International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

A Novel PMFFC -Based Software Effort Estimation Using FMGKF-DENN Algorithm

***[1]K. Harish Kumar and [2]K. Srinivas**

Submitted: 10/09/2023 **Revised**: 22/10/2023 **Accepted**: 09/11/2023

Abstract: Software Effort Estimation (SEE) is getting more concerned owing to the software industry's development. An increase in the deadline along with the budget of the project was led by an incorrect estimation. This may fail the project in turn. Using the Fisher and Mish Gaussian Kernel Function based Deep Elman Neural Network (FMGKF-DENN) algorithm, a novel Pearson and Mahalanobis-centered Farthest First Clustering (PMFFC)-centered SEE was proposed in this study. Primarily, through determining specific factors like scope, objectives, Infrastructure, and characteristics, the details regarding the provided historical projects are obtained. For efficient task completion, the projects are grouped utilizing the PMFFC algorithm grounded on the details gathered. Next, the classes split the code for those specific types of projects. After that, the input data is pre-processed. Certain features are retrieved as of the pre-processed data and the Controlled Sea Turtle Foraging Optimization (CSTFO) approach chooses the crucial features. The deviation value is deemed as a target for efficient SEE in the FMGKF-DENN, where the selected features are fed. The experiential outcomes illustrated that the SEE process was executed more accurately by the proposed framework along with outperforming various top-notch models.

Keywords: *Software, Effort estimation, project management, feature extraction, Sea Turtle Foraging Optimization (STFO), Neural Network (NN), Deviation.*

1. Introduction

Recently, conventional software development methods were replaced by the software development process, which has become more prominent in industries. Since software includes various applications in the business, banking, medical field, aerospace, defense, and science and technology, it is one amongst the significant inventions that affect human society [1]. It also contains an array of industrial utilities along with several security systems. One of the top ten vital success elements in software projects is Reliable estimation, as per the Standish Group [2]. To estimate the amount of labor or the number of hours needed for project completion, SEE approaches are employed. SEE is often described as man-hours or man-months [3, 4].

Generalizing as of a few old projects is requisite while performing Effort Estimation (EE). Generalization from these types of limited knowledge occurs naturally in open circumstances [5]. Underestimating the SEE can result in budget and schedule overruns while overestimating it can cause project loss [6].

Diverse SEE variants were presented by researchers. However, for precise estimation, the best method could not be developed yet. Machine learning, algorithmic models, and expert judgment are the most recent SEE approaches. An effort was requisite for conventional expert judgment approaches to document activities. Thus the estimation is made more complicated and time-consuming [7, 8]. Grounded on analogy and deduction, the software estimation is processed by the utilization of algorithmic techniques grounded on mathematical equations and machine learning [9, 10]. Using the FMGKF-DENN algorithm, a novel PMFFC - based SEE is developed in this paper to tackle these limitations in prevailing models.

[1]Research Scholar, Department of Computer Science & Engineering, Koneru Lakshmaiah Education Foundation, Deemed to be University, Hyderabad, Telangana, India,500075 and Assisstant Professor,Department of Computer Science&Informatics.,MahatmaGandhi University,Nalgonda,Telangna,India 508001. [1]khrsharma@gmail.com
[2]Professor, Department of Computer Science & Engineering, Koneru Lakshmaiah Education Foundation, Deemed to be University, Hyderabad, Telangana, India [2]sriracw9@klh.edu.in

1.1 Problem Definition

Certain limitation exists even though many approaches were utilized for efficient SEE. The limitations are enlisted as follows;

- The main issue with SEE is the lack of project data, incorporating the utilized element's size and quantitative counts, the number of analysis classes, the number of relationship sorts, the number of connected attributes, along with the use-case points
- Centered on Line of Code (LoC), prevailing research approaches categorized the programs. The grouping was incorrect if the LoC is identical for a diverse type of code
- During the SEE, deviation may occur.
- The major issue in SEE is the size and inefficiency.

Hence, a novel SEE algorithm is proposed here to alleviate the issues mentioned above.

This paper's remaining part is structured as follows: Section 2 shows the literature survey, Section 3 explains the proposed methodology, Section 4 illustrates the results and discussion, and finally, Section 5 concludes the paper with future work.

2. Literature Survey

Passakorn Phannachitta [11] exaggerated an analogy-centered SEE scheme to determine the effort requisite for a new software project grounded on the entire effort utilized in the earlier similar project's completion. Regarding accuracy and reliability, the presented approach's superiority was revealed by the experiential outcomes. However, the presented approach's major downside was the hyper-parameter optimization needs.

Tirimula Rao*et al.* [12] suggested a Differential evolution algorithm for Analogy-centered software development effort Estimation (DABE). Complicated multimodal optimization issues were efficiently solved by the presented approach. Nevertheless, the major criterion to be pondered was the control parameter.

Tianpei Xia *et al.* [13] framed Rapid Optimizing Methods for Estimation (ROME) where Sequential Model-centered Optimization (SMO) was employed for EE. However, for both classic waterfall and contemporary projects, better performance was attained. The wastage was elevated by resources that were allocated without appropriate estimation.

Leandro L. Minku [14] investigated an online supervised hyperparameter tuning process grounded on SEE. A poor Cross Company (CC) split was successfully avoided by the presented clustering models. Contrarily, the estimation performance was affected by the CC split's random selection.

Jose Thiago H *et al.* [15] propounded the Heterogeneous and Dynamic Ensemble Selection (HDES) model made of a set of regressors chosen dynamically by classifiers for SEE. The adapted regression algorithms avoided the overfitting issue. However, non-functional characteristics along with quality factors of that project were not estimated by this technique.

Frank Vijay [16] developed a fuzzy-centered hybrid technique for SEE. The fuzzy logic and defuzzification via the weighted average model controlled the uncertainty in the software size. Regarding Median Mean Relative Error (MMRE) and Variance Account For (VAF), better performance was acquired. Nevertheless, the incorporation of both the functional and non-functional characteristics in a single framework affected the feasibility.

Resmi *et al.* [17] introduced a fuzzy analogy with optimal firefly algorithm-grounded SEE. The '3' stages included here were Fuzzification, rule-centered fuzzy system, and defuzzification. Hence, the parameter's maximal likelihoods utilized in the presented approach were more precisely obtained. However, this mode didn't handle the missing and noisy data.

Anfal A. Fadhil *et al.* [18] established the SEE technique centered on the hybrid Dolphin and bat (Dolbat) algorithm. Better outcomes for EE were given by the measurement's outcomes. However, only fewer individuals were optimized by the presented dolphin algorithm.

Saurabh Bilgaiyan *et al.* [19] projected Chaos –centered morphological genetic algorithm for the SEE process. Mathematical morphology (MM) comprising of a hybrid-artificial neuron (Dilation Erosion Perceptron (DEP) extended as of the complete lattice theory (CLT) concept was the basis for this work. Thus, the method's worthiness was proved by the outcomes. Conversely, data overhead issues affected this model.

Saurabh Bilgaiyan *et al.* [20] presented Artificial Neural Network (ANN)-feed-forward back-propagation neural network and Elman neural network method for SEE. Eventually, better prediction accuracy was obtained by the presented model than the prevailing algorithms. However, the presented model's limitation was that better performance was not executed on the data obtained as of heterogeneous software development methods.

3. Proposed Novel Software Effort Estimation Technique

To deliver the product on time and within budget, EE is an extremely significant activity for planning and scheduling software project life cycle. So, using the FMGKF-DENN algorithm, a novel PMFFC-based SEE framework is proposed in this research. Figure 1 illustrates the proposed model's structural layout.

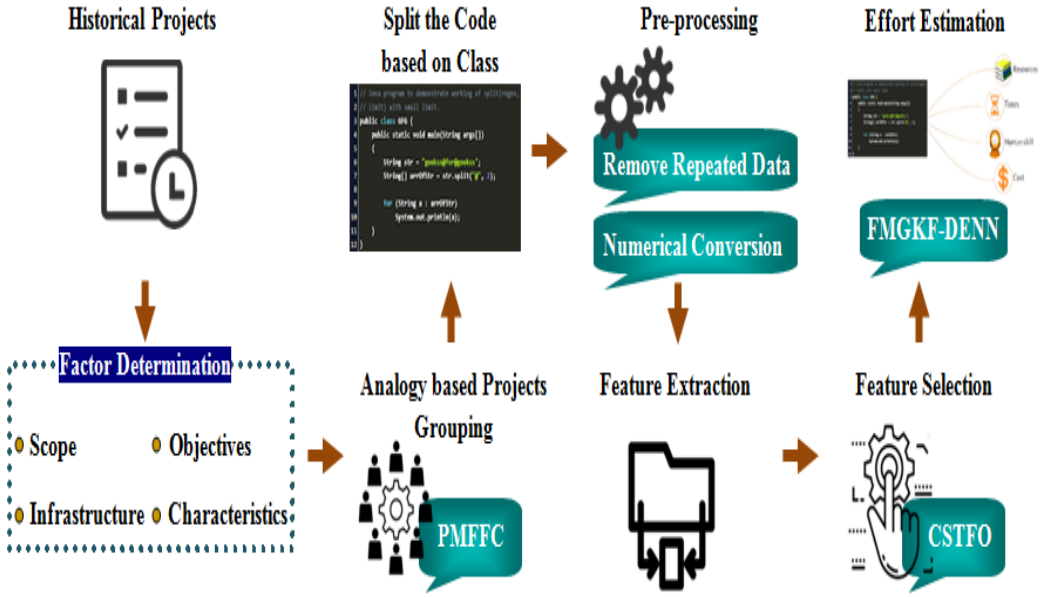

Fig 1: Structural design of the proposed model.

3.1 Factor Determination

The input pondered to start the SEE process in this work is the project's (hospitals, schools, restaurants, etc.) historical data. Only when a project is completed on schedule, within budget, and meets its objective, it is considered successful. It is preferable to determine these factors to make the SEE process more quick and reliable as the scopes, objectives, infrastructures, and other elements of each project may vary. The factors to be determined are detailed as follows.

- *Scope:* The project's intent in identifying and documenting a list of objectives, as well as its prompt delivery and deadlines are referred to as "scope". Hence, (S_f) signifies the scope factor determined.
- *Objectives:* The information regarding the project's outcome was provided by objectives, which are notated as (O_f).
- *Infrastructure:* It is the project's basic framework. Regarding the project undertaken, the infrastructure is indicated as (I_f).
- *Characteristics:* Via project characteristics (C_f), the project's distinguishing feature or quality was understood.

For the effective SEE, equation (1) gives the mathematical formulation of the factors determined (f^d) and is given as,

$$f^d = \{f^1, f^2, f^3, ..., f^n\} \quad (1)$$

Where, the factors determined for $n-$number of projects are interpreted as $d = 1,2,3,..., n$.

3.2 Analogy based projects grouping using PMFFC

Using PMFFC grounded on distance (analogy), the projects are classified after the determination of specific factors in this phase. The centroid points are selected randomly along with assigned to the projects in the cluster grounded on the maximum (Euclidean) distance as of the set of already-picked centers in the Farthest First Clustering (FFC) algorithm. Pearson correlation (δ) is generated into a conventional FFC by adding a Mahalanobis distance estimation named PM owing to the unreliability of this random selection criterion along with the sensitivity of Euclidean distance to noise or outliers. PMFFC is defined as the combination of PM and FFC. The steps involved in PMFFC are revealed further down,

1. Let the factors determined (f^d) be the input. Initially, the Pearson correlation (δ) betwixt (f^d) for each input project is determined as follows.

$$\delta = \frac{\sum\left(f^1 - \mu\left(f^1\right)\right)\left(f^2 - \mu\left(f^2\right)\right)}{\sqrt{\sum\left(f^1 - \mu\left(f^1\right)\right)^2 \sum\left(f^2 - \mu\left(f^2\right)\right)^2}} \quad (2)$$

In the above equations, the mean value of f^1 and f^2 are denoted as $\mu\left(f^1\right)$ and $\mu\left(f^2\right)$, and factors determined for projects 1 and 2 are determined as f^1 and f^2. The strongest correlation is determined as of the output obtained.

2. Now, the first centroid considered here is the strongest correlation. The next points are chosen grounded on the first centroid.

3. Since a large number of project data was employed here, M^d is used for distance calculation in PMFFC Mahalanobis distance. It is defined as,

$$M^d = \sqrt{\left(f^1 - f^2\right)^T \delta^{(-1)}\left(f^1 - f^2\right)} \quad (3)$$

4. Grounded on their farthest distance, the steps are repeated by computing the cluster centroids along with rearranging the data points grounded on the new cluster centroids. When every cluster centroid tends to converge, the procedure comes to an end. Hence, V_c represents the clustered projects. Here, $c = 1,2,\cdots,k$, that represents the number of clusters formed.

3.3 Split the code based on the class

Grounded on class, the codes are divided here for the grouped projects. To diminish the size and error that occurs during SEE, one of the defense strategies utilized is code splitting. Application's certain portions potentially dangerous are removed by this model of protection and transferred to a secure server, where they can operate in a trusted and secure environment. Thus, the splitted code $\left(\overline{\Im}_s\right)$ for each project is modeled as,

$$\overline{\Im}_s = \overline{\Im}_1, \overline{\Im}_2, ..., \overline{\Im}_K \quad (4)$$

Where, the K -number of splitted codes is represented as $s = 1,2,..., K$.

3.4 Pre-processing

Later, pre-processing is undergone by the splitted codes. A large number of repetitive data are contained since the historical data is pondered here. Therefore, to eliminate redundant or repeated data, Pre-processing is executed. Additionally, utilizing the numerical conversion process, the input data's string values are transformed into their corresponding number values, making it useful for further processing. The pre-processed data $\left(\Re^p\right)$ expression is denoted as

$$\Re^p = \left\{\Re^1, \Re^2, ..., \Re^n\right\} \quad (5)$$

Here, the n number of pre-processed splitted codes is referred to as $p = 1,2,..., n$.

3.5 Feature Extraction

From the pre-processed outcome, diverse sorts of features are extracted in this phase. The features extracted for effective estimation are the actual effort, the database system used, the development team's size, the application's experience level, the programming language used, the analyst's ability, the execution time constraints, the main storage constraints, the programmer's ability, etc. The extracted features $\left(E_f\right)$ are expressed as,

$$E_f = \left\{E_1, E_2, \cdots\cdots, E_N\right\} \quad (6)$$

Where, the number of features extracted as of the pre-processed splitted codes is determined as N . After that, for feature selection, these features are fed into the CSTFO.

3.6 Feature Selection using CSTFO

This stage involves using CSTFO to identify the optimal features needed for the SEE. A nature-inspired metaheuristic algorithm called Sea Turtle Feeding Optimisation (STFO) is grounded on the foraging habits of sea turtles for food. Typically, migration in sea turtles occurs in a straight line from one location to another. However, their capacity for exploration is constrained along with encountering local optimal issues when they go over the open sea. Therefore, the exploration control parameter is introduced into the traditional STFO method to alleviate the aforementioned problems and is known as CSTFO. The process is detailed as follows,

Step:1 Firstly, initialize the sea turtles population in the ocean as $\left\langle E_f = E_1, E_2, ..., E_N \right\rangle$ (extracted features) in d-dimensional search space. Let the food source be notated as $\wp$. Also, the N -number of sea turtle's position is initialized as

$$ST^l = ST^1, ST^2, ..., ST^d \quad (7)$$

Where, the d – number of the position for n – number of sea turtles is indicated as $l = 1,2,3,..., d$. Similarly, the food source's position $\left(\wp^p\right)$ is also initialized that is formulated as,

$$\wp^p = \wp^1, \wp^2, ..., \wp^z \quad (8)$$

Here, the z – number of food sources for sea turtles is signified as $p = 1,2,..., z$.

Step:2 Each sea turtle's velocity $\left(S_\upsilon\right)$ in the population is determined after initializing the sea turtle position in this step. Thus, the sea turtle velocity is expressed as,

$$S_\upsilon = S_1 + S_2 + ... + S_V, U^{\min} \le S_\upsilon \le L^{\max} \quad (9)$$

Where, the sea turtle population's velocity is proffered as $\upsilon = 1,2,..., V$, the upper and lower bound limit range of sea turtle velocity is notated as $U^{\max}$ and $L^{\max}$.

Step:3 Each sea turtle's fitness (deviation and estimation accuracy) $\left(\alpha^{ST}\right)$ in the population is evaluated in this step. The fittest sea turtle in the population is identified as the strongest turtle δ^{ST}. The fitness evaluation is modeled as follows,

$$\alpha^{ST} = \alpha^{ST}\left(E_f\right) \tag{10}$$

Further, equation (11) gives the expression for the strongest turtle selection

$$\delta^{ST} = \arg\max_f \left[\alpha^{ST}\right] \tag{11}$$

Step:4 Utilizing the expression (12), the ocean current velocity of every sea turtle $\left(\vartheta\left(ST^l\right)\right)$ is computed as the sea turtles travel in the open sea for searching food over long distances.

$$\vartheta\left(ST^l\right) = \vartheta\left[ST^1, ST^2, ..., ST^d\right] \tag{12}$$

Thus, using ocean current velocity $\left(S_\upsilon\left(T+1\right)\right)$, the updated sea turtle velocity is given as,

$$S_\upsilon(T+1) = S_\upsilon(T) + \vartheta\left(ST^l\right) + \left[\frac{\alpha^{ST}\left(E_f(T) - \alpha^{ST}\left(E_f(T-1)\right)\right)}{\alpha^{ST}\left(E_f(T-1)\right)}\right]\left(E_f(T) - E_f(T-1)\right) \tag{13}$$

In the above expression, the sea turtle's position at the time T is denoted as $E_f(T)$. The turtle's position at the time $T-1$ is notated as $E_f\left(T-1\right)$

Step:5 The sea turtles have seagrasses and phytoplankton as their food during their movement to long distances. DiMethyl Sulfide (DMS), which has a strong odour was released by the seagrasses, which in turn aids the sea turtle to identify their food readily. Grounded on the resemblance betwixt the sea turtle and the food source, the DMS odour's $\left(DMS^o\right)$ strength is evaluated. The mathematical formulation for the odour measurement is provided as,

$$DMS^o = \begin{cases} \alpha^{ST} > \wp, DMS^{\wp} = 0 \\ \alpha^{ST} < \wp, DMS^{\wp} = \dfrac{F\left(\wp^I\right)}{\sum_{p=1}^{z} F\left(\wp^p\right)} e^{\left[\frac{D_{\wp^I E^J}}{2\chi^2(T)}\right]} \end{cases} \tag{14}$$

Where, the strength of odour as of the food source is referred as $DMS^{\wp}$, the I^{th} food source fitness value is symbolized as $F\left(\wp^I\right)$, the parameter explaining the way the DMS spread is signified as $\chi^2(T)$ and $D_{\wp^I E^J}$ stands for the turtle E^J and the food source $\wp^I$ distance.

Step:6 By considering the maximum value of DMS^o, the best food source $Best\left(\wp^p\right)$ is now identified and is expressed as,

$$Best\left(\wp^p\right) = \arg\max\left[DMS^o\right] \tag{15}$$

Step:7 Thus, the sea turtle's position is updated $\left(E_f\left(T+1\right)\right)$ using (16)

$$E_f(T+1) = E_f(T) + \omega S_\upsilon(T+1) + Best\left(\wp^p\right)\left[\wp^p - E_f(T)\right] \tag{16}$$

Where, the control parameter is represented as ω, which is obtained using the following equation,

$$\omega = \left(G+2\right)\cdot H - 1 \tag{17}$$

Here, the constant diminishing linearly from 2 to 0 is denoted as G, and the random value ranging between $\left[0,1\right]$ is denoted as H.

Step:8 Until the optimal food source is recognized, the process is repeated. Thus, the number of features selected $\left(\overline{\overline{f_s}}\right)$ via CSTFO is modeled as,

$$\overline{\overline{f_s}} = \overline{\overline{f_1}}, \overline{\overline{f_2}}, ..., \overline{\overline{f_q}} \tag{18}$$

In the above equation, the number of optimal features selected is preferred as $s = 1,2,..., q$. The proposed CSTFO's pseudo code is displayed below.

Input: Extracted features $\left(E_f\right)$

Output: Selected features $\left(\left(\overline{\overline{f_s}}\right)\right)$

Begin

Initialize sea turtle population $\left(E_f\right)$

Initialize $ST^l = ST^1, ST^2, ..., ST^d$

Initialize food source position $\left(\wp^p\right)$

Obtain $\left(S_\upsilon\right)$

Evaluate fitness value $\alpha^{ST} = \alpha^{ST}\left(E_f\right)$

Determine $\delta^{ST} = \arg\max_f \left[\alpha^{ST}\right]$

Estimate ocean current velocity

Update $\left(S_\upsilon(T+1)\right)$

If $\alpha^{ST} > \wp$

then $DMS^{\wp} = 0$

Else if $\alpha^{ST} < \wp$

Calculate $DMS^{\wp} = \dfrac{F\left(\wp^I\right)}{\sum_{p=1}^{z} F\left(\wp^p\right)} e^{\left[\frac{D_{\wp^I E^J}}{2\chi^2(T)}\right]}$

End if

Compute best food source $Best\left(\wp^p\right)$

Update sea turtle position $\left(E_f(T+1)\right)$

Obtain $\left(\overline{\overline{f_s}}\right)$

End

3.7 Effort estimation using FMGKF-DENN

SEE occurs after the requisite optimal features have been chosen. A feed-forward network having an input layer, hidden layer, context layer, and output layer is called a "Deep Elmann Neural Network" (DENN). Here, the output is created by multiplying the incoming features by the activation function along with adding the bias values. The estimation error is maximized by the general activation function along with the utilization of random weight values and lengthens the task completion time. The traditional DENN is therefore modified to include the Fish and Mish Gaussian Kernel function, giving rise to the term FMGKF-DENN.

The EE methodology is briefed below.

- In the EE process, the target value $\left(tar^s\right)$ is pondered initially. Here, a target value is fixed for effective estimation as there exists a probability of deviating from the actual value and the estimation value.
- With X number of input layers $\left(I_X\right)$ and s number of output layers $\left(O_s\right)$, the DENN is provided. The number of hidden layers $\left(\hbar\right)$ and context layers $\left(c\right)$ ranges betwixt $X \le \hbar \le Y, X_2 \le c \le Y_2$. Thus, the input is multiplied by input-hidden weights at every node. The hidden layer output $\left(\hbar^o\right)$ is given by (19),

$$\hbar^o = \lambda\left(B_2 c^o + B_1 c^i\left(\overline{\overline{f_s}}\right)\right) \quad (19)$$

$$c^o = \hbar^o - 1 \quad (20)$$

Where, the context layer output is modeled as c^o, the context layer's input is signified as $c^i\left(\overline{\overline{f_s}}\right)$, Miss Gaussian Kernel activation function is proffered as λ, and is illustrated in (19), the input-hidden layer weight value and context-hidden layer weight value was symbolized as B_1, B_2

$$\lambda = (2\pi)^{-\varpi/2} e^{-\frac{1}{2}mn} \quad (21)$$

Here, ϖ denotes the free space dimension and constant values were indicated by mn .

- By taking the Fisher score value's mean, the DENN's weight initialization (B_i) is done and is signified as

$$B_i = \{B_1, B_2, B_3\} \text{x} \quad (22)$$

$$B_i = mean\left\{(sc^m)(sc^n + \rho)^{-1}\right\} \quad (23)$$

Here, the regularisation parameter is represented as ρ , the neural network's class scatter matrix is symbolized as sc^m , and the neural network's total scatter matrix is indicated as sc^n .

- The FMGKF-DENN's output obtained is

$$O_Y = \tilde{\lambda}\left(B_3 \overline{\overline{f_s}}\right) \quad (24)$$

Where, the hidden-output layer weight values are denoted as B_3 .

- Utilizing equation (25), the effort values (ξ) are computed after obtaining the output values.

$$\xi = \sum_{s=1}^{q}\left(tar^s - O_s\right)^2 \quad (25)$$

Thus, the effort values are more efficiently estimated by the proposed FMGKF-DENN approach.

4. Result and Discussion

In this section, the proposed PMFFC-centered SEE is implemented in the PYTHON platform along with the validation of its performance.

4.1 Database Description

COCOMO 81, COCOMOII-60, COCOMONASA-93, and Desharnais are utilized for the superiority measurement in the proposed methodology. Grounded on the effort and time requisite for such a project's development, the cost of a software package was determined by the COCOCMO [13] (Constructive Cost Estimation Model), COCOMOII [18], and COCOMONASA [16]. The information regarding the software projects is contained in the Desharnais [12] dataset.

4.2 Performance Analysis of proposed FMGKF-DENN

By analogizing the proposed FMGKF-DENN with the prevailing Deep Neural Network (DNN), Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), and Deep Belief Network (DBN), the proposed scheme's performance was analyzed regarding the accuracy, F-measure and deviation.

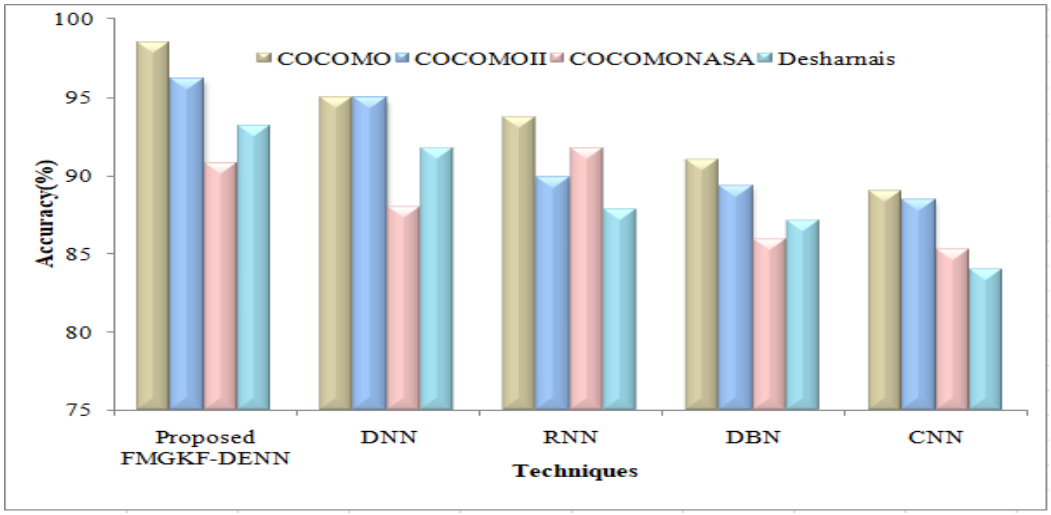

Fig 2: Performance validation based on accuracy.

Regarding accuracy, the result comparison of proposed and prevailing techniques were illustrated in figure 2. The system's better performance was revealed by high accuracy. While the other methods have lower accuracy of 95.0324% (DNN), 93.752% (RNN), 91.0547% (CNN), and 89.0124% (DBN), the proposed model attained a higher accuracy of 98.4598% for the COCOMO dataset. Identically, the accuracy may vary for other datasets also. When analogized with the prevailing models, a higher accuracy measure was exhibited by the proposed one from these values.

Table1: Comparative measurement based on F-measure.

Datasets	Proposed FNGKF-DENN	DNN	RNN	DBN	CNN
COCOMO [13]	97.3913	95.9187	93.0112	89.8812	87.1532
COCOMOII [18]	96.9541	90.1485	88.7952	85.74221	83.0414
COCOMONASA [16]	97.8752	92.1045	89.4019	86.0475	84.1032
Desharnais [12]	98.0012	94.8510	91.8854	88.0907	86.9520

The proposed technique's f-measure is depicted in table 1. When analogized with the prevailing models, the proposed model has a higher f-measure of 97.3913% in the COCOMO dataset. A lower f-measure of 83.0414% for the COCOMOII dataset was obtained by the prevailing techniques like CNN than the proposed one. Likewise, the other methods also obtained a lower f-measure than the proposed one. The prevailing technique's decreased performance was exhibited by the lower f-measure values. Hence it is evident that the proposed one tends to be more precise and accurate.

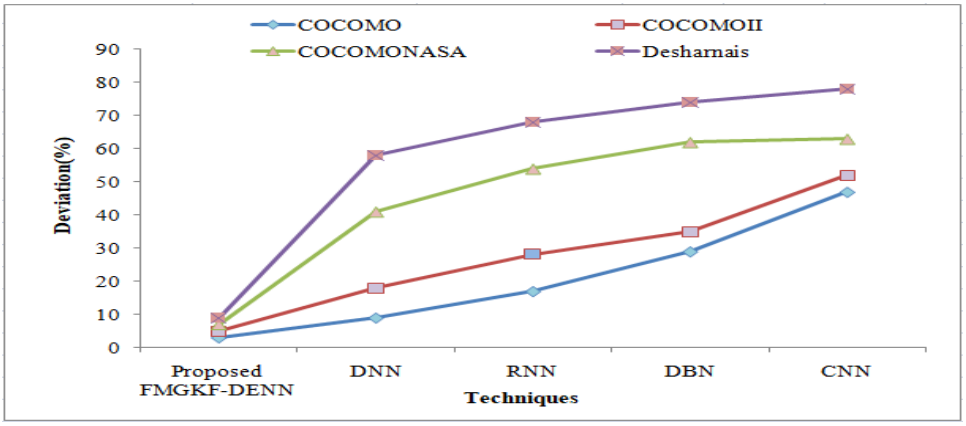

Fig 3: Performance measurement based on the deviation.

A lower deviation value of 5% was attained by the proposed model for the COCOMOII dataset, which was depicted in figure 3. A higher deviation of DNN (18%), RNN (28%), DBN (35%), and CNN (52%) was exhibited by the prevailing models on the other hand. Identically, variation occurred in the other dataset's deviation values. Hence, it is apparent that several prevailing methods were outperformed by the proposed work.

5. Conclusion

Using the FMGKF-DENN methodology, a novel PMFFC-based SEE was proposed in this paper. The proposed scheme entails a number of steps, including the determination of specific factors, project grouping, and code splitting. Then, using a feature extraction and selection technique, EE is carried out. Regarding some performance metrics, the proposed scheme along with the prevailing techniques (DNN, RNN, DBN, CNN) performance analysis and the comparative analysis is executed to validate the proposed algorithms' efficiency. Therefore, higher metrics rates of 98.4598% accuracy, 3% deviation, and 97.3913% F-measure were attained by the proposed FMGKF-DENN for the COCOMO dataset. The proposed approach can be expanded in the future by incorporating more sophisticated algorithms to improve performance.

References

[1] Sharma and N. Chaudhary, "Linear regression model for agile software development effort estimation," 2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering, ICRAIE 2020 - Proceeding, vol. 2020, pp. 4–7, 2020, doi: 10.1109/ICRAIE51050.2020.9358309.

[2] S. Shukla, S. Kumar, and P. R. Bal, "Analyzing effect of ensemble models on multi-layer perceptron network for software effort estimation," *Proceedings - 2019 IEEE World Congress on Services, SERVICES 2019*, vol. 2642–939X, pp. 386–387, 2019, doi: 10.1109/SERVICES.2019.00116.

[3] O. Malgonde and K. Chari, "An ensemble-based model for predicting agile software development effort", *Empirical Software Engineering*, 2019. doi: 10.1007/s10664-018-9647-0.

[4] Y. Mahmood, N. Kama, A. Azmi, and M. Ali, "Improving estimation accuracy prediction of software development effort: a proposed ensemble

model," *2nd International Conference on Electrical, Communication and Computer Engineering, ICECCE 2020*, no. June, pp. 12–13, 2020, doi: 10.1109/ICECCE49384.2020.9179279.

[5] R. Sari Dewi and R. Sarno, "Software effort estimation using early COSMIC to substitute use case weight," *Proceedings - 2020 International Seminar on Application for Technology of Information and Communication: IT Challenges for Sustainability, Scalability, and Security in the Age of Digital Disruption, ISemantic 2020*, X, pp. 214–219, 2020, doi: 10.1109/iSemantic50169.2020.9234227.

[6] D. S. Senevirathne and T. K. Wijayasiriwardhane, "Extending use-case point-based software effort estimation for Open Source freelance software development," *Proceedings - International Research Conference on Smart Computing and Systems Engineering. SCSE 2020*, pp. 188–194, 2020, doi: 10.1109/SCSE49731.2020.9313007.

[7] H. D. P. De Carvalho, R. Fagundes, and W. Santos, "Extreme learning machine applied to software development effort estimation," *IEEE Access*, vol. 9, pp. 92676–92687, 2021, doi: 10.1109/ACCESS.2021.3091313.

[8] K. Korenaga, A. Monden, and Z. Yucel, "Data smoothing for software effort estimation," *Proceedings - 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. SNPD 2019*, pp. 501–506, 2019, doi: 10.1109/SNPD.2019.8935841.

[9] M. Fernández-Diego, E. R. Méndez, F. González-Ladrón-De-Guevara, S. Abrahão, and E. Insfran, "An update on effort estimation in agile software development: A systematic literature review," *IEEE Access*, vol. 8, pp. 166768–166800, 2020, doi: 10.1109/ACCESS.2020.3021664.

[10] M. Daud and A. A. Malik, "Improving the accuracy of early software size estimation using analysis-to-design adjustment factors (ADAFs)," *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3085752.

[11] P. Phannachitta, "On an optimal analogy-based software effort estimation," *Information and Software Technology.*, vol. 125, no. May, p. 106330, 2020, doi: 10.1016/j.infsof.2020.106330.

[12] T. R. Benala and R. Mall, "DABE: Differential evolution in analogy-based software development effort estimation," *Swarm and Evolutionary Computation.*, vol. 38, pp. 158–172, 2018, doi: 10.1016/j.swevo.2017.07.009.

[13] T. Xia, R. Shu, X. Shen, and T. Menzies, "Sequential model optimization for software effort estimation," *IEEE Transactions on Software Engineering.*, vol. 5589, no. c, pp. 1–16, 2020, doi: 10.1109/TSE.2020.3047072.

[14] Minku, L. L., (2019). A novel online supervised hyperparameter tuning procedure applied to cross-company software effort estimation. *Empirical Software Engineering*. https://doi.org/10.1007/s10664-019-09686-w

[15] J. T. H. Jose Thiago and A. L. I. Oliveira, "Ensemble effort estimation using dynamic selection," *Journal of Systems and Software.*, vol. 175, p. 110904, 2021, doi: 10.1016/j.jss.2021.110904.

[16] J. Frank Vijay, "Enrichment of accurate software effort estimation using fuzzy-based function point analysis in business data analytics," *Neural Computing. Applications.*, vol. 31, no. 5, pp. 1633–1639, 2019, doi: 10.1007/s00521-018-3565-3.

[17] V. Resmi, S. Vijayalakshmi, and R. S. Chandrabose, "An effective software project effort estimation system using optimal firefly algorithm," *Cluster Computing.*, vol. 22, pp. 11329–11338, 2019, doi: 10.1007/s10586-017-1388-0.

[18] A. Fadhil, R. G. H. Alsarraj, and A. M. Altaie, "Software cost estimation based on dolphin algorithm," *IEEE Access*, vol. 8, pp. 75279–75287, 2020, doi: 10.1109/ACCESS.2020.2988867.

[19] Singh, J., Bilgaiyan, S., Shankar Prasad Mishra, B., Dehuri, S., (2020), "A journey towards bio-inspired techniques in software engineering", *Intelligent Systems Reference Library*, https://doi.org/10.1007/978-3-030-40928-9

[20] S. Bilgaiyan, S. Mishra, and M. Das, "Effort estimation in agile software development using experimental validation of neural network models," *International Journal of Information Technology.*, vol. 11, no. 3, pp. 569–573, 2019, doi: 10.1007/s41870-018-0131-2..

[21] Ms. Madhuri Zambre. (2012). Performance Analysis of Positive Lift LUO Converter . International Journal of New Practices in Management and Engineering, 1(01), 09 - 14. Retrieved from http://ijnpme.org/index.php/IJNPME/article/view/3

[22] Sharma, M. K. (2021). An Automated Ensemble-Based Classification Model for The Early Diagnosis of The Cancer Using a Machine Learning Approach. Machine Learning Applications in Engineering Education and Management, 1(1), 01–06. Retrieved from http://yashikajournals.com/index.php/mlaeem/article/view/1