*International Journal of Computational and Experimental Science and ENgineering (IJCESEN)*

*Vol. 11-No.2 (2025) pp. 3171-3184*
*http://www.ijcesen.com*

***ISSN: 2149-9144***



**Research Article**

# Analysis Of Software Cost Estimation and Debt Management Based on Deep Learning Approaches

**K. Ravikumar[1]*, K. Saravanakumar[2], Anand Viswanathan[3], Mathivanan Durai[4,5], S. Devi[6], S. Kalaiselvan[7]**

[1]Dhanalakshmi Srinivasan College of Engineering and Technology, Professor, Department of Information Technology, Chennai Tamilnadu, India
* **Corresponding Author Email:** ravikumarcsephd@gmail.com - **ORCID:** 0000-0003-2826-289X

[2]Sri Eshwar College of Engineering Coimbatore, Associate Professor, Department of Computer Science and Engineering, Tamil Nadu, India.
**Email:** saravanakumar.phdauc@gmail.com - **ORCID:** 0009-0003-6353-7702

[3]Ponjesly College of Engineering, Professor Department of Computer Science and Engineering, Tamil Nadu, India.
**Email:** itsanandmtech@gmail.com- **ORCID:** 0000-0002-7754-5346

[4]Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences (SIMATS), Chennai- 602105, Tamil Nadu, India,
[5]Department of Mechanical and Design Engineering, Hongik University, Sejong 30016, South Korea.
**Email:** mathivanand04@gmail.com - **ORCID:** 0000-0001-7266-1797

[6]Nandha Engineering College, Assistant professor, Computer Applications, Erode, Tamilnadu, India
**Email:** devikrishnamca@gmail.com- **ORCID:**0000-0002-7472-2319

[7]Bannari Amman Institute of Technology, Assistant Professor, Department of Mathematics, Sathyamangalam, Erode, Tamilnadu, India
**Email:** kalaiselvan@bitsathy.ac.in - **ORCID:** 0009-0002-7832-6660

**Article Info:**

**DOI:** 10.22399/ijcesen.2064
**Received :** 05 February 2025
**Accepted :** 01 May 2025

**Keywords :**

Software
cost
Margin rate
Fuzzified
Neural Network
Z-score

**Abstract:**

The precision of Software Effort Estimates (SEE) is essential for planning, managing, and thoroughly assessing software projects, ensuring they stay on budget and schedule. Achieving accurate SEE results is vital for handling future software development tasks, addressing the challenges of overestimating and underestimating resources. The method employs machine learning (ML) assessment approaches that produce highly accurate results, evaluating based on metrics, data sets, and relevant attributes. This paper analyzes potential applications of data science in management accounting. With large amounts of data, deep learning techniques can overcome some of these limitations. Initially, we collected the dataset from a standard repository, and we started the first step of data preprocessing for reducing null and unbalanced values based on Mini-Maxi-score normalization (Mm-Z-score). The final stage is classification is based on SoftMax deep Scaling Gated Adversarial Neural Network (SmDSAN2) evaluating the cost estimation debt budget schedule and reducing the false rate for analyzing the It can predict the costs required to build or develop software cost based on SmDSAN2algorithm has shown high accuracy for Predict the necessary cost to develop Software cost and effort estimation. Estimation techniques are used by categorizing the estimation of projects created using the Fuzzy margin rate to identify various debts available in different organizations. The SmDSAN2 algorithm will help software companies follow the rules and standards and reduce the cost of the overall estimation.

## 1. Introduction

Creating reliable software products requires performing several tasks within a particular cost and time limit, from collecting requirements to maintaining them [1]. Since software analysis requires accurate task estimation, software cost estimation has become significantly more important over the past two decades. The program offered cost estimation and debt management analysis for

efficient contract negotiation, planning, and resource allocation. By using evaluation methods that improve project management, the time and cost of software development are significantly reduced while improving current processes.Future and current technology projects are essential for analyzing technology investment's profitability and potential and estimating costs. Furthermore, future information technology performance requires evaluating software program development using a wide range of specific data protection applications [2]. A software tool designed to distribute software products involves transferring or selling them to the user. Additionally, the data connected to the software tool includes programs, procedures, rules, and any provided documents for operating a data processing system.Consequently, estimating software expenses beforehand during the project planning stage is crucial. Calculating and predicting the effort required to develop a software system is part of the process [3]. Furthermore, software testing, maintenance, and requirements engineering contribute to cost estimation. These estimates empower a project manager to allocate resources and set budgets or deadlines effectively.Therefore, accurately estimating the costs needed to complete the project within the designated time and budget is crucial. Project managers possess significant expertise and experience in estimating software costs for intricate project architectures. Furthermore, the technology employed introduces complications related to project scope, requirements engineering, and the development team's capabilities [4]. The software development cost is accurately assessed to determine the influence of the likelihood that the software expansion will be completed quickly and cost-effectively.Additionally, input data for cost estimation is difficult to obtain, and the work's scope is frequently unclear, resulting in rough and incomplete estimates. Even small improvements can complicate cost management if a project relies on inaccurate cost estimates [5]. Due to the characteristics of the construction industry, a large amount of capital is required to start and continue a business in a high-risk sector.The main contribution of this section is to improve the efficiency of software cost estimation techniques using the SmDSAN2 model for accurate software cost estimation. Data preprocessing begins by collecting data from standard repositories and reducing null and unbalanced values based on the Mm-Z-score. Furthermore, feature edge threshold analysis is performed based on the FSMIR model. Similarly, the PSI method calculates the nearest value to select the maximum weighted range based on the threshold value. Additionally, the SmDSAN2 algorithm was adopted to reduce the estimation and cost estimate analysis error rate. In addition, software cost and effort estimates are generated using the SmDSAN2 algorithm, and accuracy is improved by assessing the required costs based on the SmDSAN2 algorithm.

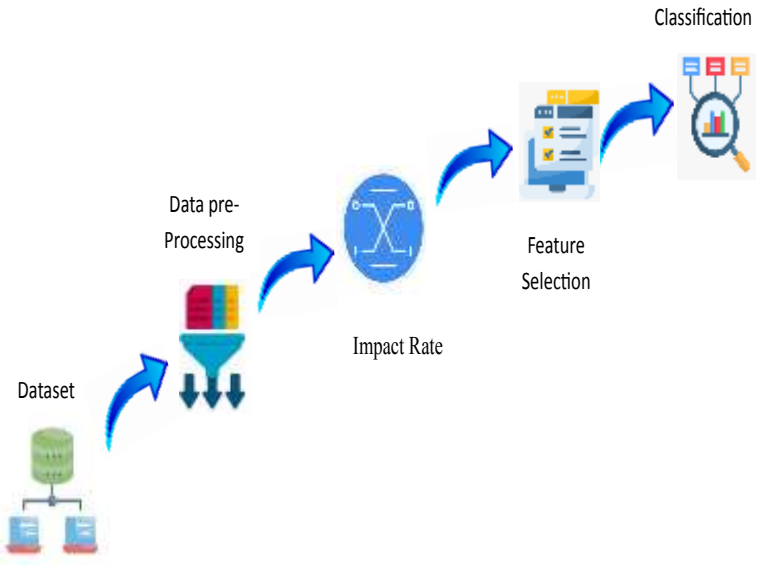


***Figure 1.*** *The Software Cost Estimation Architecture Diagram*

As shown in Figure 1, the data collected from the Kaggle dataset is processed using data pre-processing, margin impact ratio, feature selection, and classification.

## 2. Literature Survey

It examines the size distribution of software system features by assessing partial moment data through a theoretical framework that maximizes Shannon

entropy [6]. The software evaluates other systems when the component sizes exhibit a gamma distribution. A Systematic Literature Review (SLR) analyzing movements in research papers published over the past five years suggests future directions. However, this review highlights challenges and risks in providing accurate cost estimates for the software project [7]. The proposed model's Artificial Neural Network (ANN) algorithm integrates the contribution of each presented model by applying the linear combination rule to the individual estimated models [8]. However, the demand for software systems has increased significantly, and their resources, such as money, time, and labor, are in short supply. The proposed metaheuristic algorithm for parameter optimization provides near-optimal solutions at a reasonable cost [9]. Construction Cost Model (COCOMO) evaluates the accuracy of the software effort based on constraints. The proposed Artificial Bee Colony-Guided Analogy-Based Estimation (ABABE) model combines the ABC algorithm with ABE to provide accurate estimations [10]. During its training and testing stages, ABC applies the estimating method to the similarity function of ABE to generate various weights.However, choosing the best estimation algorithm is an expert and complex task [11]. To overcome these problems, this study improves the estimation using the Dolphin Bat Algorithm (DolBat) based on the COCOMO-II model. Two commonly used accuracy assessments are based on metrics such as ML ensembles. The performance of individual methods is examined on publicly available domain datasets [12]. The program uses ML techniques to integrate Convolutional Neural Networks (CNN) for time series forecasting and Particle Swarm Optimization (PSO) for cost assessment [13]. The offered method enables feature extraction and hyperparameter adjustment, reduces reliance on human input for parameter choices, and simplifies robustness calculations. The proposed method's main objective is identifying available loans for various companies [14]. This software allows agencies to maintain common standards and reduce overall evaluation costs. Evaluation of the proposed CNN method using multivariate time series data from the journal dataset [15]. CNN technology is the model input for number processing in time series problems.The author [16] used Biogeography-Based Optimization (BBO) to refine

***Table 1.*** *Software Cost Estimation Prediction Accuracy*

| Author | Year | Technique Used | Drawback | Accuracy |
|---|---|---|---|---|
| M. S. Khan [18] | 2021 | Deep Neural Network (DNN) | Software development is complex due to fast-changing needs and technology. | 0.95 |
| H. L. T. K. Nhung [19] | 2022 | Multiple Linear Regression models (MLRM) | The use case affects the prediction accuracy of point-based methods. | 75.24 |
| Varshini, P. A. G [20] | 2021 | Support Vector Machine (SVM) | Software effort estimation focuses on estimating work hours for software development/maintenance. | 0.093 |
| C. A. U. Hassan [21] | 2022 | Hybrid Of Ant Colony Optimization with BAT (HACO-BA), COCOMO | Effective software makes it difficult to achieve specific goals through project management. | 85% |
| Rao, K.E [22] | 2021 | Recursive Feature Elimination (RFE) | As the development of software products progresses rapidly, they fail to achieve their software development goals effectively. | 81.25 |
| A. Puspaningrum [23] | 2021 | Flower Pollination Algorithm (FPA) | A key issue in effort estimation is that vague and inconsistent software requirements impede accurate estimates. | 52.48% |
| K. R. Shweta [24] | 2021 | Ensemble Duck Traveler Optimization (eDTO) | Incorrect forecasts put the project at risk, and effective management requires accurate forecasts. | 91% |

COCOMO-II's coefficients for improved software project cost and effort estimation, but the resulting model's accuracy remains limited. The optimal method provided the Fuzzy Inference System (FIS) with the necessary data, cost considerations, limitations, and priorities. Calculate minimum progress and processed outputs such as cost, effort, time, and expenses [17].

Software cost estimation can be improved using defect metrics and accuracy assessments, as detailed in Table 1.

***Table 2.*** *Deep Learning Model Based on Software Effort Estimation*

| Author | Reference No | Methodology | Dataset | Performance Evaluation |
|---|---|---|---|---|
| D. K. K. Reddy | [25] | PSO | online market server | Computatinaltime, Precision, F-score |
| E. Trojovská | [26] | Zebra Optimization Algorithm (ZOA) | CEC2015, CEC2017 | Sensitivity, Scalability |
| Suresh Kumar Pemmada | [27] | Artificial Neural Network (ANN) | COCOMO, NASA dataset | Accuracy measure |
| M. Rahman | [28] | K-Nearest Neighbor (K-NN) | Albrech, Desharnais,Chia, Kemere, Mayazaki94, Maxwell,andCOCOMO | Mean Squared Error (MSE), Mean Magnitude of Relative Error (MMRE) |
| Feta, N | [30] | Fuzzy Logic Method (FLM) | COCMOI,NASA98data set | Accuracy, MMRE |
| Kassaymeh, S | [31] | Fully Connected Neural Network (FCNN) model | Software Development Effort Estimation (SEE) | Accuracy |

As Table 2 illustrates, software effort evaluations based on DL techniques were processed to evaluate various techniques, datasets, and performance metrics.

The proposed impact ratio determines the optimal clustering method and estimates the software development effort [32]. Furthermore, using the offered method based on the performance RMSE metric improved the accuracy of the FPA method to 63.68% and the EEAC method to 72.02%. Additionally, the ML technology uses fuzzy logic models based on Support Vector Regression (SVR) technology introduced by simulation [33]. Additionally, FLM models have various advantages, such as adaptability and prediction accuracy. The proposed algorithm includes various node and edge types with the solved software input problem. In addition, the initial node created an embedding to learn and predict points supporting a Heterogeneous Graph Neural Network (GGNN) model [34]. A statistical evaluation of the overall tests across both classification types indicates that the proposed activation function is highly effective, increasing the average accuracy of all competing activation functions [35]. The results also indicate that this suggested process improves the performance of the most widely employed activation functions across numerous benchmarks.

## 3. Proposed Methodology

This section utilized the proposed method for software cost estimation and debt management analysis using the Desjardins dataset obtained from the Kaggle repository. First, the Mm-Z scoring technique was used to reduce the null and unbalanced values in the dataset. Next, the feature edge was analyzed using the FSMIR method. In addition, a range-based feature selection was designed to select the closest value based on the PSI algorithm. Finally, the SmDSAN2 algorithm based on the DL model helps software companies comply with rules and standards and reduces overall evaluation costs. As shown in figure 2, the architecture diagram of the proposed SmDSAN2 project. Data from the standard repository experiences pre-processing using Mm-Z score normalization to reduce null values and imbalances. FSMIR-based thresholds define feature contour thresholds. Feature selection uses PSI to increase the weighted threshold based on these thresholds and iteratively ranks the importance of the features until the best subset is

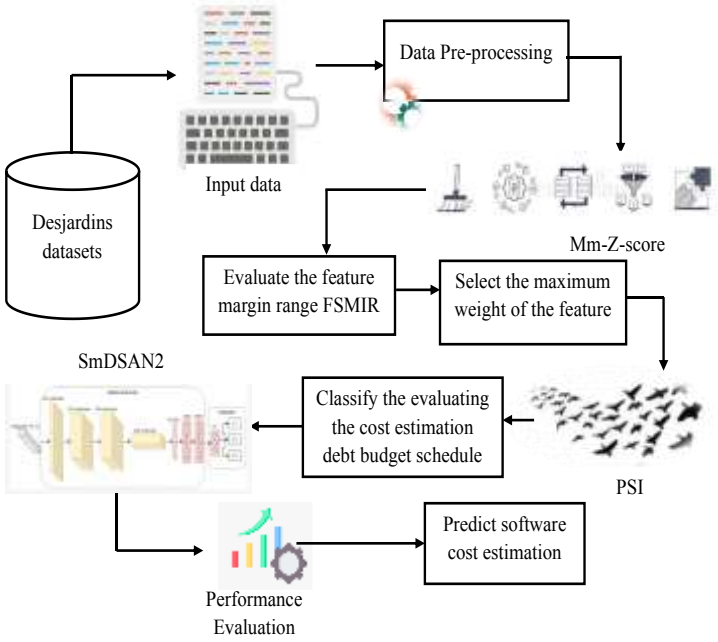


***Figure 2.*** *The Proposed SmDSAN2 Method Architecture Diagram*

obtained. Finally, SoftMax SmDSAN2 classifies data, evaluates cost estimates, minimizes false positive rates, and predicts software costs. SmDSAN2 software accurately predicts costs and efforts, classifies projects using fuzzy margin percentages, and identifies debt. This allows companies to adhere to standards and reduce overall estimated costs.

### 3.1 Dataset Collection

This section developed a proposed model to predict software cost estimates using Desjardins datasets collected from Promise repositories and Kaggle to improve accuracy. Collecting and determining the dataset's features, we analyze, process, and visualize them. Similarly, the proposed algorithm provides excellent results using Desjardin's datasets of different sizes. This project analyzes, cleans, prepares, and visualizes data to predict the organization's software effort estimation costs. In addition, the company developed an application

| id | Project | TeamExp | ManagerExp | YearEnd | Length | Effort | Transactions | Entities | Points Non Adjust | Adjustment | Points Adjust | Language |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 4 | 85 | 12 | 5152 | 253 | 52 | 305 | 34 | 302 | 1 |
| 2 | 2 | 0 | 0 | 86 | 4 | 5635 | 197 | 124 | 321 | 33 | 315 | 1 |
| 3 | 3 | 4 | 4 | 85 | 1 | 805 | 40 | 60 | 100 | 18 | 83 | 1 |
| 4 | 4 | 0 | 0 | 86 | 5 | 3829 | 200 | 119 | 319 | 30 | 303 | 1 |
| 5 | 5 | 0 | 0 | 86 | 4 | 2149 | 140 | 94 | 234 | 24 | 208 | 1 |
| 6 | 6 | 0 | 0 | 86 | 4 | 2821 | 97 | 89 | 186 | 38 | 192 | 1 |
| 7 | 7 | 2 | 1 | 85 | 9 | 2569 | 119 | 42 | 161 | 25 | 145 | 2 |
| 8 | 8 | 1 | 2 | 83 | 13 | 3913 | 186 | 52 | 238 | 25 | 214 | 1 |
| 9 | 9 | 3 | 1 | 85 | 12 | 7854 | 172 | 88 | 260 | 30 | 247 | 1 |
| 10 | 10 | 3 | 4 | 83 | 4 | 2422 | 78 | 38 | 116 | 24 | 103 | 1 |
| 11 | 11 | 4 | 1 | 84 | 21 | 4067 | 167 | 99 | 266 | 24 | 237 | 1 |
| 12 | 12 | 2 | 1 | 84 | 17 | 9051 | 146 | 112 | 258 | 40 | 271 | 1 |
| 13 | 13 | 1 | 1 | 84 | 3 | 2282 | 33 | 72 | 105 | 19 | 88 | 1 |
| 14 | 14 | 3 | 4 | 85 | 8 | 4172 | 162 | 61 | 223 | 32 | 216 | 1 |

***Figure 3.*** *Dataset Feature Collection*

designed to improve software cost estimation. This application, which is provided to internal stakeholders, allows them to interact with internal and external data and quickly access invoice estimates. To assess the software cost estimates for 82 features, a training set of 11 and a test set of 71 were utilized.

Figure 3 illustrates the significance of quality requirement attributes in assessing software effort with the feature set-based data from the Desharnais dataset. This dataset is widely recognized, much like other datasets for training and validating software cost estimation models. The dataset's features are gathered from https://www.kaggle.com/datasets/toniesteves/desharnais-dataset using the website.

### 3.2 Data Pre-Processing

An MM-Z scoring algorithm based on data preprocessing can reduce null and unbalanced values in the dataset. In addition, preprocessing steps remove errors, missing values, and outliers. All values in the dataset are numerically converted and analyzed as single numbers. In addition, the software uses the isnull () and sum () functions to check that the price estimate contains no null values. In addition, the feature values are transformed, and all measured features are scaled to the same unit of measure, determining the scale of the software effort cost estimate.

All variables were assessed by normalizing the minimum and maximum values in the dataset. Variables with different thresholds are measured and standardized as illustrated in Equations 1 and 2. Let's assume $\mathrm{P_n}$ −input variable, $\mathrm{P_{max}} - \mathrm{P_{min}}$, standardized scale value, $\mathrm{New_{max}} - \mathrm{New_{min}}$ − new minimum and maximum variable, $J$ −normalized value.

$$\mathrm{P_n} = \left(\frac{\mathrm{P_n}-\mathrm{P_{min}}}{\mathrm{P_{max}}-\mathrm{P_{min}}}\right) \times (New_{max} - New_{min}) + New_{min} \quad (1)$$

$$\mathrm{P_{max}} = maxP_{n1\le n\le J},\ \mathrm{P_{min}} = minP_{n1\le n\le J} \quad (2)$$

Calculate a Z-score to standardize the variable units of measurement based on the unit choice for each attribute, as illustrated in Equations 4 and 5. Where $I$ −interval, $A_V$ −actual value, $NA_V$ −normalized actual data.

$$I[0,1] = \frac{NA_V - min(all\ value)}{max(all\ value) - min(all\ value)} \quad (3)$$

$$I[-1,1] = \frac{NA_V - \left(max(all\ value) + min(all\ value)\right)/2}{\left(max(all\ value) + min(all\ value)\right)/2} \quad (4)$$

The software generates a cost estimate by minimizing null and unbalanced values in the dataset, employing a pre-processing technique that standardizes scaling to normalize the minimum-maximum score.

### 3.3 Fuzzified Support Margin Impact Rate (FSMIR)

This section utilizes the FSMIR model to identify various loans accessible to firms, examining the spectrum of characteristic margins. Furthermore, the fuzzy margin ratio calculates the impact ratio of development projects and enhances software evaluation techniques. The FSMIR algorithm, which analyzes fuzzy numbers, can be used to analyze the ambiguities and inaccuracies in the data. Each possible value represents a set of possible values with weights between '0' and '1'. The weight membership function is determined by dividing the mean value of fuzzy numbers into three categories: triangular, trapezoidal, and bell-shaped.

Determine the interval [0, 1] using fuzzy logic outlined in Equations 5 and 6. Calculate the classical extension of fuzzy sets into a collection of ordered pairs. Let's assume $an\ x$ −universe of discourse elements, $E$ −fuzzy set, $P$ −set of ordered pairs, and $\mu E(P)$ −membership function.

$$E = NA_V\{X|X > 6\} \quad (5)$$

$$E\{P, \mu E(P)|p\ \ P\} \quad (6)$$

As shown in equation 7, membership functions classify the ambiguities in the fuzzy set and map each point in the input space to a membership value between 0 and 1. Let's assume $m$ −modal value, $a$ −lower limit, $b$ −upper limit, $p - f, and\ f - i$ −margin value.

$$E(P) = \begin{cases} 0 & if\ P \le e\ or\ p \le f \\ (p-e)/(p-f) & if\ p \in (e,i) \\ (f-p)/(f-f) & if\ p \in (i,f) \end{cases} \quad (7)$$

Calculate the distance bounded by the upper and lower points of the curve, as shown in Equation 8. Let's assume $T(P)$ −fuzzy set,

$$T(P) = \begin{cases} 0 & if\ \ p \le e \\ 2\{(p-e)/(f-e)\}^2 & if\ p \in (e,i) \\ 1 - 2\{(p-e)/(f-e)\}^2 & if\ p \in (i,f) \\ 1 & ifp \ge f \end{cases} \quad (8)$$

As illustrated in Equation 9, estimate the lower and upper bounds of the central or kernel value. Let's assume m-value, $a - b$ −distance.

$$S(P) = \begin{cases} 0 & if (p \le e) or (p \ge h) \\ (p-e)/(p-f) & if\ p \in (e,f) \\ 1 & if\ p \in (f,g) \\ (h-p)/(h-g) & if\ p \in (g,h) \end{cases} \quad (9)$$

As shown in Equation 10, calculate the maximum ambiguity value for the left and right boundaries using the ambiguity of the triangular membership function to identify credit management based on software cost estimation. Let's assume $TFN\ (B)-$ Triangular Membership function, $i-$model value, $\beta-\alpha-$right and left boundaries, and $B-$higher value of margin impact rate.

$$Fuzziness\ TFN\ (B) = \frac{\beta-\alpha}{2i}, 0 < B < 1 \quad (10)$$

The FSMIR model identifies debt management, analyzing characteristic margins where more significant fuzziness indicates higher value.

### 3.4 Particle Swarm Intelligence (PSI)

Particle Swarm Intelligence (PSI) enhances software cost estimation accuracy through its ability to optimize the selection of cost-driving features from software project archive data. Software cost estimation requires considering project size, team experience, development tools, and complexity to determine the final expenditure. Not all the existing features generate equal precision during accuracy estimation processes. The selection of the most significant feature subset happens during PSI execution through simulated particle swarm mechanics that assign each particle to represent different feature options. The process begins by letting particles investigate different features before they readjust their positions by evaluating historical cost estimation results from their selected features. PSI performs iterations that rely on individual and group learning to drive its convergence toward an optimal feature selection, which produces minimum prediction mistakes. PSI provides automated feature selection capability using a virtual swarm of particles representing $X_i$ subsets of all features. This subset is encoded as a binary vector through equation 11,

$$X_i = B[x_{i1}, x_{i2}, \dots, x_{in}] \quad (11)$$

A value of $x_{ij} = 1$ indicates the $j-th$ feature participates in the subset, while $x_{ij} = 0$ represents exclusion from the subset. The parameter $n$ represents the number of features in the historical software project database. Particles perform space exploration by adjusting their velocity factors and position variables step by step. The velocity update equation 12 determines individual particles' motion patterns and knowledge acquisition.

$$V_i(t+1) = w.V_i(t) + c_1.r_1.\left(P_i - X_i(t)\right) + c_2.r_2.\left(G - X_i(t)\right) \quad (12)$$

The velocity vector at iteration t is represented by $V_i(t)$, and w represents the inertia weight, which dictates how much the previous direction should be kept in the process. The best-known feature subsets of individual particles and the swarm correspond to $P_i$ and $G$ variables. The acceleration coefficients $c_1$ and $c_2$ regulate how much individual and swarm experience affects the search process, and $r_1$ and $r_2$ serve as random range numbers [0,1] for ensuring randomness in movement. The position update step selects the new feature subset a particle will use after its velocity receives its update as below equation 13.

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (13)$$

A sigmoid transfer function receives the velocity signal to convert it into probability space before thresholding occurs in equation 14.

$$x_{ij}(t+1) = \begin{cases} 1\ if\ sigmoid\left(v_{ij}(t+1)\right) > \theta \\ 0 \qquad otherwise \end{cases} \quad where\ sigmoid(v) = \frac{1}{1+e^{-v}} \quad (14)$$

Each object selection occurs probabilistically under the parameters learned from the particle's behavior. Each particle evaluates its feature subset quality through a fitness function that usually measures the Mean Absolute Error (MAE) of the cost estimation model as presented in equation 15.

$$MAE = \frac{1}{m}\sum_{k=1}^{m}\left|\hat{C}_k - C_k\right| \quad (15)$$

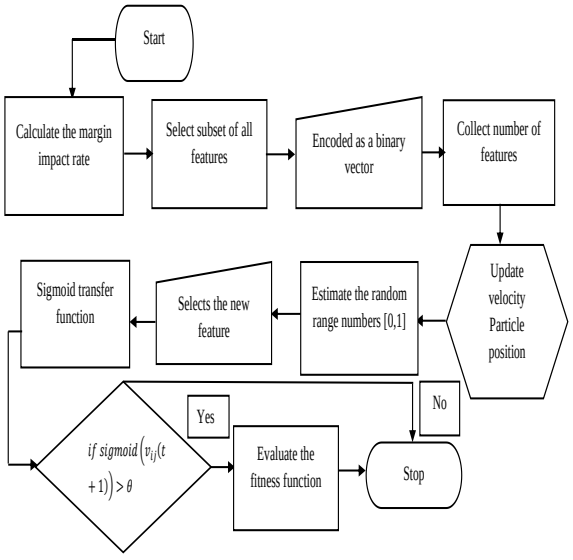


***Figure 4.*** *Flowchart Diagram Using PSI Method*

The selected features calculate estimated costs using $\hat{C}_k$ for the $k-th$ project among past software projects $m$, while $C_k$ represents the actual recorded cost. A lower Mean Absolute Error value signifies better accuracy for predicting results with a selected feature subset. The PSI process guides the swarm toward its optimal search space by finding the minimum error subset, making the model more efficient, and removing useless data to generate predictive models for software cost estimation, as shown in Figure 4.

### 3.5 SoftMax deep Scaling Gated Adversarial Neural Network (SmDSAN2)

Software cost estimation utilizes classification models that determine project cost categories between low, medium, and high by analyzing historical software project features. Using SmDSAN2-based methods enhances the performance of these classification models by effectively generating realistic synthesized data that supplements small or unbalanced training datasets. A SmDSAN2 framework operates with two neural networks: the generator generates synthetic data that resembles real-world project data through learning and a discriminator network that separates original and synthetic data. After training completion, the discriminator receives further adjustments before becoming operational as a classification tool for software cost categories. The Softmax function operates in the discriminator's output layer for multi-class classification. The Softmax function transforms output logits into class probability distributions with a sum of 1. Using synthesized data from the generator combined with the Softmax function, the classifier achieves greater robustness, mainly when dealing with limited or unbalanced datasets. The proposed classification framework, built with SmDSAN2 components, ultimately enhances the reliability and generalization capabilities of models used for software cost estimation.

Software cost estimation through SmDSAN2 starts with the generator $G$ accepting a noise vector $z$ as its input parameter. A vector z originates from a known prior distribution $p_z(z)$. The distribution most often takes the form of multivariate Gaussian or uniform types. Using noise from this distribution, the generator creates an artificial $\tilde{x}$ software feature vector that emulates project attributes such as team allotment and coding length, execution span, and programming tools applied. The generator contains parameters $\theta_G$ consisting of weights and biases that should be adjusted during training. This process is defined by equation 16,

$$z \sim p_z(z),\ \tilde{x} = G(z;\theta_G) \qquad (16)$$

The discriminator $D$ receives accurate historical software project data as $x$ vectors and synthesized data as $\tilde{x}$ vectors from the generator. The probability assignment $D(x)$ through $D$ determines the likelihood that the input belongs to accurate data. The discriminator contains parameters $\theta_D$ and operates through continual updates to better detect realistic inputs from fake ones. The algorithm ensures the generator creates more realistic feature vectors through its mechanism as below equation 17.

$$D(x;\theta_D) \in [0,1],\ D(\tilde{x}) = D\big(G(z)\big) \qquad (17)$$

In SmDSAN2, adversarial training operates through minimax optimization, where the generator works to deceive the discriminator while the discriminator dedicates efforts to detecting actual versus fake data instances. The accurate data distribution, known as $p_{data}$, has an expectation that matches the noise distribution, $p_z$, which generates synthetic samples. The equation 18 represents the fundamental learning principle between these elements.

$$\min_G \max_D L_{GAN} = E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p_z}\big[\log 1 - D\big(G(z)\big)\big] \qquad (18)$$

Through cost estimation, this method enables the creation of synthetic projects that contain genuinely cost-related characteristics that boost the model's generalization ability. An attached discriminator classifier generates output logits as a vector $[z_1, z_2, \ldots, z_K]$ where each $z_i$ value indicates the scoring for class $i$, using $K$ as the total number of categorized costs (such as low or high). The logging results pass through the Softmax function, which transforms them into cost class probabilities while maintaining the validity of distribution probabilities as below equation 19.

$$P(y = i|x) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, for\ i = 1,2,\ldots,K \qquad (19)$$

The predicted probability that software project x belongs to cost class $i$ is expressed through $P(y = i|x)$. This approach allows for probabilistic predictions that are also easy to interpret. The selected cost class prediction relies on equation 20, which determines the output from Softmax probability distributions.

$$\hat{y} = \arg\max_i P(y = i|x) \qquad (20)$$

The ultimate classified outcome, $\hat{y}$, determines which category will identify a software project among the low, medium, or high-cost options, thus

assisting with managerial resource planning. The network receives training with cross-entropy loss to enhance classification results by comparing predicted probabilities and true one-hot vector $[y_1, y_2, \ldots, and\ y_K]$ class labels, where only one element in the vector contains a value of 1. The model absorbs penalties through the loss function that arises from wrong predictions in equation 21.

$$L_{CE} = -\sum_{i=1}^{K} y_i \log P(y = i|x) \quad (21)$$

The loss direction prompts the classifier to generate elevated probabilistic outputs for proper cost categories, which results in better performance with labeled data. The training regime that unites generation with classification components runs their combined loss function with control parameters $\lambda_1$ and $\lambda_2$ to balance their weight as below equation 22.

$$L_{total} = \lambda_1 . L_{GAN} + \lambda_2 . L_{CE} \quad (22)$$

The combined loss mechanism enables the generator to generate valid project data, contributing to high classification accuracy for software cost levels. Thus, the generator can solve real-world assessment operations with minimal or skewed data sets.

## 4. Result and Discussion

Several test and training sets are provided in this section to assess the software effort estimation's performance. The outcomes of three different estimate techniques—SVM, MLRM, and DNN—obtained using the prior framework are compared to the results of the suggested approach. Additionally, software evaluation was enhanced utilizing features supplied by Kaggle for all approaches employing training and test datasets. The suggested SmDSAN2 methodology outperforms other models in terms of accuracy verification. Therefore, MMRE and Root Mean Square Error (RMSE) criteria are used to assess accuracy, recall, F1 score, and precision to increase the accuracy of software cost assessment.

***Table 3.*** *Simulation Parameter*

| Simulation | Variable |
|---|---|
| Dataset Name | Desharnais dataset |
| No of Dataset | 82 |
| Training | 71 |
| Testing | 11 |
| Language | Python |
| Tool | Jupyter |

The simulation uses the parameter-based variables as shown in Table 3. In addition, software cost estimation is enhanced by leveraging Jupyter tools in Python with training and testing data.

***Table 4.*** *Confusion Metrix*

| Metrices | Formula |
|---|---|
| Accuracy | $\frac{TP+TN}{TP+FP+TN+FN}$ |
| Precision (Pre) | $\frac{TP}{TP+FP}$ |
| Recall (Rec) | $\frac{TP}{TP+FN}$ |
| F1-Score | $\frac{2*Pre*Rec}{Pre+Rec}$ |
| MMRE | $\frac{1}{N}\sum_j \frac{E_j-\hat{E}_j}{E_j}$ |
| RMSE | $\sqrt{\frac{\sum(P_i-O_i)^2}{n}}$ |

Table 4 illustrates that the costs associated with software deployment can be optimized by applying different formulas from the confusion matrix.

***Table 5.*** *Performance of Precision*

| No of Records | SVM | MLRM | DNN | SmDSAN2 |
|---|---|---|---|---|
| 25 | 58 | 62 | 66 | 69 |
| 50 | 62 | 67 | 70 | 73 |
| 75 | 67 | 71 | 74 | 77 |
| 100 | 71 | 75 | 78 | 81.8 |

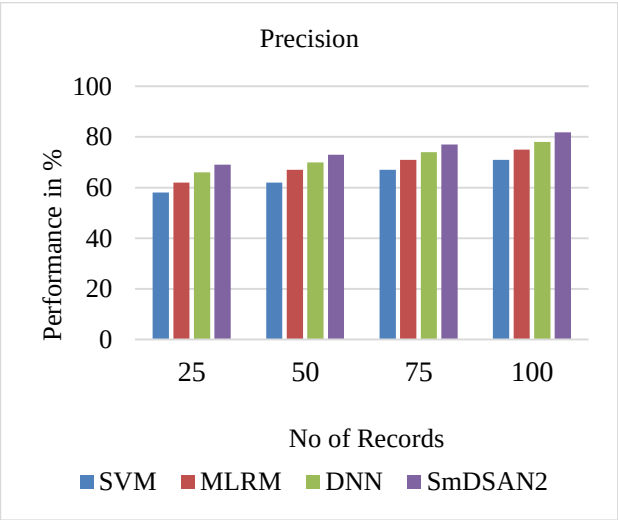


***Figure 5.*** *Analysis of Precision*

As shown in Figure 5 and Table 5, the proposed SmDSAN2 model accuracy performance was validated by comparing it with other models using different data sets and evaluation criteria. Furthermore, comparisons between the previous SVM, MLRM, and DNN methods and the proposed SmDSAN2 method show that the precision of software cost estimation has improved. The proposed SmDSAN2 method achieved an analysis precision of 81.8%, surpassing the earlier methods: SVM at 71%, MLRM at 75%, and DNN at 78.9%.

***Table 6.*** *Performance of Recall*

| No of Records | SVM | MLRM | DNN | SmDSAN2 |
|---|---|---|---|---|
| 25 | 60 | 65 | 69 | 72 |
| 50 | 65 | 69 | 73 | 76 |
| 75 | 70 | 74 | 76 | 80 |
| 100 | 74 | 78 | 79 | 84.6 |

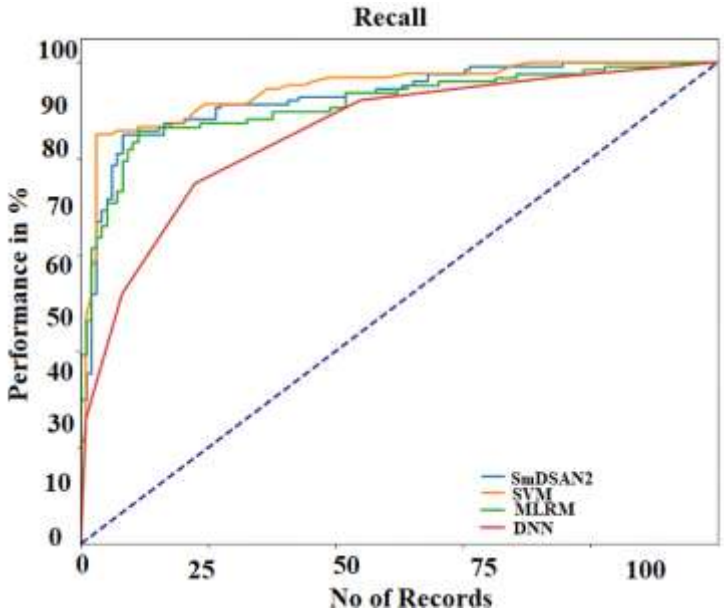


***Figure 6.*** *Analysis of Recall*

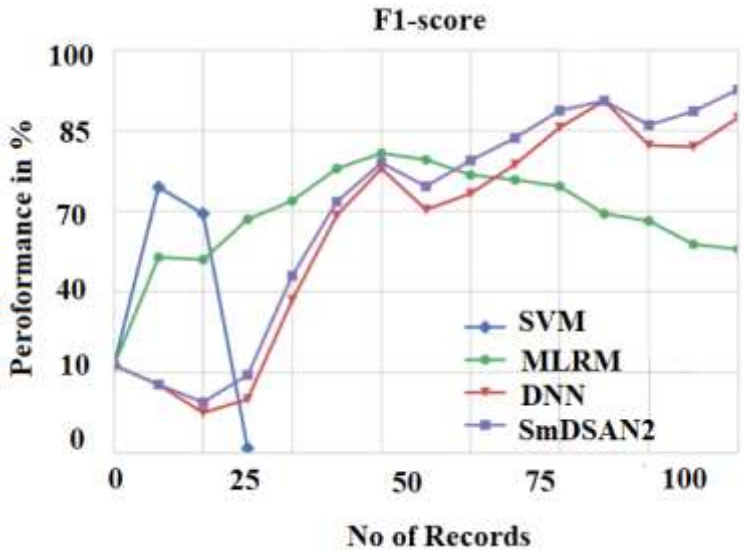


***Figure 7.*** *Analysis of F1-Score*

Figure 6 and Table 6 illustrate the recall performance of the proposed SmDSAN2 model alongside various other models across different datasets and evaluation measures to validate its effectiveness. In addition, when we compare the prior SVM, MLRM, and DNN techniques with the SmDSAN2, an enhancement in recall performance can be observed in software cost estimation. The

SmDSAN2 method achieves a recall performance of 84.6%, while the previous methods show recall rates of 74%, 78%, and 79%.

***Table 7.*** *Performance of F1-Score*

| No of Records | SVM | MLRM | DNN | SmDSAN2 |
|---|---|---|---|---|
| 25 | 64 | 67 | 70 | 74 |
| 50 | 69 | 70 | 74 | 78 |
| 75 | 72 | 76 | 78 | 82 |
| 100 | 75 | 79 | 82 | 86.7 |

Figure 7 and Table 7 show the suggested SmDSAN2 model's F1-score performance along with that of several other models on various datasets and evaluation metrics to confirm its efficacy. Furthermore, the SmDSAN2 improves F1-score performance in software cost estimation compared to the previous SVM, MLRM, and DNN approaches. The recall performance of the SmDSAN2 approach is 86.7%, whereas the F1-score rates for the earlier methods were 75%, 79%, and 82%, respectively.

***Table 8.*** *Performance of Accuracy*

| No of Records | SVM | MLRM | DNN | SmDSAN2 |
|---|---|---|---|---|
| 25 | 67 | 70 | 75 | 78 |
| 50 | 70 | 73 | 79 | 82 |
| 75 | 74 | 79 | 84 | 89 |
| 100 | 79 | 84 | 89 | 95.6 |

Figure 8 and Table 8 show the accuracy performance of the proposed SmDSAN2 model and several models on different datasets and evaluation scales to validate its performance. Additionally, SmDSAN2 improves the accuracy of software cost estimation compared to previous SVM, MLRM, and DNN methods. The SmDSAN2 method obtained an accuracy performance of 95.6% compared to 79%, 84%, and 89% of the previous method.

***Table 9.*** *Performance of MMRE*

| No of Records | SVM | MLRM | DNN | SmDSAN2 |
|---|---|---|---|---|
| 25 | 21 | 19 | 17 | 15 |
| 50 | 18 | 17 | 15 | 12 |
| 75 | 14 | 15 | 13 | 10 |
| 100 | 16 | 13 | 11 | 0.7 |

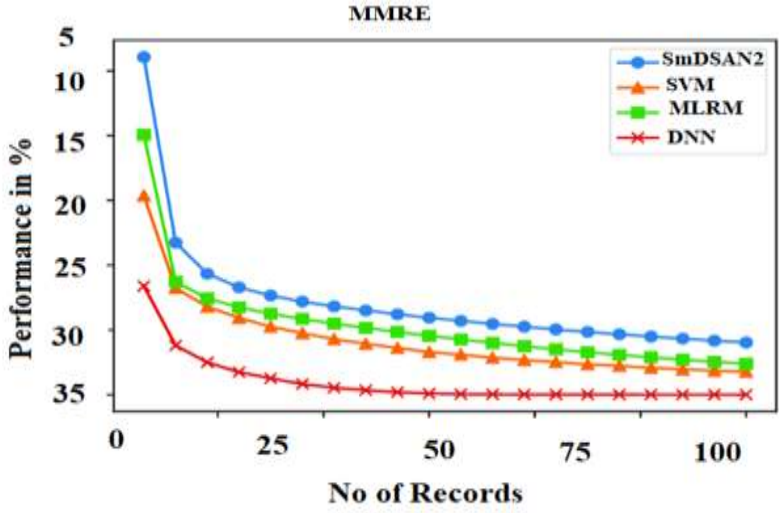


***Figure 9.*** *Analysis of MMRE*

Figure 9 and Table 9 illustrate the suggested SmDSAN2 model's MMRE performance compared to several other models on various datasets and evaluation metrics to confirm its efficacy. Furthermore, the SmDSAN2 exhibits better MMRE performance in software cost estimates than the earlier SVM, MLRM, and DNN approaches. The SmDSAN2 approach's MMRE efficiency is 0.7%, but the MMRE rates of the previous methods are reduced by 16%, 13%, and 11%, respectively. Figure 10 and Table 10 demonstrate the RMSE performance comparison of the proposed SmDSAN2 model with several models on various datasets and the evaluation metrics to validate its

***Table 10.*** *Performance of RMSE*

| No of Records | SVM | MLRM | DNN | SmDSAN2 |
|---|---|---|---|---|
| 25 | 20 | 18 | 15 | 12 |
| 50 | 18 | 16 | 13 | 9 |
| 75 | 17 | 15 | 11 | 8 |
| 100 | 15 | 12 | 0.9 | 0.5 |

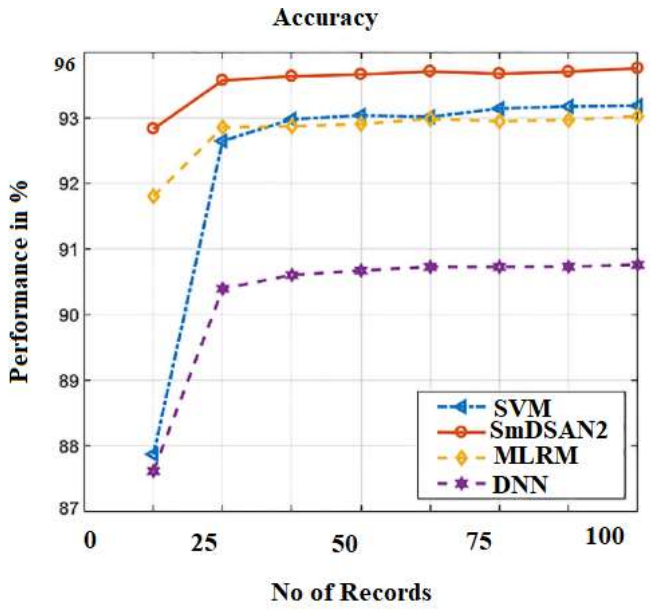


***Figure 8.*** *Analysis of Accuracy*

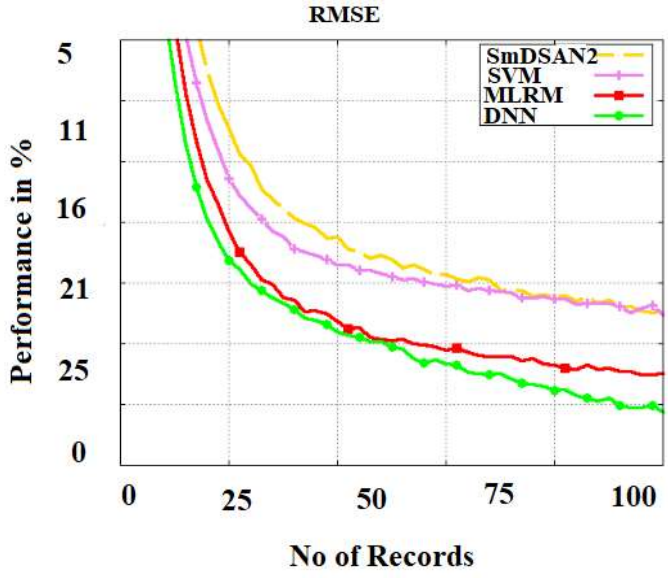


***Figure 10.*** *Analysis of RMSE*

performance. The SmDSAN2 approach's RMSE performance is 0.5%, while the RMSE ratios of previous methods are lower: 15%, 12%, and 0.9%, respectively. Furthermore, SmDSAN2 exhibits better RMSE performance in software cost estimation than previous SVM, MLRM, and DNN methods.

## 5. Conclusion

In conclusion, the effectiveness of DL-based software cost estimation using SmDSAN2 for accurate software cost estimation. Data pre-processing first gathers data from a standard repository and then reduces nulls and unbalanced values based on Mm-Z scores. Also, feature edge thresholding is analyzed based on the FSMIR model. Similarly, the PSI method selects the maximum weighted range by calculating the closest value based on a range. In addition, the SmDSAN2 algorithm is used to reduce the estimation error rate and perform cost estimation analyses. In addition, software cost and effort estimates are generated using the SmDSAN2 algorithm, and accuracy is improved by estimating the required costs based on the SmDSAN2 algorithm. Additionally, SmDSAN2

enhances the accuracy of software cost estimates compared to previous SVM, MLRM, and DNN methods. The SmDSAN2 method achieved an accuracy performance of 95.6%, compared to the 79%, 84% and 89% accuracy of the previous methods.

## Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

## References

[1] Vanathi, D., Anusha, K., Ahilan, A., & Suniram, S. E. (2024). Software cost and effort estimation using dragonfly whale optimized multilayer perceptron neural network. *Alexandria Engineering Journal, 103*, 30–37. https://doi.org/10.1016/j.aej.2024.04.043

[2] Qassem, N. T., & Saleh, I. A. (2023). Survey of cost estimating software development using machine learning. *International Research Journal of Innovations in Engineering and Technology (IRJIET), 7*(12), 67–72. https://doi.org/10.47001/IRJIET/2023.712009

[3] Hammad, M. (2023). Software cost estimation using stacked ensemble classifier and feature selection. *International Journal of Advanced Computer Science and Applications (IJACSA), 14*(6).

[4] Rashid, C. H., Shafi, I., Khattak, B. H. A., Safran, M., Alfarhood, S., & Ashraf, I. (2025). ANN-based software cost estimation with input from the COCOMO: CANN model. *Alexandria Engineering Journal, 113*, 681–694. https://doi.org/10.1016/j.aej.2024.11.042

[5] Hashemi, S. T., Ebadati, O. M., & Kaur, H. (2020). Cost estimation and prediction in construction projects: A systematic review on machine learning techniques. *SN Applied Sciences, 2*, 1703. https://doi.org/10.1007/s42452-020-03497-1

[6] Sharma, S., Pendharkar, P. C., & Karmeshu. (2022). Learning component size distributions for software cost estimation: Models based on arithmetic and shifted geometric means rules. *IEEE Transactions on Software Engineering, 48*(12), 5136–5147. https://doi.org/10.1109/TSE.2021.3139216

[7] Rashid, C. H., et al. (2023). Software cost and effort estimation: Current approaches and future trends. *IEEE Access, 11*, 99268–99288. https://doi.org/10.1109/ACCESS.2023.3312716

[8] Ali, S. S., Ren, J., Zhang, K., Wu, J., & Liu, C. (2023). Heterogeneous ensemble model to optimize software effort estimation accuracy. *IEEE Access, 11*, 27759–27792. https://doi.org/10.1109/ACCESS.2023.3256533

[9] Alsheikh, N. M., & Munassar, N. M. (2023). Improving software effort estimation models using grey wolf optimization algorithm. *IEEE Access, 11*, 143549–143579. https://doi.org/10.1109/ACCESS.2023.3340140

[10] Shah, M. A., Jawawi, D. N. A., Isa, M. A., Younas, M., Abdelmaboud, A., & Sholichin, F. (2020). Ensembling artificial bee colony with analogy-based estimation to improve software development effort prediction. *IEEE Access, 8*, 58402–58415. https://doi.org/10.1109/ACCESS.2020.2980236

[11] Fadhil, A. A., Alsarraj, R. G. H., & Altaie, A. M. (2020). Software cost estimation based on dolphin algorithm. *IEEE Access, 8*, 75279–75287. https://doi.org/10.1109/ACCESS.2020.2988867

[12] Mahmood, Y., et al. (2022). Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation. *Software: Practice and Experience, 52*(1), 39–65.

[13] Mohamed, M., Emam, O., & Azzam, S. M. (2024). Software cost estimation prediction using a convolutional neural network and particle swarm optimization algorithm. *Scientific Reports, 14*(1), 13129. https://doi.org/10.1038/s41598-024-63025-8

[14] Ravikumar, K., & Gunasekaran, G. (2018). Software cost estimation technique with technical debt management using orchestration. *International Journal of Advanced Research in Engineering and Technology (IJARET), 9*(6), 104–110.

[15] Putra, A. B., et al. (2022). PSO-based hyperparameter tuning of CNN multivariate time-series analysis. *Journal of Online Information, 7*(2), 193–202.

[16] Ullah, A., Wang, B., Sheng, J., Long, J., Asim, M., & Sun, Z. (2021). Optimization of software cost estimation model based on biogeography-based optimization algorithm. *Intelligent Decision Technologies, 14*(4), 441–448.

[17] Vanathi, D., Anusha, K., Ahilan, A., & Salinda Eveline Suniram, A. (2024). Software cost and effort estimation using dragonfly whale optimized multilayer perceptron neural network. *Alexandria Engineering Journal*, 103, 30-37. https://doi.org/10.1016/j.aej.2024.04.043.

[18] Khan, M. S., et al. (2021). Metaheuristic algorithms in optimizing deep neural network model for software effort estimation. *IEEE Access, 9*, 60309–60327.

[19] Nhung, H. L. T. K., Van Hai, V., Silhavy, R., Prokopova, Z., & Silhavy, P. (2022). Parametric software effort estimation based on optimizing correction factors and multiple linear regression. *IEEE Access, 10*, 2963–2986.

[20] Varshini, P. A. G., Kumari, A. K., & Varadarajan, V. (2021). Estimating software development efforts using a random forest-based stacked ensemble approach. *Electronics, 10*, 1195.

[21] Hassan, C. A. U., et al. (2022). Optimizing deep learning model for software cost estimation using hybrid meta-heuristic algorithmic approach. *Computational Intelligence and Neuroscience, 2022*, 1–20.

[22] Rao, K. E., & Rao, G. A. (2021). Ensemble learning with recursive feature elimination integrated software effort estimation: A novel approach. *Evolutionary Intelligence, 14*, 151–162. https://doi.org/10.1007/s12065-020-00360-5

[23] Puspaningrum, A., Muhammad, F. P. B., & Mulyani, E. (2021). Flower pollination algorithm for software effort coefficients optimization to improve effort estimation accuracy. *JUITA Jurnal Informatika, 9*(2), 139–144.

[24] Shweta, K. R. (2021). Software cost and effort estimation using ensemble duck traveler optimization algorithm (eDTO) in earlier stage. *Turkish Journal of Computer and Mathematics Education, 12*, 3300–3311.

[25] Reddy, D. K. K., & Behera, H. S. (2020). Software effort estimation using particle swarm optimization: Advances and challenges. In *Proceedings of CIPR* (pp. 243–258).

[26] Trojovská, E., Dehghani, M., & Trojovský, P. (2022). Zebra optimization algorithm: A new bio-inspired optimization algorithm for solving optimization algorithm. *IEEE Access, 10*, 49445–49473.

[27] Pemmada, S. K., & Behera, H. S. (2020). Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades. *Computer Science Review, 38*, 100288. https://doi.org/10.1016/j.cosrev.2020.100288

[28] Khan, B., Khan, W., Arshad, M., & Jan, N. (2022). Software cost estimation: Algorithmic and non-algorithmic approaches. *International Journal of Data Science and Advanced Analytics, 2*(2), 1–5.

[29] Rahman, M., Sarwar, H., Kader, M. A., Gonçalves, T., & Tin, T. T. (2024). Review and empirical analysis of machine learning-based software effort estimation. *IEEE Access, 12*, 85661–85680. https://doi.org/10.1109/ACCESS.2024.3404879

[30] Feta, N. R. (2022). Integration of fuzzy logic method and algorithm to prediction timeliness and software development cost. *Jurnal Teknologi Nusa Mandiri, 19*(1), 46–54.

[31] Kassaymeh, S., et al. (2024). Software effort estimation modelling and fully connected artificial neural network optimization using soft computing techniques. *Cluster Computing, 27*(1), 737–760.

[32] Van, H. V., et al. (2022). Toward improving the efficiency of software development effort estimation via clustering analysis. *IEEE Access, 10*, 83249–83264.

[33] Upreti, K., et al. (2022). Fuzzy logic-based support vector regression (SVR) model for software cost estimation using machine learning. In *ICT Systems and Sustainability: Proceedings of ICT4SD 2021* (pp. [pages]). Springer.

[34] Phan, H., & Jannesari, A. (2022). Heterogeneous graph neural networks for software effort estimation. In *Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*.

[35] Alkhouly, A. A., Mohammed, A., & Hefny, H. A. (2021). Improving the performance of deep neural networks using two proposed activation functions. *IEEE Access, 9*, 82249–82271.