

PARTICLE SWARM OPTIMIZATION FOR TUNING HYPER PARAMETERS OF EXTREME LEARNING MACHINE-BASED SOFTWARE DEVELOPMENT COST ESTIMATION

V. Venkataiah, Sarmista Rath and Sambari Koushik Kumar

Department of Computer Science and Engineering, CMR College of Engineering and Technology, India

Abstract

Estimation of software cost is crucial and essential in efficient project management. Modern projects exhibit high complexity and nonlinear relationships, and inaccurate estimations lead to budget overruns, delivery delays and inefficient resource utilization. This study suggests an Extreme learning machine-based cost estimation. To handle scale variation and reduce bias this method includes preprocessing historical data and applies single layer feed-forward ELM architecture consisting of several hidden parameters. Particle Swarm Optimization (PSO) and feature selection is used for improved performance, and it is validated through performance metrics like MMRE and RMSE. This model provides lower estimation error, and the model maintains relatively faster training compared to traditional models which results in faster, scalable prediction accuracy and strong generalization capability in software project planning. Also, the predicted effort is further applied to estimate project budget and create a phase-based SDLC schedule.

Keywords:

PSO, Hyperparameter Tuning, Extreme Learning Machine, Cost

1. INTRODUCTION

Software cost estimation plays an essential role in project management since it helps organizations plan budgets, schedule development activities and allocate resources effectively. Accurate cost estimation can manage risks and resources can be managed efficiently. Modern projects show high complexity and several non-linear relationships and inaccurate estimations lead to budget overruns, schedule delays and inefficient utilization of resources and project planning. Accurate software cost estimation enables effective decision making and management of various risks occurred during the development of the software. Even after decades of research, problems related to software estimation continue because of the growing complexity and dynamic nature of software system [1].

This study focuses on software engineering and machine learning based predictive modelling. Traditional models rely on algorithmic and parameter-based models that use software size and fixed cost drivers. The constructive cost model (COCOMO) represents one of the earliest and most influential models for software cost estimation [4]. COCOMO estimates development effort using Lines of code (LOC) and various complexity factors. Though this model is transparent, it assumes that relationships between cost factors and efforts remain stable, which limits its flexibility for different projects [6]. Several studies have expanded and examined upon the traditional estimation methods through systematic surveys and empirical comparisons [11],[12],[14]. Studies of COCOMO based models show that their accuracy relies on precise size estimation and proper project classification [2]. Researchers have observed that these models work best on historical datasets similar to those used during calibration, but their accuracy declines when used on modern

projects that involve reuse, rapid changing requirements and iterations [7]. These altogether reveal the limitations of static, equations-based estimation models.

To handle these problems, researchers are increasingly using data-driven and analogy-based methods. Analogy-based software cost estimation predicts efforts by comparing a new project to same or likely same past projects [5]. Research indicates that analogy-based models can outperform algorithmic methods when high-quality historical data is available [3]. However, determining the similarity measures and attributes add complexity due to this the analogy-based approach becomes unreliable when the historical data is limited or inconsistent [5].

With progress in computational intelligence, machine learning techniques are becoming more popular in software cost estimation. Neural networks, support vector machines and ensemble learning methods show improved accuracy of the prediction by modelling the non-linear relationships among cost drivers [8]. Fuzzy logic and hybrid intelligent systems enhance flexibility by managing uncertainty in projects attributes [9], [16], [17]. These methods reduce the dependence on fixed assumptions and adapt better to diverse datasets than traditional models.

Despite improved accuracy, current machine learning methods face significant challenges. Many neural-based network models need iterative training, extensive parameter tuning, and substantial computational power, which restricts their scalability [8]. Ensemble methods, while accurate, often lack clarity and depend on large, labelled datasets [9]. Many studies suggested improving accuracy but give little attention to training efficiency and model simplicity [3].

These challenges highlight a clear gap in research: the lack of an estimation framework that combines high prediction accuracy, quick learning speed, low computational cost and efficient implementation. An effective solution should capture complex non-linear relationships in software project data without the drawbacks of iterative learning and effective parameter optimization.

To fill this gap, our work presents an Extreme Learning Machine (ELM)- based framework enhanced with hyperparameter optimization and self-adaptability mechanism for software cost estimation. ELM uses a single hidden layer feed-forward neural network, where hidden parameters are assigned randomly and output weights are assigned randomly and output weights are computed analytically. This approach eliminates the need for iterative backpropagation and efficiently reduces the training time while maintaining strong generalization capabilities [1]. By using ELM, the proposed framework keeps the predictive power of neural models while simplifying training and deployment.

The proposed model stands out from previous models in several important aspects. Unlike COCOMO and analogy-based models, it doesn't depend on fixed coefficients. Compared to traditional neural networks and ensemble methods, it achieves competitive accuracy with much lower computing costs. By applying ELM to benchmark datasets this work demonstrates that efficient learning doesn't need complex architecture.

The main contribution of this study is using Extreme Learning Machine as a practical and effective alternative to both traditional and complex machine learning based models. By balancing accuracy, efficiency and scalability this approach improves software effort estimation and supports more effective decision making in modern and complex software development projects. The framework also consists of a post-processing module to transform predicted effort into project budget and SDLC phase-wise schedule to be practically used.

2. RELATED WORK

Table.1. Summary of Related Works

Authors	Dataset Used	Proposed Model	Results / Findings
Ali Idri et al. [10]	COCOMO'81 dataset (252 projects)	RBF Neural Network using C-Means and APC-III clustering	C-Means based RBFNN achieved MMRE = 29.81% and Pred (25) = 73.81%, outperforming APC-III.
Nevena et al. [13]	COCOMO81, COCOMO2000, NASA60, Kemerer	ANN18 and ANN27 optimized using Taguchi Orthogonal Arrays Method	ANN27 achieved better estimation accuracy with lower MMRE values and faster convergence.
Somya et al. [15]	COCOMO datasets	ANN-enhanced COCOMO model	Improved effort estimation accuracy compared with traditional COCOMO models.
Mohamed et al. [18]	Software project dataset with duration, complexity, and experience attributes	Deep Neural Network with six hidden layers	Achieved average prediction error of 0.38 and standard deviation of 3.35.
Gaurav et al. [19]	NASA software projects dataset	Neuro-COCOMO using Backpropagation Neural Network	Improved software effort and development time estimation.
deBarcelos et al. [20]	COCOMO and ISBSG datasets	ANN and Linear Regression models	ANN outperformed Linear Regression based on MMRE and R ² metrics.
El-Sebakhy [29]	Kemerer, IBM, Hallmark datasets	Functional Network model	Achieved lower MAPE compared with MLP and nonlinear regression models.
Li et al. [30]	Four software project datasets	Neural Network-Based Nonlinear Adjusted	NABE outperformed ANN,

		Analogy-Based Estimation (NABE)	CART, and SWR models.
Ghose et al. [31]	Lopez et al. dataset	Generalized Regression Neural Network (GRNN)	Produced superior MMRE, MdMRE, BRE, and Pred(25) values.
Kaushik et al. [32]	COCOMO and NASA93 datasets	Feed-Forward Backpropagation Neural Network integrated with COCOMO	Hybrid model improved effort prediction accuracy.
Benala et al. [33]	COCOMO81, NASA, and Desharnais datasets	Hybrid FLANN with UKW/DBSCAN clustering	Achieved higher estimation accuracy than conventional FLANN.
Meenaksh et al. [34]	Mutation testing dataset on credit approval application	Backpropagation Neural Network Testing Oracle	Successfully detected faults in mutated software versions.

3. EMPLOYED TECHNIQUES

3.1 EXTREME LEARNING MACHINE (ELM)

Extreme Learning Machine (ELM) is a training designed for single-layer feed-forward neural networks that enables very fast model training and good performance. ELM randomly sets input weights and hidden layer biases and calculates output weights analytically. This method lowers computational complexity and eliminates the iterative training. ELM plays a key role in software estimation because projects attributes often have non-linear relationships that traditional models struggle to capture. ELM also requires less training time and requires fewer adjustments to parameters. The architecture of Extreme Learning Machine consists of three layers:

- **Input layer:** The input layer contains neurons $I_1, I_2, \dots, I_n$, which represent input features. Each neuron sends signals to every neuron in hidden layer. In software cost estimation these inputs include Lines of Code (LOC), Complexity, Cost factors, Team experience, Environment factors.
- **Hidden layer:** The hidden layer contains neurons $J_1, J_2, \dots, J_f$. These weights are not updated iteratively. Here Input weights are randomly assigned. Bias values b_1, b_2, b_3 are randomly generated.

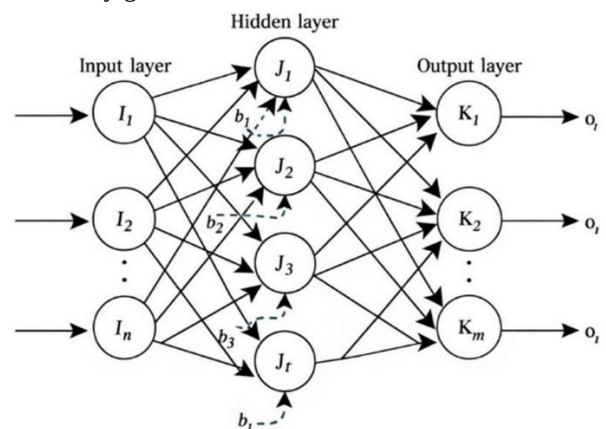


Fig.1. ELM

Activation functions are applied. Hidden layer output matrix is expressed as:

$$H = g(WX + B) \quad (1)$$

where W is input weight matrix, X is input data matrix, B is bias vector of hidden neurons, g is activation function, and H is hidden layer output matrix.

- **Output layer:** The output layer contains neurons $K_1, K_2, \dots, K_m$. The output weights are computed analytically rather than learned through backpropagation. This layer provides the final prediction. The output of ELM is expressed as:

$$H\beta = T \quad (2)$$

where H is hidden layer output matrix, B is out weight matrix, T is target output matrix.

The output weights are calculated using:

$$\beta = H^{-1}T \quad (3)$$

where H^{-1} denotes the Moore-Penrose pseudo-inverse of H

3.2 PARTICLE SWARM OPTIMIZATION (PSO)

It is a subset of natural selection and has the capability of optimization. It is used to solve non-linear and multidimensional problems efficiently [16]. The PSO was proposed by James Kennedy and Russel Ebhart has the nature of stochastic optimization based on social simulation models. PSO basic principle is all particles of swarm travel into the problem domain, learns it from environment and gives better solution. In PSO each particle is treated as a candidate solution in a problem space. The fitness function as an objective function of the problem statement interprets all the candidate solutions.

PSO begins with a set of random particles and then starts to find an optimal solution by updating their generations. The two fitness values help to calculate each particle in problem space in any iteration. Initially the best value of each particle must be memorized as P_{best} . The second, best value is G_{best} is choosing from the all the P_{best} values which is an optimal solution for a given problem. In PSO, the velocity and inertia weight of the particle determines the direction of movement of each particle and the position of the particle.

$$v[i] = w * v[i] + c_1 * rand() * (P_{best}[i] - present[i]) + c_2 * rand() * (G_{best} - present[i]) \quad (5)$$

$$present[i] = present[i] + v[i] \quad (6)$$

where v is velocity of the particle and $present$ is current position (solution). The $rand()$ is a random number between 0 and 1, w is inertia weight, c_1 is “cognitive parameter” and c_2 is “social parameter”. The convergence properties of the algorithm are determined by the combination of the parameters. The Particle velocities on each dimension are clamped to maximum velocity. If the sum of accelerations causes the velocity of the corresponding dimension to exceed a user-specified parameter, then the velocity of that dimension is constrained to [28].

4. PROPOSED METHODOLOGY

The proposed framework is developed to perform software cost estimation using Extreme Learning Machine (ELM) with

hyper parameter optimization and self-adaptive learning. This framework was designed to address the problems caused by overfitting and under fitting, and high training time and commonly observed in traditional neural based methods. Particle Swarm Optimization (PSO) was used in hyper parameter optimization for choosing the correct no of hidden nodes and activation function and feature selection and added Online Sequential Extreme Learning Machine (OSELM) to enable incremental learning without restraining the model from scratch, thereby reducing computational cost and training time.

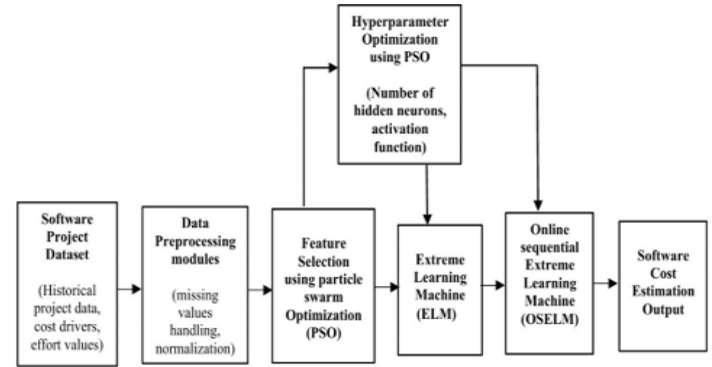


Fig.2. Proposed Architecture

4.1 DATA PREPROCESSING

Prior to model training the dataset was preprocessed to improve data quality and remove noise. Missing values were handled using appropriate statistical techniques and normalization was applied to ensure all input attributes were within common scale. Feature selection was performed using PSO to eliminate irrelevant and redundant features.

4.2 HYPERPARAMETER OPTIMIZATION USING PSO

Particle Swarm Optimization (PSO) is applied to tune the hyperparameters of the proposed ELM model [25]. The number of hidden nodes and the activation function were treated as optimization variables. PSO was selected due to its simplicity and ability to efficiently explore the search space [25],[26]. During optimization, particles iteratively updated their positions based on personal and global best solutions. The fitness of each particle was evaluated using prediction error metrics. Through this process optimal ELM configurations were identified, reducing the risks of overfitting and underfitting while improving estimation accuracy.

4.3 EXTREME LEARNING MACHINE

Extreme Learning Machine is used as a core prediction model in the proposed framework. ELM is a single feed forward neural network in which input weights and hidden layer biases are randomly assigned, and output weights are analytically computed. ELM is faster and takes less time to train.

4.4 SELF ADAPTIVE LEARNING USING OSELM

To further reduce the training time and support dynamic learning, OSELM was introduced into framework. OSELM enabled the model to update its parameters incrementally as new project data become available, instead of retraining the model

from scratch, the existing weights are adjusted using sequential learning.

4.5 EFFORT BASED BUDGET AND TIMELINE GENERATION

Once software effort is predicted by software effort model which is PSO-ELM model, the software estimate is again being used to obtain project budget and development schedule. The budget of the project is estimated using the fixed cost per person-hour. Besides, the overall effort is allocated into established stages of Software Development Life Cycle (SDLC) in a predetermined portion (e.g., Analysis 10%, Design 20%, Implementation 40%, Testing 20%, Deployment 10%).

Pseudocode:

Input: Dataset D

Output: Predicted Effort, RMSE, MMRE, R^2

1. Load dataset D
2. Separate features x and target y
3. Preprocess data:
 - Scale numerical features
 - One-hot encode categorical features
4. Initialize PSO for features selection and hyperparameter tuning
5. For each particle in PSO: Generate binary features mask
Select features X_s
Train temporary ELM
Compute RMSE and MMRE Evaluate fitness
6. Obtain optimal features subset
7. Initialize ELM with random input weights W and bias b
8. Compute hidden layer output $H = g(XW + b)$
9. Compute output weights $\beta = \text{pseudo-inverse}(H) \times y$
10. Train final ELM model
11. Predict effort values
12. Compute RMSE, MMRE, R^2
13. Compute project budget:
 - Budget = Effort $\times$ Cost_per_person_hour.
14. Create phase SDLC distribution:
 - Analysis = 10% of Effort
 - Design = 20% of Effort
 - Implementation = 40% of Effort
 - Testing = 20% of Effort
 - Deployment = 10% of Effort
15. Return: Predicted Effort, RMSE, MMRE, R^2 , Estimated Budget, Phase wise Effort Distribution.

5. RESULTS AND DISCUSSION

5.1 EXPERIMENTAL SETUP

The proposed model was implemented using python language with good hardware configuration for the experimental evaluation was conducted to ensure reliable evaluation, 10-fold cross-

validation was used due to the limited dataset size (81 projects). In each fold, the dataset was split into training and testing subsets. Feature selection and hyperparameter optimization using PSO were performed only on the training data to avoid data leakage. The trained model was then evaluated on the unseen test data. This Extreme Learning Machine (ELM), Hyper parameter optimization using PSO (PSO-ELM) and PSO based feature selection (PSO-FSELM). Particle Swarm Optimization was employed to optimize hidden layer weights, bias parameters, and feature subsets. Model performance was assessed using Root Mean Square Error (RMSE), Mean Magnitude of Relative Error (MMRE) and R^2 Score, which are standard metrics in software cost estimation literature.

5.2 DATASET CHARACTERISTICS

The wide range of effort values introduces variability and noise, making the dataset challenging and suitable for evaluating model robustness. Statistical descriptions were given in Table.2.

Table.2. dataset summary

Attribute	Description
Number of Projects	81
Number of Features	12
Effort Range	~800 to ~24000
Dataset Type	Real-world industrial projects

The results clearly indicate that PSO-ELM significantly outperformed the baseline ELM by reducing RMSE and improving the coefficient of determination. Further improvement was observed when PSO-based feature selection was incorporated, demonstrating the effectiveness of eliminating redundant and noisy features. The results clearly indicate that PSO-FS-ELM shows improved performance compared to baseline models. Due to small datasets the result should be interpreted with caution and feature validation on large datasets is required. The high R^2 is attributed to effective feature selection and hyperparameter tuning using PSO.

Table.3. dataset summary

Actual Effort	Prediction Effort	MRE
5152	5953.839070	0.155636
5635	3978.664595	0.293937
805	442.754183	0.449995
3829	4080.710921	0.065738
2149	3935.164509	0.831161
2821	3342.591546	0.184896
2569	2767.190968	0.077147
3913	3812.863161	0.025591

The proposed model achieves reliable predictions across most projects particularly for medium-scale developments where relative error remains low. Increased relative error in a small number of low-effort projects highlights the sensitivity of MMRE to smaller target values rather than a deficiency in model performance than deficiency in model performance. Besides prediction of efforts, the proposed framework also produces

realistic outputs which include estimation of project budget and generation of timeline of SDLC phases. The proposed effort is approximated to monetary cost, as per the standard cost assumptions. Moreover, the effort is spread over various levels of development of the project to render a systematic implementation plan. This makes the model more usable in the actual project planning and allocation of resources.

Table.4. Results

Model	RMSE	MMRE	R ²
ELM	9276.9	2.6	- 6.5
PSO- ELM	8033.6	3.2	- 4.6
PSO-FS-ELM	1084.0	0.2	0.9

Besides prediction of efforts, the proposed framework also produces realistic outputs which include estimation of project budget and generation of timeline of SDLC phases. The proposed effort is approximated to monetary cost, as per the standard cost assumptions. Moreover, the effort is spread over various levels of development of the project to render a systematic implementation plan. This makes the model more usable in the actual project planning and allocation of resources.

The improved performance of the proposed model can be attributed to effective feature selection and hyper parameter optimization using PSO. By selecting most relevant cost drivers, the model reduces noise and improves generalization. Additionally, optimizing the number of hidden neurons and activation function enables the ELM to better capture non – linear relationships in datasets. The feature selection process identifies importance of the project attributes that significantly influence effort estimation. By focusing on these selected features, the model increase prediction accuracy while avoiding redundant information.

Previously the models were limited to the rigidity, sensitivity to input and high training demands. In contrast, this proposed framework adapts to the changing project characteristics while keeping low computational costs. The project makes a significant impact by enabling accurate, scalable and efficient cost estimation. Moreover, it enables better decision making, reducing the risks and improve the resource planning for the various modern and complex software development. The combination of effort-based budget estimation and timeline generation enhances the feasibility of the model to a more practical use. It gives the project managers a chance to estimate effort, as well as plan resources and development phases.

6. CONCLUSION

In this paper, a PSO -enhanced Extreme Learning Machine (ELM) framework was proposed for software cost estimation. The model improved prediction accuracy through effective feature selection and hyperparameter optimization. The experimental results demonstrate that the proposed approach performs better than baseline models in terms of RMSE, MMRE, R². However, the study is limited by a small dataset size (Desharnais dataset with 81 projects), which may affect the generalizability of the results. Additionally, the use of PSO introduces additional computational cost compared to standard ELM. Therefore, further

validation on larger and diverse datasets is required to confirm the robustness of proposed approach. Budget estimation and SDLC-based generation of timeline are also offered as the practical outputs in the proposed framework, which improves its application in the real-life project planning.

7. FUTURE WORK

The future direction of the work is to test the proposed model using bigger and multiple datasets to enhance generalizability. As the present study uses the Desharnais dataset only, the study should be tested using different datasets to be robust. More studies can be conducted on better method of feature selection and sophisticated optimization methods to improve prediction accuracy. Moreover, it is possible to enhance the efficiency of the model through research of the ways of minimizing the computational overhead caused by PSO. Lastly, use of interpretability techniques can assist in comprehending the effects of various cost drivers to the estimation of software effort. Further research can be conducted on adaptive phase allocation schemes rather than constant percentage allocation of the SDLC phases.

REFERENCES

- [1] B.W. Boehm, "Software Engineering Economics", *IEEE Transactions on Software Engineering*, Vol. 10, No.1, pp. 4-21, 1984.
- [2] K. Lavingia, R. Patel, V. Patel and A. Lavingia, "Software Effort Estimation with Machine Learning Algorithms", *Scalable Computing: Practice and Experience*, Vol. 25, No. 2, pp. 1276-1285, 2024.
- [3] M. Hosni, A. Idri, A.B. Nassif and A. Abran, "Heterogeneous Ensembles of Software Development Efforts Estimation", *Proceedings of International Conference on Soft Computing and Machine Intelligence*, Vol. 8, pp. 174-178, 2020.
- [4] K. Srinivasan and D. Fisher, "Machine Learning Methods to Predict Software Development Effort", *IEEE Transactions on Software Engineering*, Vol. 21, No. 2, pp. 126-137, 1995.
- [5] M. Jorgensen, "A Review of Literature on Expert Assumption of Software Development Effort", *Journal of Systems and Software*, Vol. 70, No. 2, pp. 37-60, 2004.
- [6] L.C. Briand, T. Langley and I. Wiecezorek, "A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques", *Proceedings of International Conference on Software Engineering*, Vol. 87, pp. 377-386, 2000.
- [7] A. Idri, A. Abran and L. Kjiri, "COCOMO Cost Model based on Fuzzy Logic", *Proceedings of International Conference on Fuzzy Theory and Techniques*, Vol. 8, pp. 1-11, 2000.
- [8] E. Kocaguneli, T. Menzies and J.W. Keung, "On the Value of Ensemble Effort Estimation", *IEEE Transactions on Software Engineering*, Vol. 38, No. 6, pp. 1403-1416, 2012.
- [9] G.B. Huang, Q.Y. Zhu and C.K. Siew, "Extreme Learning Machine: Theory and Applications", *Neurocomputing*, Vol. 70, No. 3, pp. 489-501, 2006.
- [10] Idri Ali, Alain Abran and Samir Mbarki, "An Experiment on the Design of Radial Basis Function Neural Networks for Software Cost Estimation", *Proceedings of International*

- Conference on Information and Communication Technologies*, Vol. 1, pp. 1612-1617, 2006.
- [11] F. Akhbardeh and H. Reza, "A Survey of Machine Learning Methods to Software Cost Estimation", *Proceedings of International Conference on Electro Information Technology*, Vol. 20, pp. 405-408, 2021.
- [12] S.M. Chirra and H. Reza, "A Survey on Software Cost Estimation Techniques", *Journal of Software Engineering and Applications*, Vol. 12, No. 6, pp. 1-10, 2019.
- [13] Rankovic Nevena, Dragica Rankovic, Mirjana Ivanovic and Ljubomir Lazic, "A New Approach to Software Effort Estimation using Different Artificial Neural Network Architectures and Taguchi Orthogonal Arrays", *IEEE Access*, Vol. 9, pp. 26926-26936, 2021.
- [14] A. Yasmin, W. Haider, A. Daud and A. Banjar, "Machine Learning based Software Estimation via Development-Centric Feature in Crowdsourcing Platform", *Intelligent Data Analysis*, Vol. 28, No. 1, pp. 299-329, 2024.
- [15] Goyal Somya and Anubha Parashar, "Machine Learning Application to Improve COCOMO Model using Neural Networks", *International Journal of Information Technology and Computer Science*, Vol. 3, pp. 35-51, 2018.
- [16] S. Malathi and S. Sridhar, "Optimization of Fuzzy Analogy in Software Cost Estimation through Linguistic Variables", *Procedia Engineering*, Vol. 38, pp. 177-190, 2012.
- [17] V. Sharma and H.K. Verma, "Optimized Fuzzy Logic-based Framework to Estimate Effort in Development of Software Applications", *IEEE Access*, Vol. 7, pp. 1-11, 2010.
- [18] A. Mohamed Hamada, Abdelrahman Abdallah, Mahmoud Kasem and Mohamed Abokhalil, "Neural Network Estimation Model to Optimize Timing and Schedule of Software Projects", *Proceedings of International Conference on Smart Information Systems and Technologies*, Vol. 8, pp. 1-7, 20219.
- [19] G. Senthil Kumar, "Automation of Software Cost Estimation using Neural Network Technique", *International Journal of Computer Applications*, Vol. 98, No. 20, pp. 11-17, 2014.
- [20] De Barcelos Tronto, Iris Fabiana, Jose Demisio Simoes Da Silva and Nilson Sant Anna, "Comparison of Artificial Neural Network and Regression Models in Software Effort Estimation", *Proceedings of International Conference on Neural Networks*, Vol. 89, pp. 771-776, 2007.
- [21] A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino and E. Mendes, "Investigating the Application of Support Vector Regression to Web Effort Estimation", *Empirical Software Engineering*, Vol. 16, No. 2, pp. 211-243, 2011.
- [22] D.S. Senevirathne and T.K. Wijayasiriwardhane, "Extending Use-Case Point-based Software Effort Estimation: Open Source Freelance Software Development", *Proceedings of International Conference on Smart Computing and Systems Engineering*, Vol. 7, pp. 188-194, 2020.
- [23] M. Ahmed, N. Iqbal, F. Hussain, M.A. Khan, M. Helfert and J. Kim, "Blockchain-based Software Effort Estimation: An Empirical Study", *IEEE Access*, Vol. 10, pp. 120412-120425, 2022.
- [24] V. Venkataiah, R. Mohanty, J.S. Pahariya and M. Nagaratna, "Application of Ant Colony Optimization Techniques to Predict Software Cost Estimation", *Computer Communication, Networking and Internet Security*, Vol. 8, pp. 315-325, 2017.
- [25] V. Venkataiah, M. Nagaratna and R. Mohanty, "Application of Chaotic Increasing Linear Inertia Weight and Diversity Enhanced Particle Swarm Optimization, Software Cost Estimation", *International Journal of Electrical and Electronics Research*, Vol. 10, No. 2, pp. 154-160, 2022.
- [26] V. Venkataiah, M. Nagaratna and R. Mohanty, "Integration of Diversity Enhancement of Particle Swarm Optimization and Neighbourhood Search of Software Cost Estimation", *International Journal of Intelligent Systems and Applications in Engineering*, Vol. 10, pp. 348-362, 2022.
- [27] V. Kumari, "Deep Learning Paradigm and its Bias Characterized by the Coronary Artery Wall Segmentation in Intravascular Ultrasound Scans", *Journal of Cardiovascular Development and Disease*, Vol. 10, No. 12, pp. 1-11, 2023.
- [28] R.D. Reddy, S.A. Reddy, M. Kotha and N. Suresh, "A Fast Evolvment in the Banking Sector in Terms of Convolutional Neural Network Signature Verification", *Applications of Mathematics in Science and Technology*, Vol. 23, pp. 259-265, 2025.
- [29] El-Sebakhy and A. Emad, "Functional Networks as a Novel Data Mining Paradigm in Forecasting Software Development Efforts", *Expert Systems with Applications*, Vol. 38, No. 3, pp. 2187-2194, 2011.
- [30] Li, Yan-Fu, Min Xie and T.N. Goh, "A Study of the Non-Linear Adjustment for Analogy based Software Cost Estimation", *Empirical Software Engineering*, Vol. 14, No. 6, pp. 603-643, 2009.
- [31] Ghose, Mrinal Kanti, Roheet Bhatnagar and Vandana Bhattacharjee, "Comparing Some Neural Network Models for Software Development Effort Prediction", *Proceedings of International Conference on Emerging Trends and Applications in Computer Science*, Vol. 8, pp. 1-4, 2011.
- [32] Anupama Kaushik, Ashish Chauhan, Deepak Mittal and Sachin Gupta, "COCOMO Estimates using Neural Networks", *International Journal of Intelligent Systems and Applications*, Vol. 4, No. 9, pp. 22-28, 2012.
- [33] Benala, Tirimula Rao, Rajib Mall, Satchidanada Dehuri and Koradda Chinna Babu, "Software Effort Prediction using Unsupervised Learning (Clustering) and Functional Link Artificial Neural Networks", *World Congress on Information and Communication Technologies*, Vol. 12, pp. 115-120, 2012.