Journal of Intelligent Systems and Internet of Things
Vol. 15, No. 02, PP. 29-40, 2025
DOI: https://doi.org/10.54216/JISIoT.150203

Modelling Software Development Effort Using Data-Driven Models

Zainab Rustum Mohsin[1,*], Firoj Khan[2]
[1]College of Computer Science and Mathematics, University of Thi-Qar, Thi-Qar, Iraq
[2]PGT(IT), Navodaya Leadership Institute, South Goa, India
Emails: zainabrustum@utq.edu.iq; khanfiroj.01@gmail.com

Abstract

Software effort estimation is highly significant for project management regarding the bidding process since underestimation leads to financial losses, while overestimation may bring the chance of losing a competitive bid. Whereas numerous models have been designed up until now, those developed upon machine learning, Adaptive Neuro-Fuzzy Inference Systems (ANFIS) and Artificial Neural Networks (ANN) have emerged as preeminent technologies. The proposed research will explore the effectiveness of using the ANN and ANFIS approaches in the estimation of effort for NASA datasets by 13 observations used for training and the rest for the test. To check the precision of models, several measures are used to evaluate the accuracy of the developed model, including the correlation coefficient, RMSE, and MMRE. The findings demonstrate that ANN and ANFIS exhibit superior performance, yielding much higher prediction accuracy compared to conventional Models including Walston-Felix, Doty, Bailey-Basili, and Halstead. It emphasizes ANN and ANFIS as reliable and straightforward software effort estimating methodologies, hence yielding significant enhancements in estimation precision and competitiveness. Their high performance underlines their usefulness to project managers who seek accurate predictions. This study strongly recommends the application of data-driven approaches like ANN and ANFIS to enhance the overall estimation accuracy in software project bidding.

Received: September 20, 2024 Revised: November 15, 2024 Accepted: January 06, 2025

Keywords: NASA; Data-Driven Models; ANN; ANFIS; Software Effort

1. Introduction

Accurate effort estimation in software development is a rudimentary objective of software project management, because it ensures effective roles in the planning and resource allocation for the projects. Similarly, critical to this is an assessment of the accuracy and dependability of the models for estimations, because this would provide the most valuable clue to the software engineering fraternity. Estimating in software development involves predicting resources, time, and effort in the development or maintenance of a software application. Estimation then forms the basis for an effective project management and decision-making process. Proper estimation may involve accurate forecasting of software costs in planning, budgeting, controlling, and ultimately efficient project management. Additionally, estimating effort might be benefit in predicting the resources that will be needed in the future [1]. During designing and developing software, accurate cost and time prediction are crucial for reasonable resource allocation and planning. The success of a project heavily relies on effective planning, since this stage entails estimating the time and financial restrictions needed to finish [2]. Inaccurate effort estimation can have significant consequences: underestimating may lead to delays and budget overruns, potentially causing project failure, while overestimating could result in inefficient resource allocation, adversely affecting other critical projects [3]. Consequently, precise cost estimation is essential to ensure project success.

In the last four decades, researchers have developed various models for estimating software project expenses. These models encompass Boehm's COCOMO models and several empirical methodologies [4] to the methods of analysis used in research [5-7]. Empirical methods rely on data from past projects to assess the current one, obtaining basic formulas by an examination of the database at available. On the other hand, the analytic model

employs formulas based on global assumptions, like the developer's rate of problem solving and the number of available problems [8]. The project manager and development team should conceptualize and support a good software cost estimate. Numerous models were investigated to provide better effort estimation [9-12]. The following are examples of major models, which are frequently used as standards to gauge software effort:

Walston-Felix model.

Doty model (for KLOC > 9).

Bailey-Basili model.

Halstead model.

To explore the relationship between project size and development effort, these models were constructed by analyzing a vast collection of completed software projects spanning various applications and organizations. However, these methods remain incapable of accurately estimate software effort.

According to more recent research, two of the most important data-driven methods for estimating software effort are Artificial Neural Networks (ANNs) and Adaptive Neural-Based Fuzzy Inference System (ANFIS) [13-17]. Other than traditional software effort estimation methods. Among the key merits of the proposed Artificial Intelligence (AI) techniques is their enhanced capability to learn from examples for better generalization, resulting in reduced prediction errors. This research applies to these state-of-the-art AI-driven methodologies, ANN and ANFIS, for estimating software effort. Using the NASA dataset for both training and testing, a proposed model showed excellent predictability. Comparing proposed models to established models presented in the literature further validated this research through strong consistency from previously presented findings.

This paper is structured as follows: Section 2 presents an overview of pertinent studies that have used various data-driven methodologies for effort estimate in software development. The formulae presently used for effort calculation are delineated in Section 3. Section 4 describes the technical specifics of approaches utilizing ANN and ANFIS. The dataset used while conducting this study is specified in Section 5. Section 6 lists the various evaluation metrics that were considered to apply to the proposed models. Section 7 discusses, in detail, the experiments conducted with their results. Section 8 concludes this paper by summary of major findings and implications.

2. Literature Review

Accurate and reliable estimation of development effort is crucial for effective software project management. Despite numerous studies conducted over the past decades, predicting software effort remains a challenge for the software community. Over time, researchers have introduced a variety of techniques to address this issue.

Sandhu, et al. [8] performed a broad study using datasets from NASA to predict effort in software development, facilitated by two advanced data-driven models: Neuro-Fuzzy Inference Systems and Fuzzy-GA. Their analysis indicated the NF model to be a lot more accurate compared to other methodologies. On a related note, Mohsin [14] tested the performance of an ANN model against various statistical regression techniques based on the COCOMO'81 database that included 63 software projects. The metrics of MMRE and R were used for the comparison, and it was determined that the ANN model outperformed all the regression models and established it as reliable and accurate for the purpose of effort estimation in software projects. Detailed research on ISBSG datasets by using four different types of ANNs: Multi-Layer Perceptron (MLP), General Regression Neural Networks (GRNN), Cascade Correlation Neural Networks (CCNN), and Radial Basis Function Neural Networks (RBFNN), was performed by Nassif et al. [16] From the comprehensive performance evaluation concerning multiple performance criteria, the best model was Cascade Correlation Neural Networks. Mohsin [13] suggests a model for estimating software effort using ANFIS approach. The same was trained and validated using the comprehensive database called the COCOMO dataset consisting of data from 63 projects published in 1981. Performance assessment was evaluated with the key metrics, MMRE, and R. The findings demonstrated that the performance of ANFIS outperformed other traditional models, including FPA, Basic COCOMO, and SLIM in terms of robustness and performance for the effort estimation. Nanda et al., [19] proposed a more efficient model for software development effort estimation by integrating Particle Swarm Optimization (PSO) with the Adaptive Neuro-Fuzzy Inference System technique (ANFIS). In the study, the authors utilized the NASA dataset and subsequently optimized 17 cost driver parameters with the PSO algorithm. The optimized parameters improved the accuracy of predictions when the Neuro-Fuzzy prediction model is trained using it as an input. The results showed that the hybrid PSO-ANFIS could yield significant improvements in accuracy and, therefore, can be considered as a robust and effective method for software development effort estimation. Mohsin [20] utilized both Adaptive Neuro-Fuzzy Inference System (ANFIS) and Artificial Neural Network (ANN) models to predict software development effort. The forecasts from these soft computing models were juxtaposed with those of regression models. The COCOMO dataset, comprising 16 input variables and one output variable, was employed

to develop the models. While both models successfully predicted software effort, ANFIS outperformed ANN based on two statistical indicators: MMRE and R. Kamal *et al*., [21] utilized a fuzzy logic methodology to forecast software development effort. The linguistic fuzzy values in the COCOMO II model were represented by the triangle membership function. Using measures like Mean Magnitude of Relative Error (MMRE), Variance Accounted for (VAF), and Prediction (PRED), the accuracy of the suggested model was assessed and contrasted with regression models. The results demonstrated that the fuzzy logic method outperformed other approaches in modeling software development effort. Edinson and Muthuraj [22] explored three Artificial Intelligence (AI) techniques for software effort estimation: Elman Neural Networks (Elman NN), and two Adaptive Neuro-Fuzzy Inference System (ANFIS) variants—ANFIS with Fuzzy C-Means clustering (ANFIS-FCM) and ANFIS with Subtractive Clustering (ANFIS-SC). The study utilized multiple datasets, including the COCOMO dataset, Desharnais dataset, Maxwell dataset, and IKH dataset, which integrates data from the IBM, Kemerer, and Hallmark databases. Comparative analysis of prediction results revealed that the ANFIS-FCM model had the lowest prediction error, outperforming the other AI models in estimating software development effort.

The primary objective of this study is to identify the most effective approach for quantifying software development effort. Subsequently, two distinct data-driven approaches (i.e., ANFIS and ANN) were examined to formulate the suggested models. Three performance indicators were used to assess the suggested models.

3. Existing Models Used for Effort Calculation

Several models have been developed to estimate the effort of software development: Walston-Felix, Doty, Bailey-Basili, and Halstead. All these models are based on mathematical equations that estimate the effort parameters. Their formulation is briefly summarized in Table 1.

Table 1: The considered equations

Model Name	Equation
Walston-Felix model	$\text{Effort} = 5.2(\text{KLOC})^{0.91}$
Doty model	$\text{Effort} = 5.288\ (\text{KLOC})^{1.047}$
Bailey-Basili model	$\text{Effort} = 5.5 + 0.73(\text{KLOC})^{1.16}$
Halstead model	$\text{Effort} = 0.7(\text{KLOC})^{1.5}$

4. Prediction Methods

4.1. ANN

A neural network is a nonlinear computation framework consisting of a number of interconnected processing elements or nodes, also called artificial neurons, in a structure inspired by the biological nervous system [23]. as depicted in Figure 1, The architecture of ANN, basically consists of the following architectural components: input parameters, weights, transfer functions, activation functions, thresholds, and outputs.

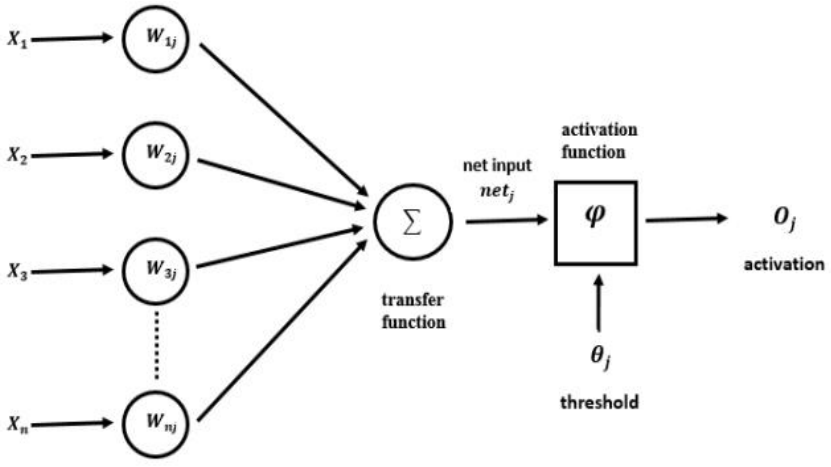

Figure 1. Typical neural network topology [24].

Thus, an artificial neuron works according to different multiple inputs that are given with appropriate weights and calculate a weighted sum which, once the transfer functions are applied, returns their outward value [25], Equation 1 defines thus the operation taking place.

$$\text{Total activation:} \quad net\ j = \sum_{i=1}^{n}(w_{ij})x_i - \mathrm{T}_j \qquad (1)$$

In this context *j* represents the node number, *n* is the total number of nodes, w_{ij} denotes the weight of the connection between nodes. where T_j is the bias given to the node and x_i is the input variable.

The overall activation is dependent on the amount of the internal threshold T_j. If T_j is positive or large, indicating a high internal threshold for the neuron, which inhibit node-firing. Conversely, if T_j is negative or zero, then the neuron has a low internal threshold, and this low value of T_j excites node-firing [25]. After that, a transfer function modifies this activity to produce the final output of the node using the equation below [25]:

$$x_j = \varphi(u_j - \theta_j) \qquad (2)$$

Where x_j denotes the desired output, θ_j denotes the j node's bias term. [26]. The most used activation function (logistic sigmoid) is:

$$\varphi_x = \frac{1}{1+\exp(-x)} \qquad (3)$$

In this research, the feedforward backpropagation algorithm was used for developing the ANN model. Three distinct layers make up an ANN's design: an input layer that records input parameters; one or more hidden layers where computations take place; and an output layer with a single node that provides a software development effort forecast. Experimental findings indicated that a neural network comprising two hidden layers surpassed those with a single hidden layer. Specifically, the optimal ANN structure was achieved with two hidden layers—the first consisting of six nodes and the second with one node—resulting in the lowest Root Mean Square Error (RMSE). Figure 2 illustrates the schematic diagram for the ANN model. Two variables, representing the KLOC and Methodology, were chosen as input parameters for this study. Training a neural network means that the connection weights should be adjusted to minimize an error between the model's forecasted results and the real values. This will, therefore, ensure that an acceptable accuracy level is met. The number of hidden layers, the number of neurons per layer, the choice of activation functions, and the normalization of input data are important factors to take into account while optimizing models. These parameters must, therefore, be put under careful settings to improve the network performance. Once the errors are minimized, the model is evaluated using a separate test dataset, distinct from the training data, to assess its predictive capability. Upon completion of the training process, the neural network produces a model that can reliably predict the desired output for given input patterns.

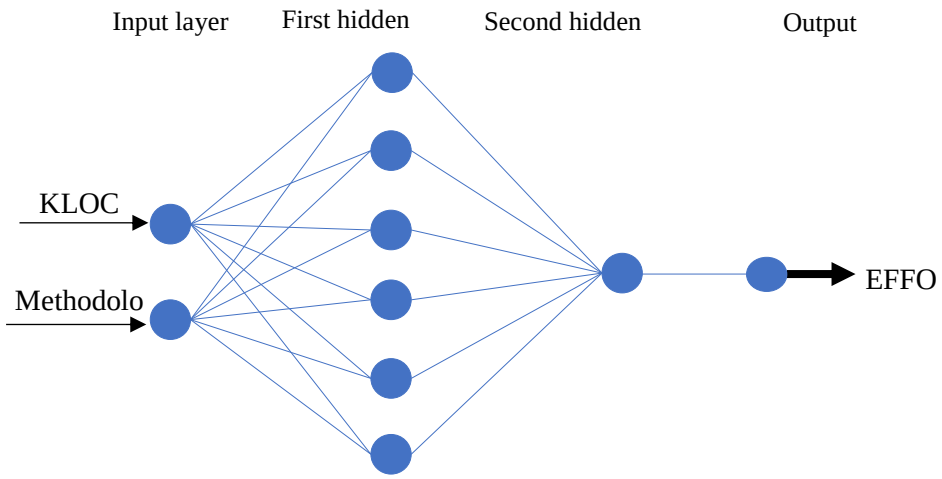

Figure 2. The architecture of the proposed ANN model.

4.2. ANFIS

The ANFIS, introduced by Jang [28], is a widely used neuro-fuzzy model that integrates a fuzzy system within an adaptive network framework. Neural networks excel in learning mappings between inputs and outputs with a self-organizing structure, while fuzzy logic offers an effective means to represent system behavior using fuzzy IF-THEN rules [29]. Each node in the five levels that make up ANFIS performs a distinct node function on the incoming signals [28]. Circular and square nodes are used to denote different adaptive capabilities. Adaptive learning parameters are improved by incremental learning rules to establish the appropriate correlation between input and output properties.

In first order Sugeno fuzzy inference systems with two inputs, x_1 and x_2 and an output, *y*, a typical rule base will have two IF-THEN statements [30] Given below is the form used for these rules:

Rule 1: If x_1 *is* A_1 and x_2 *is* B_1. then $y_1 = f_1 = p_1x_1 + q_1x_2 + r_1$

Rule 2: If x_1 *is* A_2 and x_2 *is* B_2. then $y_2 = f_2 = p_2x_1 + q_2x_2 + r_2$.

In this context, A_i and B_i are fuzzy set linguistic labels, and p_1, q_1, r_1, p_2, q_2, r_2 are parameters defining the output functions. These parameters of the model are determined and tuned through the training and testing phases. The architecture of the ANFIS is shown in Figure 3.

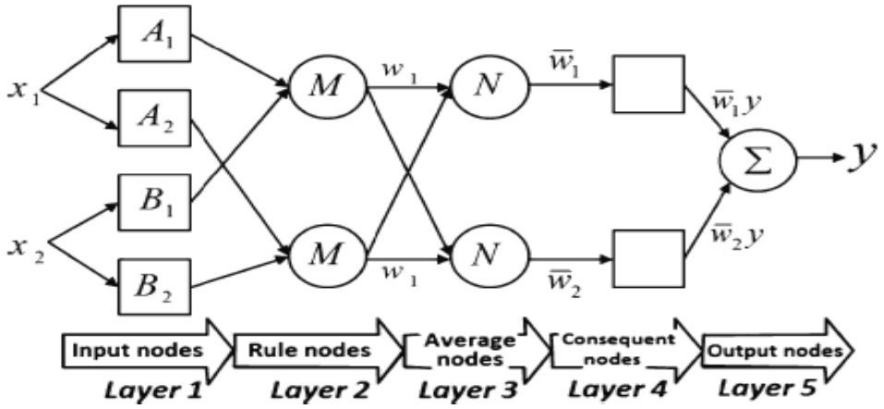

Figure 3. The general architecture of ANFIS technique [28].

The following are explanations of each layer's properties in the architecture:

Input nodes (Layer 1): The node function for each node in this layer is calculated as follows:

$$O_i^1 = \mu_{A_i}(x) \qquad i = 1,2 \qquad (4)$$

Within this architecture, x signifies the input to node *i*, A_i denotes the linguistic label linked to node *i*, and μ_{Ai} indicates the membership function of the linguistic label A*i*, Premise parameters are the usual term used to describe the parameters in this layer.

Rule Nodes (Layer 2): The nodes in this layer are fixed and labeled as M, as illustrated in Figure 3. The outputs of this layer are denoted as O_i^2 are computed as a product of incoming signals transmitted from Layer 1:

$$O_i^2 = w_i = \mu_{Ai}(x)\,\mu_{Bi}(x). \qquad i = 1,2 \qquad (5)$$

Average Nodes Layer 3: The nodes in this layer are static and designated as N (Figure 3). Each node in this layer, particularly the *ith* node, computes the ratio of the *ith* rule's firing strength to the aggregate firing strength of all active rules. The computation that follows is as follows:

$$O_i^3 = \bar{w}_i = \frac{w_i}{w_1 + w_2}. \qquad i = 1,2 \qquad (6)$$

Consequent Nodes Layer 4: Each Node *i* in this layer acts as an adaptive node. Such nodes can be defined to realize mainly:

$$O_i^4 = \bar{w}_i f_i = \bar{w}_i (p_i X_1 + q_i X_2 + r_i), \qquad i = 1,2 \qquad (7)$$

where $\overline{w}_i$ is the output of layer 3, and $\{p_i \,.\, q_i . r_i\}$ is the consequent parameter .

Output Nodes (Layer 5): This layer includes a single node, named as Σ, it represents the node that is to compute the overall output of the ANFIS model. The node function aggregates the contributions from the preceding layers to generate the final output:

$$O_i^5 = \sum_i \overline{w}_i \, y_i = \frac{\sum_i \overline{w}_i \, y_i}{\sum_i \overline{w}_i} \qquad i = 1,2 \qquad (8)$$

More information on the ANFIS algorithm and its mathematical foundation can be found in.[28].

5. Data Description

In this study, NASA [31] dataset have been used to develop the ANFIS and ANN models for effort estimation. The dataset utilized comprises two independent variables: created lines and methods, and one dependent variable: effort. Developed lines are quantified in KLOC, while effort is quantified in person-months. Table 2 displays the dataset for the 18 projects that are used for constructing the proposed models. These data were randomly divided into two groups: a training set comprising 13 projects, and a testing set containing 5 projects.

Table 2: Dataset of NASA Software Projects [31]

Project Number	KLOC	Methodology	Actual Effort
	46.2	20	96
	90.2	30	115.8
	46.5	19	79
	31.1	35	39.6
	12.8	26	18.9
	3.1	26	7
	67.5	29	98.4
	2.1	28	5
	21.5	31	28.5
	4.2	19	9
	54.5	20	90.8
	7.8	31	7.3
	10.5	34	10.3
	5	29	8.4
	78.6	35	98.7
	9.7	27	15.6
	12.5	27	23.9
	100.8	34	138.3

6. Performance Criteria

a. MMRE represents the average of absolute percentage errors, serving as a key metric for assessing prediction accuracy [32].

$$MMRE = \frac{1}{N}\sum_1^N \frac{|\text{Actual Effort} - \text{Predicted Effort}|}{\text{Actual Effort}} \qquad (9)$$

where N denotes the total number of software projects.

b. The *R* quantifies the strength and direction of the linear association between real and estimated values. The range is from -1 to 1, with values approaching 1 signifying a more robust positive correlation and enhanced model performance.

$$R = 1 - \sqrt{\frac{\sum_{i=1}^{N}(\text{Actual Effort} - \text{Predicted Effort})^2}{\sum_{i=1}^{N}(\text{Actual Effort})^2}} \quad (10)$$

Where N representing the total count of input variables.

c. RMSE is a commonly used metric for evaluating a model's accuracy in predicting numerical data.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(\text{Actual Effort} - \text{Predicted Effort})^2}{N}} \quad (11)$$

Where N is the total number of input parameters.

7. Results and Discussion

7.1 Generation of the ANN model

In this study, ANN were used to estimate software development effort using a feedforward backpropagation technique. The Neural Network Toolbox in MATLAB was used to create the neural network model [33] The output layer used a linear activation function, whereas the input and hidden layers utilized a logistic sigmoid activation function. To facilitate model training, it was essential to meticulously calibrate many key parameters: The quantity of concealed layers, the aggregate number of nodes within each concealed layer, and the count of training epochs. Root means square error RMSE was the metric utilized as a yardstick in training and testing to ensure the predictive accuracy of the ANN model. Experimental results demonstrated that this network with two hidden layers outperformed its single-layer counterpart considerably. Through a process of trial and error, the best hidden layer architecture was determined, where the number of nodes varied from 1 to a total of 10. The configuration of six neurons in the first hidden layer and one neuron in the second hidden layer yielded the lowest RMSE. The ANN (2-6-1-1) model was identified as the most appropriate for effort prediction. The network was trained iteratively; weights were updated until the error goal of 0.00312 was reached after 132 epochs. Table 3 summarizes the predicted values of the test dataset that used different models for effort estimation. Figure 4 shows a comparison of predicted vs actual effort during the testing phase, which proves that the current ANN model is very accurate. Similarly, Figure 5 represents a one-to-one performance comparison of actual and predicted efforts. From this figure, it has been observed that the proposed ANN model fits the target data very closely. Further analysis of the test set has yielded an RMSE of 4.457, a correlation coefficient (R) of 0.996, and an MMRE of 0.0903, further establishing the ANN model as of very high predictive accuracy. The strength and reliability of the suggested ANN configuration for software work estimation are highlighted by these results.

Table 3: Actual and predicted effort using different effort estimation methods.

Project ID.	Actual	ANFIS	ANN	Walston-Felix model	Doty model	Bailey-Basili model	Halstead model
14	8.4	5.49	7.39	22.494	28.518	10.222	7.826
15	98.7	104.6	106.7	275.955	510.269	120.849	487.789
16	15.6	16.21	15.69	41.112	57.074	15.685	21.147
17	23.9	19.12	18.4	51.784	74.431	19.169	30.936
18	138.3	129.6	136.3	346.061	662.086	159.435	708.417

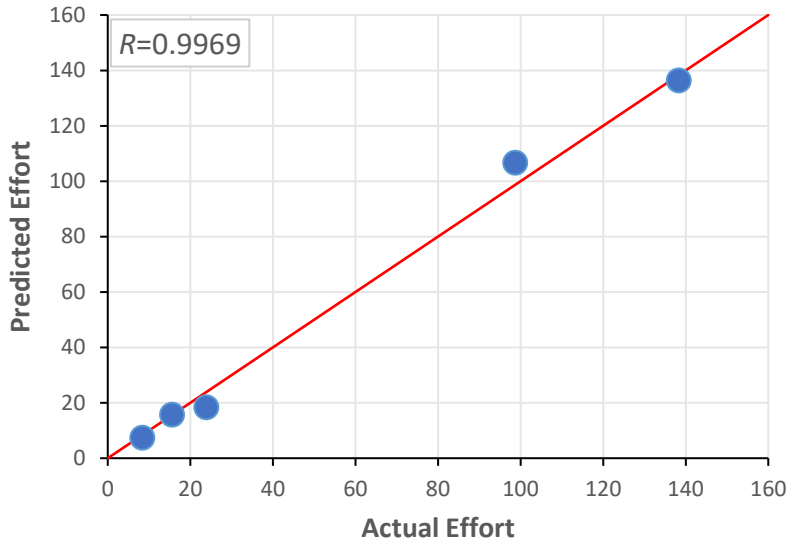

Figure 4. ANN predictions vs. actual (target) results for the test data.

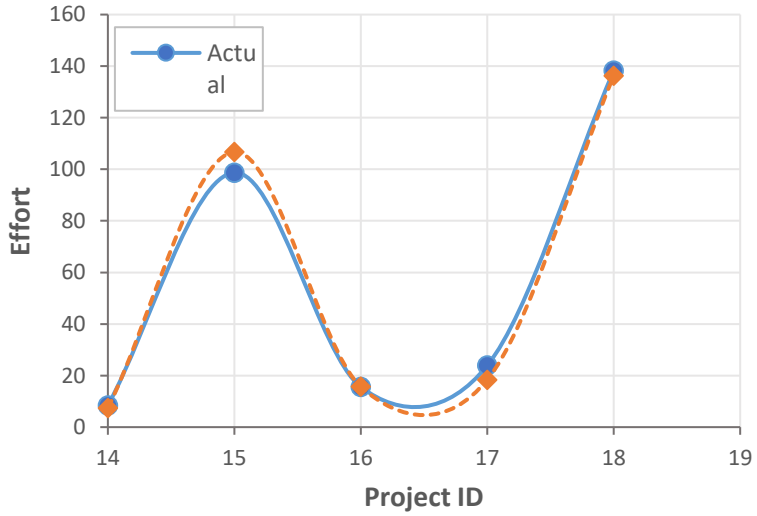

Figure 5. A comparison of the proposed ANN model for the test data with the actual (target) effort.

7.2 Generation of the ANFIS Model

In this work, the model based on ANFIS was developed in MATLAB R2013a. The proposed ANFIS was developed using the *genfis2* function of MATLAB, which is supported by the subtractive clustering technique as introduced by Chiu. This function allows an easy way of creating an initial rule base by arranging any number of input-output data in clusters. Every data point may be a possible center for the first cluster, according to the theory of subtractive clustering. Once identified, all points lying within this radial distance from the center of this cluster are eliminated. The algorithm repeatedly computes the successive centers of clusters by iterating through the same process and guarantees that every data point within a given radius of each cluster center is assigned to the appropriate cluster [35]. Each cluster center's degree of effect across all data dimensions is determined by the r. The larger the radius, the fewer the number of clusters, turning this model more generalized and coarse; the smaller the radius, the greater the number of clusters, it allows the model to be more detailed and fine [36]. Among the popular fuzzy inference systems, The most widely used and effective models in numerous applications are the Mamdani and Sugeno models [37, 38]. The suggested ANFIS model in this research has been developed using the Sugeno-type fuzzy inference system. The subtractive clustering method was used to build the suggested model, which was then trained on 13 NASA datasets and tested on 11 additional test datasets. This model gives the best performance with a hybrid learning methodology after trying an extensive set of learning algorithms and numbers

of epochs. This approach has integrated the backpropagation algorithm, which fine-tunes the premise parameters, while the least squares is used to adjust consequent parameters. Gaussian membership functions for the input variables and linear membership functions for the output variable were used in this context in the proposed ANFIS model. The best configuration under optimization was obtained with a cluster radius of r = 0.22 and 50 training epochs. This setting outperformed the other configurations; It was chosen as the best model as a result. After 50 epochs of testing, the optimal ANFIS model yielded a final error of 5.36, demonstrating its effectiveness in software effort estimation. The predicted values for each testing dataset are presented in Table 3, while Figure 6 shows a comparison between the actual effort made during the testing period and the anticipated outcomes of the suggested ANFIS model. The *R* is 0.995, indicates that the model demonstrated high predictive accuracy. A thorough comparison between the actual effort levels and the forecast accuracy of the suggested ANFIS model is shown in Figure 7. The results demonstrate a strong alignment between the actual and anticipated datasets, highlighting the model's excellent effort estimation, accuracy and dependability. Overall, the ANFIS model performed satisfactorily in modeling software effort, with the evaluation metrics showing an RMSE of 5.36, R of 0.995, and an MMRE of 0.142.

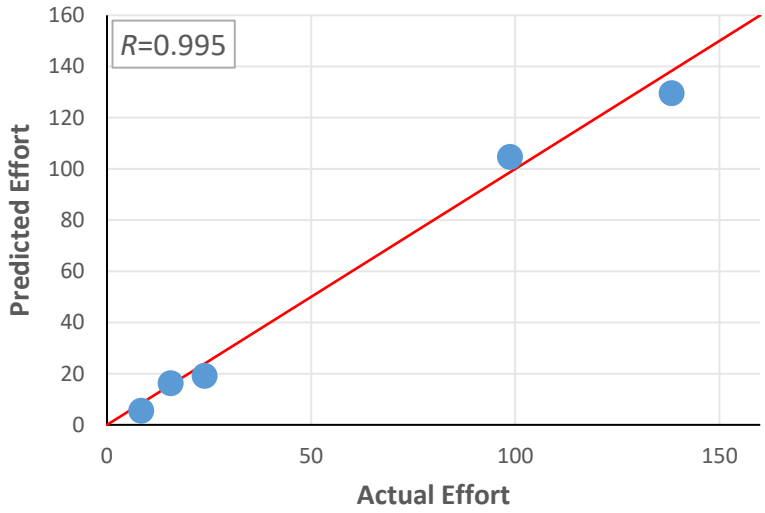

Figure 6. ANFIS predictions vs. actual (target) results for the test data.

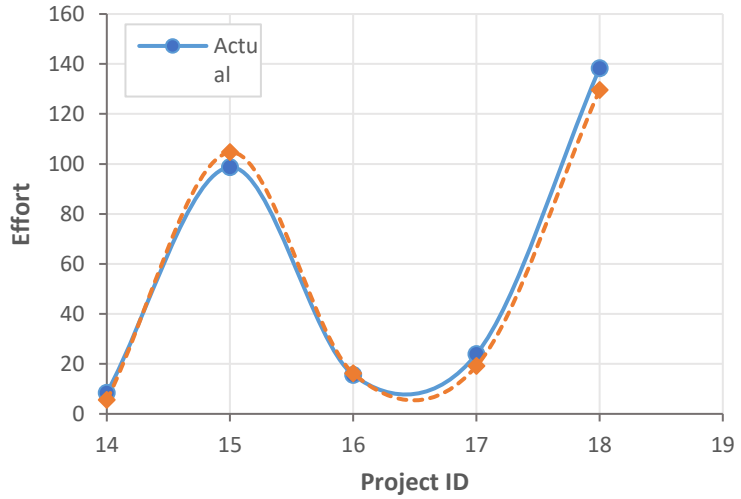

Figure 7. A direct comparison between the actual (target) effort values and the predictions of the proposed ANFIS model for the test dataset.

7.3 Comparison of Techniques

Figure 8 demonstrates the estimated values versus the actual values of software effort from ANN, ANFIS, and those obtained from four existing methods for the testing data sets. Predictions by ANFIS for testing are close to those predicted by ANN (ANN is slightly better than ANFIS). With careful investigation of the Figure 8, It is evident that the ANFIS and ANN models outperform the other four models currently in use. To further support this conclusion, statistical measurements like RMSE and MMRE were computed and are presented in Table 4. According to Table 4, although ANN performed slightly better than ANFIS, both ANN and ANFIS achieved the lowest values of RMSE and MMRE compared to the other approaches. It can be said that out of all the models, the data-driven models in this study perform the best. In fact, it is evident that both ANN and ANFIS produce a higher accuracy compared with the existing models and thus can serve as simple and reliable tools to estimate the software development effort.

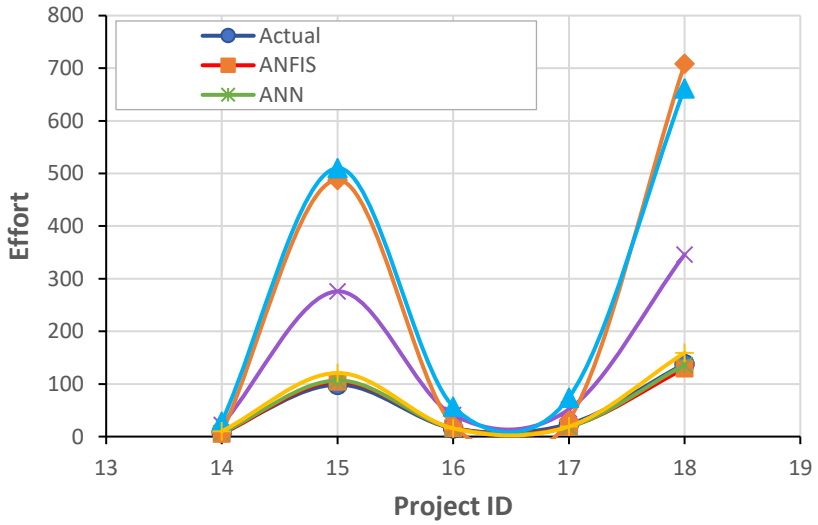

Figure 8. Difference among the test dataset's predicted predictions and actual effort values.

Table 4: Comparison of performance measured for the different models.

Performance Criteria	Model Used					
	ANFIS	**ANN**	**Walston-Felix model**	**Doty model**	**Bailey-Basili model**	**Halstead model**
RMSE	5.36	4.457	123.708	299.47	13.878	308.708
MMRE	0.142	0.0903	1.556	3.025	0.16	1.757

8. Conclusion

In this study, two data-driven approaches, ANN and ANFIS, were developed to predict software project effort. The proposed models used methodology and KLOC (thousands of lines of code) as input variables, with effort as the target output. A dataset of 18 projects from NASA software projects was utilized, with 5 initiatives for testing and 13 to training. Three variables were applied to measure the models' effectiveness: R, RMSE, and MMRE for ANFIS. After testing various structures, the optimal model was identified using a hybrid-learning algorithm with a cluster radius (r) of 0.22 and 50 epochs. The ANN model's optimal structure was determined through a trial-and-error process, resulting in ANN (2-6-1-1) being selected. Both models were further compared to traditional effort estimation methods, including the Walston-Felix, Doty, Bailey-Basili, and Halstead models. The results demonstrated that ANN and ANFIS significantly outperformed these traditional methods in effort estimation. Between the two, the ANN model produced slightly more accurate predictions than the ANFIS model. This study confirmed that both ANN and ANFIS are simple yet reliable tools for estimating software development effort.

References

[1] S. McConnell, *Rapid Development: Taming Wild Software Schedules*. Pearson Education, 1996.

[2] A. Idri and I. Abnane, "Fuzzy analogy-based effort estimation: An empirical comparative study," in *Proc. 2017 IEEE Int. Conf. on Computer and Information Technology (CIT)*, 2017, pp. 114–121.

[3] S. H. S. Moosavi and V. K. Bardsiri, "Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation," *Eng. Appl. of Artif. Intell.*, vol. 60, pp. 1–15, 2017.

[4] B. Boehm, *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[5] G. Cantone, A. Cimitile, and U. De Carlini, "A comparison of models for software cost estimation and management of software projects," in *Computer Systems: Performance and Simulation*, Elsevier, 1986.

[6] L. H. Putnam, "A general empirical solution to the macro software sizing and estimating problem," *IEEE Trans. Softw. Eng.*, vol. SE-4, pp. 345–361, 1978.

[7] F. N. Parr, "An alternative to the Rayleigh curve model for software development effort," *IEEE Trans. Softw. Eng.*, vol. SE-6, pp. 291–296, 1980.

[8] P. S. Sandhu, M. Prashar, P. Bassi, and A. Bisht, "A model for estimation of efforts in development of software systems," *Int. J. Comput. Syst. Eng.*, vol. 3, no. 8, pp. 1931–1935, 2009.

[9] S. Devnani-Chulani, "Modeling software defect introduction," in *Proc. California Softw. Symp.*, 1999.

[10] B. Clark, S. Devnani-Chulani, and B. Boehm, "Calibrating the COCOMO II post-architecture model," in *Proc. 20th Int. Conf. Softw. Eng.*, 1998, pp. 477–480.

[11] A. M. R. Ahmed, M. A. Ali, N. Ahmed, M. F. B. Zamal, and F. M. J. M. Shamrat, "Software defect prediction with Bayesian approaches," *Mathematics*, vol. 11, no. 11, pp. 2524, 2022.

[12] R. E. D. Reverter, L. L. D. G. Nunes, and P. M. C. K. L. Nunes, "Software defect prediction using Bayesian networks," *Empirical Softw. Eng.*, vol. 19, no. 2, pp. 419–456, 2014.

[13] Z. R. Mohsin, "Investigating the use of an adaptive neuro-fuzzy inference system in software development effort estimation," *Iraqi J. Comput. Sci. Math.*, vol. 2, no. 2, pp. 18–24, 2021.

[14] Z. R. Mohsin, "Application of artificial neural networks in prediction of software development effort," *Turkish J. Comput. Math. Educ.* (TURCOMAT), vol. 12, no. 14, pp. 4186–4202, 2021.

[15] P. Rijwani and S. Jain, "Enhanced software effort estimation using multi-layered feedforward artificial neural network technique," *Procedia Comput. Sci.*, vol. 89, pp. 307–312, 2016.

[16] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho, "Neural network models for software development effort estimation: A comparative study," *Neural Comput. Appl.*, vol. 27, no. 8, pp. 2369–2381, 2016.

[17] A. Heiat, "Comparison of artificial neural network and regression models for estimating software development effort," *Inf. Softw. Technol.*, vol. 44, no. 15, pp. 911–922, 2002.

[18] A. Salam, A. Khan, and S. Baseer, "A comparative study for software cost estimation using COCOMO-II and Walston-Felix models," in *Proc. 1st Int. Conf. Innovations in Comput. Sci. Softw. Eng.*, ICONICS, 2016, pp. 15–16.

[19] S. Nanda and B. Soewito, "Modeling software effort estimation using hybrid PSO-ANFIS," in *2016 Int. Sem. Intell. Technol. Appl.* (ISITIA), 2016, pp. 219–224.

[20] Z. R. Mohsin, "Comparative study for software effort estimation by soft computing models," *J. Educ. Pure Sci.-Univ. Thi-Qar*, vol. 11, no. 2, pp. 108–120, 2021.

[21] S. Kamal and J. A. Nasir, "A fuzzy logic-based software cost estimation model," *Int. J. Softw. Eng. Appl.*, vol. 7, no. 2, pp. 7–18, 2013.

[22] E. P. Edinson and L. Muthuraj, "Performance analysis of FCM-based ANFIS and ELMAN neural network in software effort estimation," *Int. Arab J. Inf. Technol.*, vol. 15, no. 1, pp. 94–102, 2018.

[23] H. Tanarslan, M. Secer, and A. Kumanlioglu, "An approach for estimating the capacity of RC beams strengthened in shear with FRP reinforcements using artificial neural networks," *Constr. Build. Mater.*, vol. 30, pp. 556–568, 2012.

[24] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[25] D. R. Baughman and Y. A. Liu, *Neural Networks in Bioprocessing and Chemical Engineering*. Academic Press, 2014.

[26] A. M. Melesse and R. S. Hanley, "Artificial neural network application for multi-ecosystem carbon flux simulation," *Ecol. Model.*, vol. 189, no. 3–4, pp. 305–314, 2005.

[27] M. Bilgili, B. Sahin, and A. Yasar, "Application of artificial neural networks for the wind speed prediction of target station using reference stations data," *Renew. Energy*, vol. 32, no. 14, pp. 2350–2360, 2007.

[28] J.-S. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, 1993.