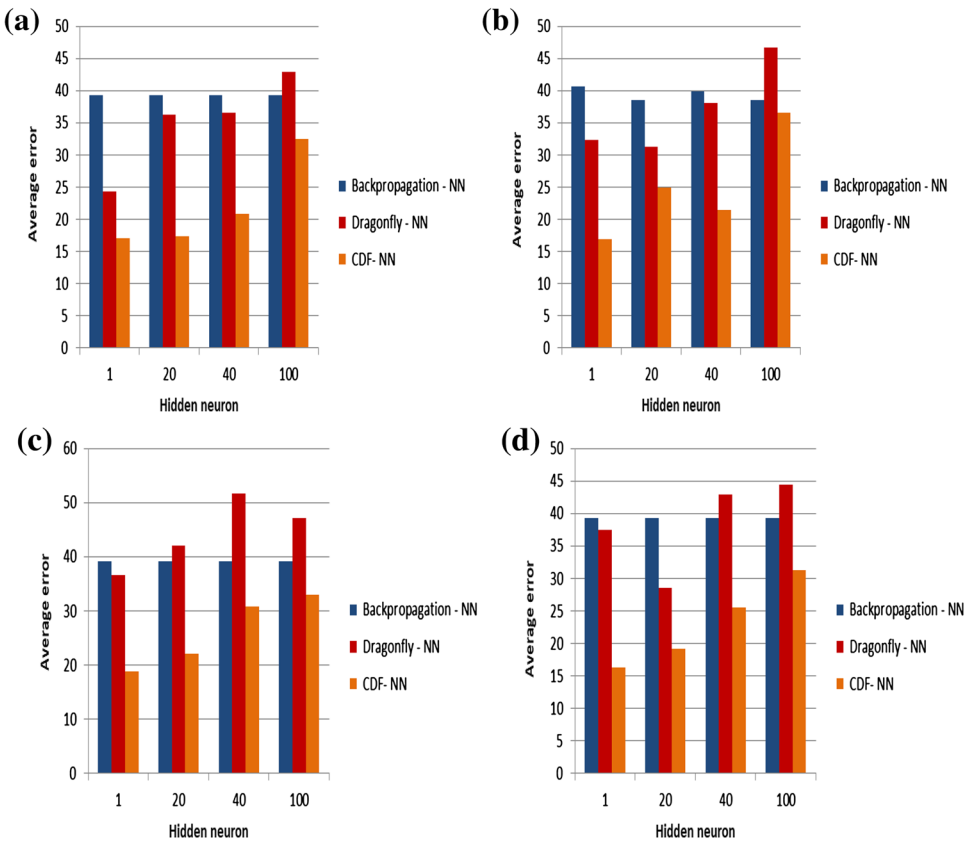

Fig. 8 Average error analysis of proposed model over conventional models by varying the data set (**a**) 60% of data set (**b**) 70% of data set (**c**) 80% of data set (**d**) 90% of data set

5.1.1 Analysis by varying training data set

Average error Figure 8 reviews the attained average error using the proposed CDF-NN, back propagation with NN and dragonfly with NN by varying the data sets to 60, 70, 80, and 90%. From the analysis, it is evident that, for hidden neuron 1, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 60% data set are 17.004, 24.268 and 39.216, respectively. For hidden neuron 20, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 60% data set are 17.335, 36.193 and 39.218. For hidden neuron 40, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 60% data set are 20.802, 36.598 and 39.218, respectively. For hidden neuron 100, the average error using CDF-NN,

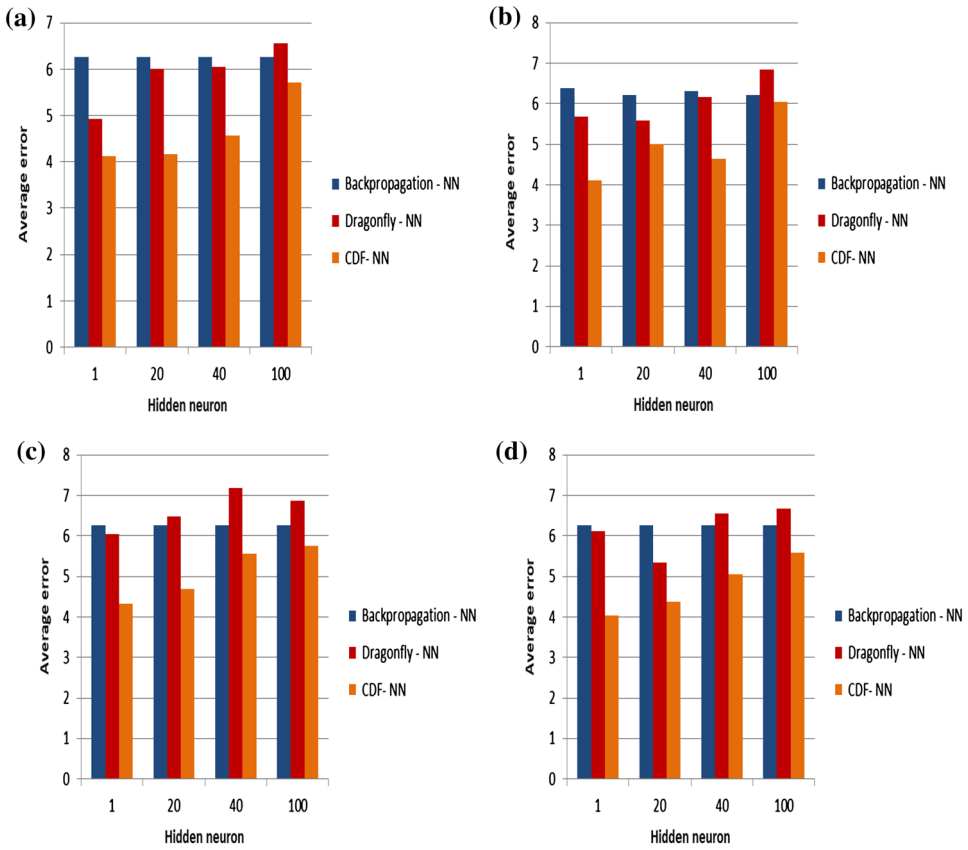

Fig. 9 RMSE analysis of proposed model over conventional models by varying the data set (**a**) 60% of data set (**b**) 70% of data set (**c**) 80% of data set (**d**) 90% of data set

dragonfly with NN and backpropagation with NN for 60% data set are 32.530, 42.954 and 39.218, respectively.

Figure 8b shows the average error using CDF-NN, a dragonfly with NN and backpropagation with NN for 70% data set. From the figure, it is found that for the hidden neuron 1, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 70% data set are 16.874, 32.351 and 40.657, respectively. Moreover, for hidden neuron 20, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 70% data set are 24.970, 31.203 and 38.518, respectively. For hidden neuron 40, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 70% data set are 21.426, 38.017 and 39.930, respectively. For hidden neuron 100, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 70% data set are 36.606, 46.651 and 38.446, respectively.

The average error using CDF-NN, dragonfly with NN and backpropagation with NN for 80% data set is illustrated in Figure 8c. From the figure, it is found that for hidden neuron 1, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 80% data set are 18.774, 36.538 and 39.214, respectively. For hidden neuron 20, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 80% data set are 22.004, 42.055 and 39.218, respectively. For hidden neuron 40, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 80% data set are 30.856, 51.617 and 39.218, respectively. For hidden neuron 100, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 80% data set are 33.057, 47.093 and 39.218, respectively.

The average error using CDF-NN, dragonfly with NN and backpropagation with NN for 90% data set is illustrated in Figure 8d. From the figure, it is observed that for hidden neuron 1, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 90% data set are 16.329, 37.450 and 39.214, respectively. For hidden neuron 20, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 90% data set are 19.167, 28.501 and 39.218, respectively. For hidden neuron 40, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 90% data set are 25.446, 42.957 and 39.218, respectively. For hidden neuron 100, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 90% data set are 31.296, 44.496, and 39.218, respectively. The efficiency of proposed CDF-NN is proved over other methods in terms of minimum average error.

RMSE The RMSE of proposed CDF-NN and other methods, like a dragonfly with NN and backpropagation with NN by varying the data set to 60, 70, 80, and 90% is illustrated in Fig. 9. Figure 9a shows the attained RMSE using CDF-NN, a dragonfly with NN and backpropagation with NN for 60%

data set. It is observed that for hidden neuron 1, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 60% data set are 4.123, 4.926, and 6.262, respectively. For hidden neuron 20, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 60% data set are 4.163, 6.016 and 6.262, respectively. For hidden neuron 40, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 60% data set are 4.560, 6.049, and 6.262, respectively. For hidden neuron 100, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 60% data set are 5.703, 6.553 and 6.262, respectively.

Figure 9b shows the attained RMSE using CDF-NN, a dragonfly with NN and backpropagation with NN for 70% data set. It is observed that for hidden neuron 1, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 70% data set are 4.107, 5.687 and 6.376, respectively. For hidden neuron 20, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 70% data set are 4.997, 5.586 and 6.206, respectively. For hidden neuron 40, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 70% data set are 4.628, 6.165 and 6.319, respectively. For hidden neuron 100, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 70% data set are 6.050, 6.830 and 6.200, respectively.

The attained RMSE using CDF-NN, a dragonfly with NN and backpropagation with NN for 80% data set is illustrated in Fig. 9c. It is reviewed that for hidden neuron 1, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 80% data set are 4.332, 6.044, and 6.262, respectively. For hidden neuron 20, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 80% data set are 4.690, 6.484 and 6.262, respectively. For hidden neuron 40, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 80% data set are 5.554, 7.184 and 6.262, respectively. For hidden neuron 100, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 80% data set are 5.749, 6.862 and 6.262, respectively.

The attained RMSE using CDF-NN, a dragonfly with NN and backpropagation with NN for 90% data set is illustrated in Fig. 9d. It is evident that for hidden neuron 1, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 90% data set are 4.040, 6.119and 6.262 respectively. For hidden neuron 20, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 90% data set are 4.378, 5.338 and 6.262, respectively. For hidden neuron 40, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 90% data set are 5.044, 6.554 and 6.262, respectively. For hidden neuron 100, the average error using CDF-NN, dragonfly with NN and backpropagation with NN for 90% data set are 5.594, 6.670 and 6.262, respectively. From the analysis, it is observed that the proposed model has minimum RMSE, which shows the superiority of the model over other methods.

5.1.2 Analysis by varying Bi

Further, the analysis of average error and RMSE is made by varying the limit of B_i value to 2, 4, 5 and 6. Fig. 10a illustrates the average error of proposed over conventional methods. It is observed that the average error using CDF-NN, a dragonfly with NN and backpropagation with NN for B_i value 2 are 21.767, 28.914 and 39.216, respectively. For B_i value 4, the average error using CDF-NN, dragonfly with NN and backpropagation with NN are 18.771, 33.008 and 39.218, respectively. For B_i value 5, the average error using CDF-NN, dragonfly with NN and backpropagation with NN are 23.254, 40.762 and 39.218, respectively. For B_i value 6, the average error using CDF-NN, dragonfly with NN and backpropagation with NN are 29.284, 49.540 and 39.218, respectively. Thus, it is observed that the average error of proposed CDF-NN is very low, which shows the enhancement of model over other methods.

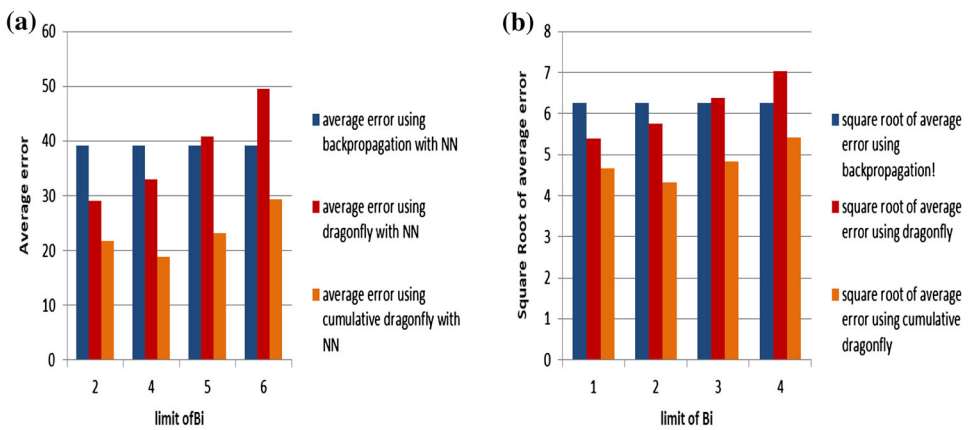

Fig. 10 Analysis of proposed CDF-NN method over existing methods by varying B_i (**a**) Average error. (**b**) RMSE

Figure 10b illustrates the RMSE of proposed over conventional methods by varying the limit of B_i value to 2, 4, 5 and 6. It is found that the RMSE value using CDF-NN, a dragonfly with NN and backpropagation with NN for B_i value 2 are 4.665, 5.377 and 6.262, respectively. For B_i value 4, the average error using CDF-NN, dragonfly with NN and backpropagation with NN are 4.332, 5.745 and 6.262, respectively. For B_i value 5, the average error using CDF-NN, dragonfly with NN and backpropagation with NN are 4.822, 6.384and 6.262, respectively. For B_i value 6, the average error using CDF-NN, dragonfly with NN and backpropagation with NN are 5.411, 7.038 and 6.262, respectively. Thus, it is observed that by varying the limit of B_i value, the proposed method attained minimum average error and RMSE, which proves the efficiency of proposed CDF-NN.

5.2 Analysis by varying epoch

Similarly, the analysis of CDF-NN is made in terms of average error and RMSE by varying the epochs to 500, 1000, 2500 and 5000, which is illustrated in Fig. 11. Figure 11a shows the average error obtained by proposed method over other methods by varying epochs. It is observed that for epoch 500, the average error using CDF-NN, dragonfly with NN and backpropagation with NN are 16.944, 25.980 and 39.214, respectively. For epoch 1000, the average error using CDF-NN, dragonfly with NN and backpropagation with NN are 26.593, 38.852 and 39.218, respectively. For epoch 2500, the average error using CDF-NN, dragonfly with NN and backpropagation with NN are 30.670, 39.290 and 39.218 respectively. For epoch 5000, the average error using CDF-NN, dragonfly with NN and backpropagation with NN are 24.699, 47.652 and 39.218, respectively.

Figure 11b shows the RMSE of CDF-NN and other methods by varying the epochs to 500, 1000, 2500 and 5000. From Fig. 11b, it is observed that the RMSE using CDF-NN, a dragonfly with NN and backpropagation with NN for epoch 500 are 4.116, 5.097 and 6.262, respectively. For epoch 1000, RMSE of CDF-NN, a dragonfly with NN and backpropagation with NN are 5.156, 6.233, and 6.262, respectively. For epoch 2500, RMSE of CDF-NN, a dragonfly with NN and backpropagation with NN are 5.538, 6.268 and 6.262, respectively. For epoch 5000, RMSE of CDF-NN, a dragonfly with NN and backpropagation with NN are 4.969, 6.903 and 6.262, respectively. From the analysis, it is evident that both the average error and RMSE are very less for the proposed CDF-NN when compared to other methods, which determines the efficacy of the developed method with the better optimal prediction of students' performance.

5.3 Discussion

The overall performance of proposed model over other conventional methods like Back propagation-NN, Dragonfly-NN and CDF-NN are tabulated in Table 1. The analysis is made in terms of two cases: best case and means case. From the Table 1, it is observed that MSE of proposed CDF-NN for best case is 30.17 and 55.92% better than Dragonfly-NN and back propagation-NN, respectively. The mean case of proposed CDF-NN is 37.54 and 38.79% superior to Dragonfly-NN and back propagation-NN, respectively. For RMSE, the proposed model for the best case is 69.83 and 62.03% better from Dragonfly-NN and back propagation-NN, respectively, and for the worst case, the proposed model is 21.19 and 22.36% better than the models like Dragonfly-NN and back propagation-NN, respectively. Hence the superiority of the proposed model is proved over other methods.

Table 1 Overall comparative analysis of proposed and conventional models

Methods	MSE		RMSE	
	Best	Mean	Best	Mean
Back propagation-NN	38.446	39.246	6.200	6.264
Dragonfly-NN	24.268	38.456	4.926	6.171
CDF-NN	16.944	24.019	4.665	4.863

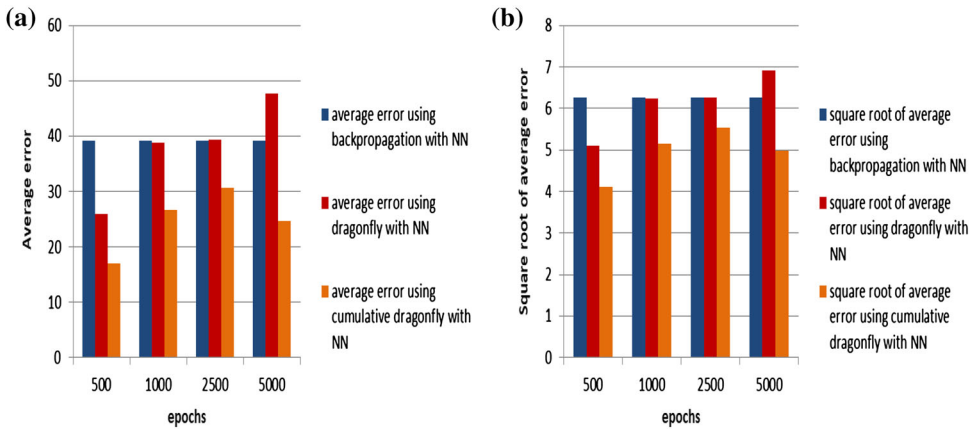

Fig. 11 Analysis of proposed CDF-NN method over existing methods by varying epochs. (**a**) Average error. (**b**) RMSE

6 Conclusion

This paper presents the student performance prediction model by proposing the Map-reduce architecture based CDF-NN. Initially, the marks of the students from semester 1 to semester 7 are collected from different colleges. Then, this information is provided to the map-reduce framework. In the training phase, the features are selected from the student's information and the intermediate data is generated by the map function. Then, the intermediate data is provided to the reducer function which is built with the CDF-NN for producing the trained model. At the testing phase, the information of the students is provided to the map function which is built with the trained CDF-NN and the intermediate data is generated. Then, the intermediate data is provided to the reducer function. The reducer function provides the estimated marks of the students in semester eight. The performance of the proposed prediction model is evaluated and the implementation is done using Java. The proposed method is compared with the existing methods, such as Dragonfly- NN and Back prorogation algorithm for the evaluation metrics, MSE and RMSE. The proposed prediction model obtains the MSE of 16.944 and RMSE of 4.665 for the best case. The future enhancement of this work is to use hybrid optimization algorithms and deep learning to further improve the prediction performance.

References

1. Mirjalili, S.: Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput. Appl. **27**(4), 1053–1073 (2016)
2. Kotsiantis, S.B.: Use of machine learning techniques for educational proposes: a decision support system for forecasting students grades. Artif. Intell. Rev. **37**(4), 331–344 (2012)
3. Minaei-Bidgoli, B., Punch, W.F.: Using genetic algorithms for data mining optimization in an educational web-based system. In: Proceedings of Genetic and Evolutionary Computation Conference, pp. 2252–2263, Springer, Berlin, 2003
4. Wolff, A., Zdrahal, Z., Herrmannova, D., .Knoth, P.P.: Predicting student performance from combined data sources. In: Educational Data Mining, Studies in Computational Intelligence, vol. 524, pp. 175– 202, Springer, Cham, 2014
5. Guarín, C.E.L., Guzman, E.L., González, F.A.: A model to predict low academic performance at a specific enrollment using DATA mining. IEEE J. Learn. Technol. **10**(3), 119–125 (2015)
6. García,E.P.I., Mora P.M.: Model prediction of academic performance for first-year students. In: Proceedings of International Conference on Artificial Intelligence, pp. 169-174, Puebla, Mexico, 2011
7. Touron, J.: The determination of factors related to academic achievement in the university: implications for the selection and counseling of students. High. Educ. **12**(4), 399–410 (1983)
8. Lassibilille, G., Gomez, L.N.: Why do higher education students dropout? Evidence from Spain. Educ. Econ. **16**(1), 89–105 (2008)
9. Chen, J.F., Hsieh, H.N., Do, Q.H.: Predicting student academic performance: a comparison of two meta-heuristic algorithms inspired by cuckoo birds for training neural networks. J. Algorithms **7**(4), 538–553 (2014)
10. Reynolds C.W.: Flocks, herds, and schools: a distributed behavioral model. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, pp. 25–34, ACM, New York, USA, 1987
11. Ramanathan, L., Geetha, A., Khalid, M., Swarnalatha, P.: Angelina Geetha, Khalid, M., Swarnalatha, P.: Student performance prediction model based on lion-wolf neural network. Int. J. Intell Eng. Syst. **10**(1), 114–123 (2017)
12. Malvandi, S., Farahi, A.: Provide a method for increasing the efficiency of learning management systems using educational data mining. Indian J. Sci. Technol. **8**(28), 1–10 (2015)
13. Ibrahim, Z., Rusli, D.: Predicting students' academic performance: comparing artificial neural network, decision tree and linear regression. In: 21st Annual SAS Malaysia Forum, Shangri-La Hotel, Kuala Lumpur, 2007
14. Ibrahim, Z., Rusli, N.M., Janor, R.M.: Predicting students' academic achievement: comparison between logistic regression, artificial neural network, and neuro-fuzzy. In: Proceedings of the International Symposium on Information Technology, Kuala Lumpur, Malaysia, 2008
15. Bhatnagar, K., Gupta, S.C.: Investigating and modeling the effect of laser intensity and nonlinear regime of the fiber on the optical link. J. Opt. Commun. **38**(3), 341–353 (2017)
16. Yang, X.-S.: Nature-Inspired Meta Heuristic Algorithms, 2nd edn. Luniver Press, Frome (2010)
17. Cui, Z., Shi, Z.: Boid particle swarm optimization. J. Int. J. Innov. Comput. Appl. **2**(2), 77–85 (2009)
18. Arsad, P.M., Buniyamin,N., Manan, J.A.: A neural network students' performance prediction model (NNSPPM). In: Proceedings of IEEE International Conference on Smart Instrumentation, Measurement and Applications, Kuala Lumpur, Malaysia, 2013
19. Avoyan, H.: Machine Learning Is Creating A Demand For New Skills. www.forbes.com/sites/forbestechcouncil/2017/06/26/machine-learning-is-creating-a-demand-for-new-skills/#325004dc7ae2
20. Mantri, R., Jewalikar, A.: Implementation and performance analysis of academic-MapReduce algorithm (AcdMR). Int. J. Comput. Appl. **121**(19), 17–20 (2015)
21. Tajunisha, N., Anjali, M.: Predicting student performance using MapReduce. Int. J. Eng. Comput. Sci. **4**(1), 9971–9976 (2015)
22. Romero, C., López, M.I., Luna, J.M., Ventura, S.: Predicting students' final performance from participation in on-line discussion forums. Comput. Educ. **68**, 359–472 (2013)
23. Asogwa, O.C., Oladugba, A.V.: Of students academic performance rates using artificial neural networks (ANNs). Am. J. Appl. Math. Stat. **3**(4), 151–155 (2015)
24. Tung-Kuang Wu, Shian-Chang Huang, Hsiu-Ting Kao, Hsu Chang, and Ying-Ru Meng, "A MapReduce Implementation of the Genetic-Based ANN Classifier for Diagnosing Students with Learning Disabilities," In Proceedings of the Seventh International Conference on Advanced Engineering Computing and Applications in Sciences, pp. 30–35, Barcelona, Spain, 2013
25. Montana, D.J., Davis, L.: Training feedforward neural networks using genetic algorithms. In: Proceedings of the 11th international joint conference on Artificial intelligence, vol. 1, pp. 762–767, Detroit, Michigan, 1989
26. Brajevic,I., Tuba, M.: Training feed-forward neural networks using firefly algorithm. In: Proceedings of the 12th International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, pp. 156-161, 2013
27. Mirjalili, S., Hashim, S.Z.M., Sardroudi, H.M.: Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. Appl. Math. Comput. **218**(22), 11125–11137 (2012)

28. Kadrovach, B.A., Lamont, G.B.: A particle swarm model for swarm-based networked sensor systems. In: Proceedings of the 2002 ACM Symposium on Applied Computing, pp. 918–924, Madrid, Spain, 2002
29. Cui, Z.: Alignment particle swarm optimization. In: Proceedings of 8th IEEE international conference Cognitive Informatics, pp. 497–501, Kowloon, Hong Kong, China, 2009
30. Pires, E.S., Machado, J.T., de Moura Oliveira, P.B., Cunha, J.B., Mendes, L.: Particle swarm optimization with fractional-order velocity. Nonlinear Dyn. **61**(1–2), 295–301 (2010)
31. Wu, K., Zhong, Y., Wang, X., Sun, W.: A novel approach to subpixel land-cover change detection based on a supervised back-propagation neural network for remotely sensed images with different resolutions. IEEE Geosci. Remote Sens. Lett. **14**(10), 1750–1754 (2017)

M. R. M. VeeraManickam is currently working as an Assistant Professor in Dept. of Information Technology, Trinity College of Engineering and Research, affiliated to SavitribaiPhulePuneUniversity (SPPU), Pune. And also as part-time Research Scholar in Dept. of Computer Science and Engineering at Karpagam Academy of Higher Education, Karpagam University, and Coimbatore, India. He received his B.Tech. degree in Information Technology from LVEC, Anna University, Chennai, India, in 2006, and M.Tech.degree in Information Technology from Sathyabama University, Chennai, India, in 2011.He is working on the Funding Research Project entitled as "Classroom Note's sharing using Smart E-learning& Internet of Things" under SPPU BCUD Research Grant Scheme Ay: 2016-18 of ?75k funding. His main research work focuses on E-learning, Social Network, Internet of Things and Artificial Neural Network.

M. Mohanapriya is working as Professor & Head of Dept.-CSE at Karpagam University. She received his B.E. degree in Computer Science Engg. From SKEC, Bharathiar University, Coimbatore, India, in 2002, and M. E degree in Computer Science Engg. From CEC, Anna University, Chennai, India, in 2004, And Received her Doctor of Philosophy from Anna University, Chennai, India in Department of Information & Communication Engineering. She is working as REVIEWER FOR JOURNAL MANUSCRIPTS for following journal IEEE Transactions on Mobile Computing, Computers and Electrical Engineering, Elsevier Publications, Arabian Journal of Science and Engineering, Springer

Bishwajeet K. Pandey is a co-founder of Gyancity Research Lab Private Limited. Every year, Gyancity Research Lab organizes three international conferences (ICGCET.ORG, RTCSE.ORG, and CICN.IN) across the globe. Now, he is associated with GSSI, Italy. He has worked as Asst. Professor in Department of Research at Chitkara University, Junior Research Fellow (JRF) at South Asian University and Lecturer at Indira Gandhi National Open University (IGNOU). He has completed Master of Technology (IIIT Gwalior), Master of Computer Application (IGNOU), R&D Project in CDAC-Noida. He is a Life Member (CSI), Professional Member (IEEE) and Associate Member (IEI). He has authored and coauthored over 150+ papers (11 ESCI/SCIE, 75 IEEE Explore, 108 Scopus) with 152 co-authors from 68 University/Institution/Lab across the globe. His paper has been presented in every corner of the globe like Vietnam, India, Indonesia, Sri Lanka, Singapore, Pakistan, Hong Kong, Korea, Australia, Russia, Croatia, Cyprus, Ireland, Lebanon, Malaysia, France, Denmark, Spain, China, USA and UAE. He has filled 2 patents in Patent Office in India and also authored 5 books available for sale on Amazon.

Sushma Akhade is currently working as an Assistant Professor in Dept. of Computer Science and Engineering, KJ college of Engineering and Research, Pune, India. she received her B.E. and M.E. in Computer Science and Engineering. Her research interest in computer networking.

S. A. Kale is working as Dean, Research at Trinity College of Engineering and Research, Pune affiliated to Savitribai Phule University Pune (Previous Pune University), India from 2010. He has completed his Ph.D. and Master's Degree from Pune University. He has more than 17 years industrial, academic and research experience. He has published more than fifty research papers in various conferences & journals and filed eleven patents. He has worked as Guest Editor for publishing many journal special issues. He has worked as an Editor for two edited book "Renewable Energy and Sustainable Development" and "Renewable Energy Systems" published by Nova Science Publishers, USA. He is an author of a first book published on Multi-rotor Wind Turbine. He has worked as Conference Chair for four international con-

ferences and one national conference. He is working as Editorial Board member and reviewer for many journals and Conference Committee member for ASME and others. He has received Eminent Researcher Award and Outstanding Faculty in Engineering Award in 2017. His basic research areas are wind and solar energy, renewable energy and composite materials.

Reshma Patil She received B.E. in computer science engineering from shivaji University, M.E. In computer engineering from SPPU. Her research interest in Area Wsn And ANN. And workingas kjcoemr as Assistant Professor.

M. Vigneshwar Assistant Professor, Department of Computer Science and Engineering, Faculty of Engineering, Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India. He pursued his Bachelor of Engineering in Computer Science and Engineering from Bannari Amman Institute of Technology, Sathyamangalam and Masters of Engineering from SriKrishna College of Engineering and Technology, Coimbatore. He has around 15 years of Experience spanning around 8 years in Industry & Research and 7 years in Academic. He has won four Awards from Computer Society of India. He has published over 25 International Conference and 64 National Conference Publications. He has 13 International Journal Publications. He has served as Technical Programme Committee Member & Reviewer for various International Conferences and has done numerous funded programs, workshops, Seminars and Conferences. He has been an active member and Board Members of various other Societies and Professional Bodies. His areas of Interests are Medical Imaging, Computational Biology, Computational Neurosciences, Artificial Intelligence, Machine Learning and Computational Sciences relating to Applied Research.