



# Adaptive Neuro-Fuzzy Inference Expert System for Agile-Inspired Software Development

Dharmendra Pathak<sup>1\*</sup> and Mohit Arora<sup>2</sup>

<sup>1,2</sup> Lovely Professional University, Punjab, India  
\*dharmendra.32553@lpu.co.in

**Abstract.** Precise effort prediction in Agile settings where needs often shift, is a big hurdle due to a lack of trustworthy data tools and dependence on past project know-how. Due to this, poor guesses and project failures may happen. To address this issue, we have created Agilator, an expert system to bridge the gap between real and guessed user story efforts. We have integrated machine learning models i.e., Adaptive Neuro-Fuzzy Inference System (ANFIS) and Genetic Algorithms to provide optimal predictions. We have tested Agilator on datasets containing 162 data points with important attributes like "Number of Story Points" and "Project Speed". The model achieved a training variance score of 0.99,  $R^2$  score of 0.99, and Root Mean Squared Error (RMSE) of 2.00. In testing, we got a variance score of 0.99,  $R^2$  score of 0.99, and RMSE of 1.93, which is better than other traditional algorithms. Our model has also used real-time adjustment and visual data profiling, which leads to optimal predictions. These outcomes show its potential as a useful and reliable tool to estimate Agile effort and manage projects.

**Keywords:** Agile, Expert System, Machine Learning, Software Effort Estimation, Story Point.

## 1 Introduction

The International Cost Estimating and Analysis Association (ICEAA) and the Standish Group Chaos Manifesto report that many software projects struggle with wrong guesses about work and money needed [1]. In the last 20 years, the biggest change has been from Waterfall to Agile methods. This switch from heavy to light process models has fixed many problems that slow down software projects. The two main issues are: (i) wrong effort estimates and (ii) unclear software needs. People have tried and used many ways to estimate, from old methods to machine learning models, to help with Agile methods [2]. Because software projects are complex and unpredictable adaptive neuro-fuzzy models work well to solve these problems. Despite this, many IT managers working in Agile environments continue to depend on conventional estimation techniques like planning poker and expert judgment, which are often influenced by individual bias. This underscores the critical need for an expert system in this area. The Agilator architecture, built on adaptive networks and neuro-fuzzy principles, aims to assist managers in making well-informed decisions about resource allocation for their projects.

© The Author(s) 2025

S. Bhalerao et al. (eds.), *Proceedings of the International Conference on Recent Advancement and Modernization in Sustainable Intelligent Technologies & Applications (RAMSITA-2025)*, Advances in Intelligent Systems Research 192,  
[https://doi.org/10.2991/978-94-6463-716-8\\_36](https://doi.org/10.2991/978-94-6463-716-8_36)

A typical expert system contains a knowledge pool of expertise, inference engine and user interface [3] as shown in Figure 1.

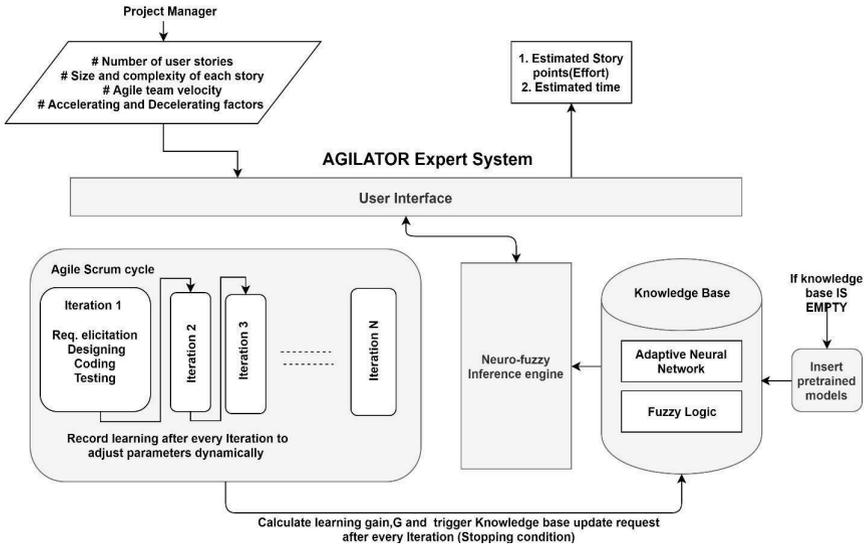


Fig. 1. Agilator Iterative Expert System.

In Figure 1, an abstract view of Agilator iterative expert system is shown wherein an authorized project stakeholder like project managers will provide certain inputs through defined user interface and observe expected outputs. The inputs will be selected by popular dimensionality reduction technique i.e., Principal Component Analysis. A neuro fuzzy inference engine will do the most essential work creating complex relationships and rules for our knowledge base. The knowledge base is a pool of pretrained models and update-train solutions recorded after every iteration through learning gain,  $G$  given in Equation (1) [4]:

$$G = \left( \frac{M_{new} - M_{prev}}{M_{prev}} \right) * 100 \quad (1)$$

Where:

- $G$ : Learning gain (in percentage).
- $M_{new}$ : The performance metric (e.g., accuracy, loss reduction) of the model after the current iteration.
- $M_{prev}$ : The performance metric of the model before the current iteration.

The remaining work has been bifurcated as follows. Section 2 discusses about related work, Section 3 briefly described about background work. Section 4 describes about the optimized hybrid proposed approach. Section 5 describes the experimental results. Section 6 discusses various statistical tests and proving the effectiveness of the proposed model. Section 7 discusses the threat to validity. Section 8 describes the conclusion and future scope.

## 2 Related Work

The reviewed literature indicates that effort estimation remains a critical focus of research in software engineering due to its indispensable role in the IT industry. Over time, there has been a noticeable transition in process models, shifting from Waterfall to Agile approaches [5]. Traditional estimation methods, such as empirical techniques and Delphi-Cost, are increasingly seen as less suitable for Agile frameworks [6]. Agile's emphasis on adaptability aligns well with the capabilities of soft computing techniques, which address inherent challenges and provide reliable estimation outcomes [7-8].

Machine learning techniques have been applied either independently or in combination with other machine learning methods [9-11]. The comprehension review summarizes various machine learning techniques used for effort estimation, highlighting their accuracy parameters, datasets, and comparative performance.

Optimized LSTM models achieved superior results across datasets such as China, Desharnais, and Cocomo81, surpassing KNN, Random Forest, and Gradient Boosted Trees. Deep learning methods like CNN, RNN, and LSTM performed well on MDP NASA and COCOMO II datasets, while GEHO-based Neuro-Fuzzy Networks excelled on COCOMO81 and NASA datasets, outperforming other neuro-fuzzy and GA-based models. Techniques like Random Forest, Decision Tree, and SVM showed varying MARE values across open-source and software design datasets. Advanced approaches like hybrid ABC-PSO algorithms (Mean Absolute Relative Error (MARE): 0.0569) and Stacking Ensemble Methods (Mean Absolute Error (MAE): 0.0383) outperformed individual and traditional models, while CNN with fuzzy clustering and ensemble methods like Gradient Boosting achieved lower error rates and superior predictive accuracy on benchmark datasets [12-23].

## 3 Background Work

The proposed approach is inspired by the Adaptive Neuro-Fuzzy Inference System (ANFIS), recognized as a universal estimator. ANFIS, in its original form, has shown substantial potential and provided effective solutions to challenges inherent in heavyweight process models for software estimation. However, its direct application in Agile environments reveals certain limitations that reduce its effectiveness [24-25]. These limitations include significant computational overhead due to its complex structure and reliance on gradient-based learning, resulting in slower performance for large datasets.

### 3.1 Standard ANFIS Architecture

Both the Takagi-Sugeno Fuzzy System and the Adaptive Neuro-Fuzzy Inference System (ANFIS) form a single framework that delivers the advantages of fuzzy logic and neural networks. Five main layers, each made up of perceptrons, make up the

typical ANFIS architecture, which is depicted in Figure 2 [26-29]. These layers are detailed as follows:

1. Fuzzification Layer: This layer comprises adaptive nodes that contain premise parameters, responsible for converting crisp inputs into fuzzy sets.
2. Implication Layer: Here, the neurons calculate the product of inputs, representing the firing strength of rules.
3. Normalization Layer: In this layer, neurons are fixed and normalize the firing strengths.
4. Defuzzification Layer: This layer includes adaptive neurons with consequence parameters, which contribute to output generation.
5. Combination Layer: The final layer has a single neuron that sums up all the inputs to produce the final output.

The structure of the ANFIS system is depicted in Figure 2, providing a clear visualization of its functional components [30].

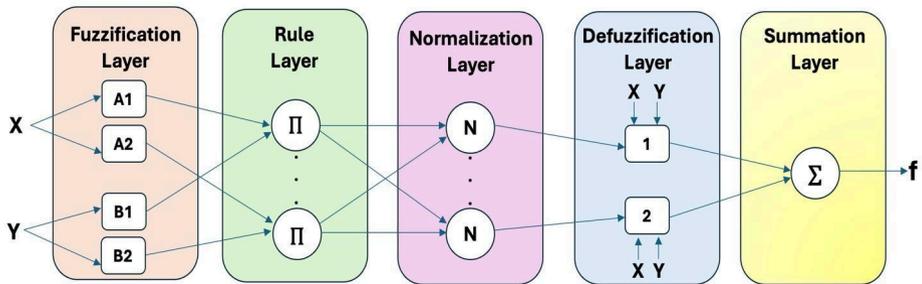


Fig. 2. Standard ANFIS Architecture.

## 4 Approach

To initiate the effort estimation process for new projects, the system requires input data to train specific project parameters. These parameters are primarily stored in the Agilator knowledge base. However, to ensure the data is suitable for processing, it is first passed through a data preparation module. This section elaborates on the proposed algorithm and the framework of the expert system, Agilator.

### 4.1 Methodology

The methodology involves using Agile project data from various software companies. Before being utilized by the algorithm, this data undergoes preprocessing steps such as normalization and scaling. The methodology is outlined through an algorithm and a flowchart presented in Figure 3, divided into four main categories as described below.

## Data Preparation

The dataset for this study has been sourced from Agile projects implemented in various software companies. Since the data requires preprocessing, the following steps have been performed:

1. Data Understanding and Loading: Load the dataset containing Agile project details.
2. Feature Selection: Identify and select key features such as the number of story points and project velocity, with the actual effort serving as the label.
3. Data Transformation: Box-Cox transformation has been applied to prepare the dataset for further analysis.
4. Normalization: Normalize the dataset to make the dataset uniform.
5. Scaling: Scaling is performed on story points and project velocity data to the range of [0-1]. For individual data point, Equation (2) has been used for normalization (N):

$$N = \frac{(i - \min(D))}{(\max(D) - \min(D))} \quad (2)$$

Where  $\max(D)$  and  $\min(D)$  are the maximum and minimum values of the data set  $D$ , respectively.

## Data Partitioning and Model Selection

The dataset is divided into training and testing after normalization and scaling operations for model training and evaluation.

1. We have split the dataset into training (80%) and testing (20%) subsets using the `train_test_split` function available in `sklearn` library along with a fixed `random_state` to 0 for reproducibility.
2. Selection of suitable models like regression models and ANFIS with optimization techniques.
3. Randomized Search and Grid Search methods are used for hyperparameter tuning.
4. Then, model training using the training dataset.

## Testing Part

Testing means evaluating the trained model on the test dataset.

1. Perform predictions on both testing and training datasets.
2. Comparison of the predicted values with the actual values for the performance evaluation.

## Performance Evaluation

It is performed using various statistical and error metrics.

1. Calculation of the loss function i.e., Mean Squared Error (MSE).
2. Determine the average variance between the predicted and actual dataset values.

3. Use additional performance metrics, such as Mean Absolute Relative Error (MARE), Mean Relative Error (MRE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE), to assess the model’s effectiveness.
4. Compare the results obtained from these metrics to identify the best-performing model.

This structured approach ensures a comprehensive evaluation of model performance, facilitating the selection of the most accurate and efficient method for effort estimation.

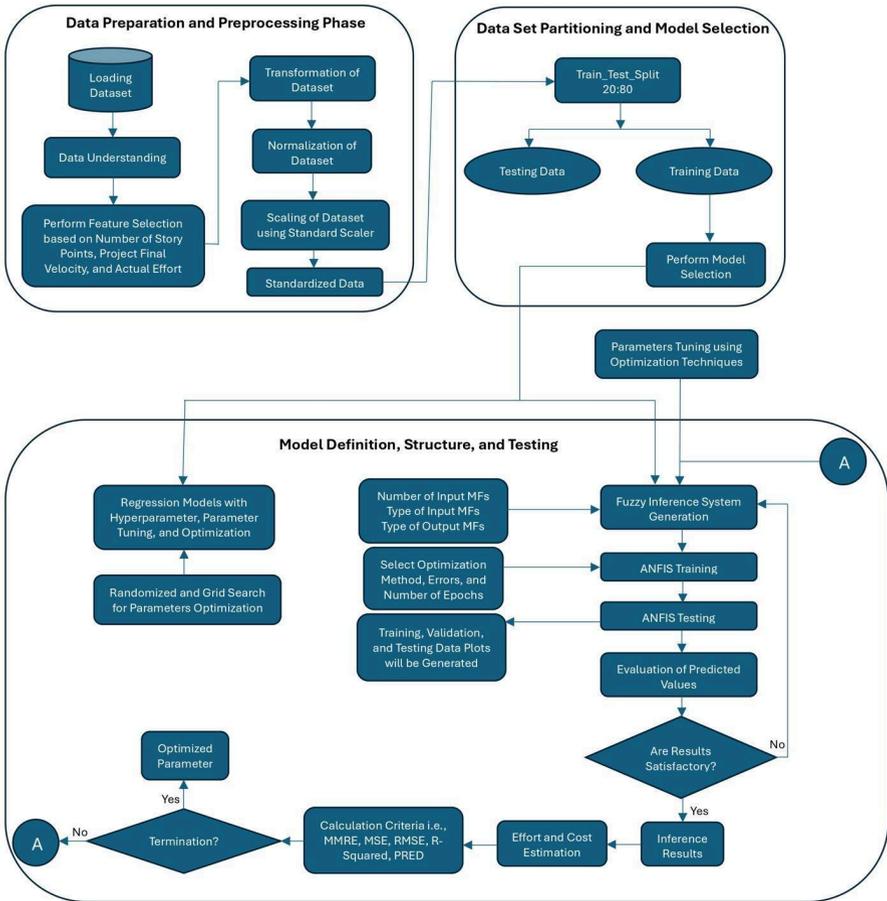


Fig. 3. Methodology for Agile Data Project Processing.

## 5 Results and Discussion

This section presents and analyses the results based on the dataset provided. The findings clearly indicate that the proposed system outperforms existing algorithms in

the same domain. The accuracy achieved highlights the effectiveness of the proposed approach.

### 5.1 Data Understanding and Preprocessing

Dataset to test the proposed framework is flexible to adapt and incorporate the changes, as and when required.

#### Dataset Sample

The dataset sample has been given in Table 3. It has different fields, but all may not be important for the project under consideration for estimation.

**Table 3.** Dataset Sample with Description.

Features		Labels
Number of Story Points	Project Velocity	Actual Effort
160	2.72	64
206	2.53	93
177	3.31	57
336	3.89	87

#### Dataset Profiling

Various profiles of the selected data fields can also be generated using the expert system; thus, it will assist managers to view the data as different plots and figures. The dataset profiles of “Number of Story points” are presented in Figure 4 and same has been performed on “Project Velocity” and “Actual effort”.

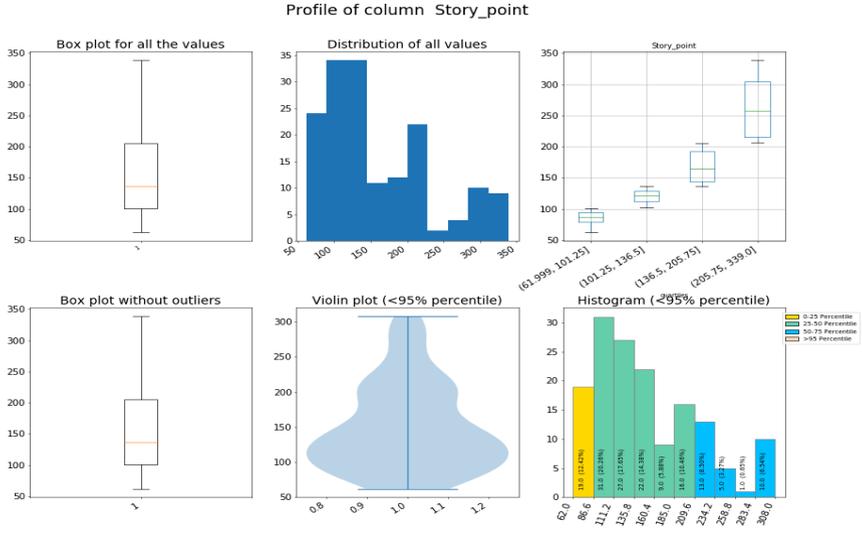


Fig. 4. Number of Story Points Profile.

### Dataset Transformation

We have applied Box-Cox data transformation to stabilize variance and make data more closely follow a normal distribution. It also improves the performance of our models by addressing skewness in the data. The dataset transformations of “Number of Story points” are presented in Figure 5 and same has been performed on “Project Velocity” and “Actual effort”.

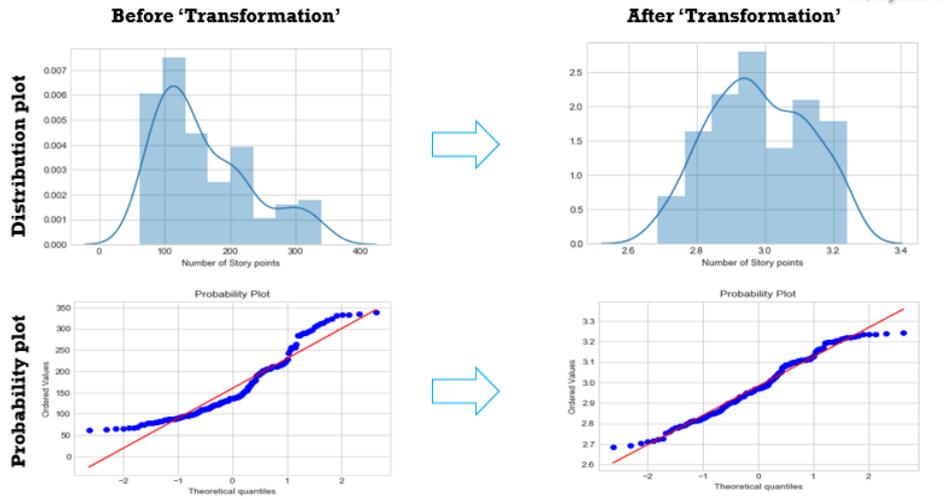


Fig. 5. Box-Cox Transformation of Number of Story Points.

**5.2 Model Selection**

The Adaptive Neuro-Fuzzy Inference System (ANFIS), enhanced with parameter tuning using a Genetic Algorithm, has been applied to the processed and transformed dataset after performing the train\_test\_split.

In the proposed model, the following principal components were identified using Principal Component Analysis (PCA):

- a. Number of Story Points (x)
- b. Project Velocity (y)

**Data loading and Generate Fuzzy Inference system**

After incorporating the selected features into the model, the antecedent layer generates the following input Membership Functions (MFs):

- 1. Number of inputs: 2
- 2. Number of outputs: 1
- 3. Number of inputs MFs: 3 (3 for each input)
- 4. Number of train data pairs: 162
- 5. Partitioning method: Grid partition
- 6. Input MF type: gaussmf
- 7. Output MF type: Linear
- 8. Base fuzzy system: genfis

**5.3 Agilator Training**

The experimental scores have been discussed in this section. The values and the training scores of variances,  $r^2$  and RMSE are tabulated in Table 4 given below:

**Table 4.** Training Score of Proposed Models.

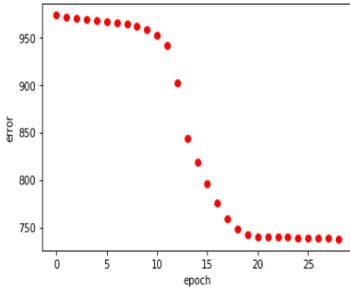
Legends	Training Scores
train var	0.9925606934334674
train $r^2$	0.9925604556232663
train rmse	2.005180551804164

Table 4 shows the training scores for variance,  $R^2$ , and RMSE based on the experimental results, representing the optimal performance of the proposed model with values of 0.99256, 0.99256, and 2.0051, respectively. Optimal parameters for the

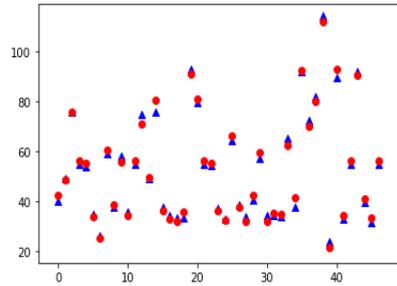
Agilator model were derived from the Genetic Algorithm, getting an average fitness of 0.9866 and a best fitness of 0.9920. Using the parameter optimization, we got the optimal solution (2.65, 7.44), which shows the algorithm's effectiveness.

#### 5.4 Testing Model

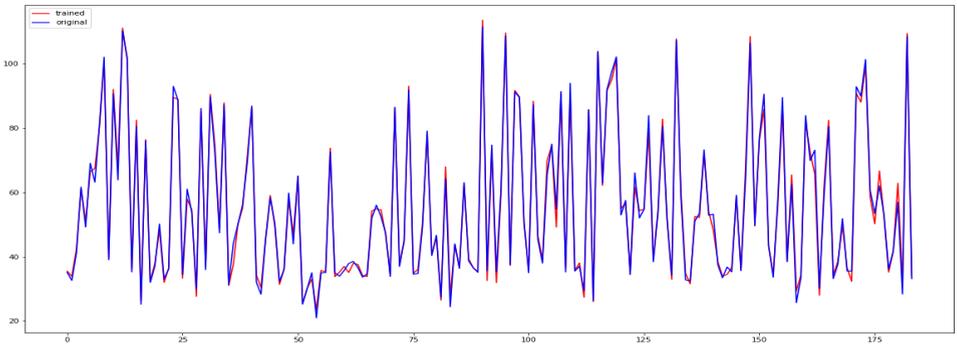
Incorporate the optimized parameters derived from the algorithm into the fuzzy system and evaluate the system's performance by calculating the error metrics as shown in Figure 6, 7, and 8.



**Fig. 6.** ANFIS Error vs Epochs.



**Fig. 7.** ANFIS Predicted vs Estimated Scatter Plot.



**Fig. 8.** Predicted vs Actual Line Plot of Proposed Model.

#### 5.5 Model Performance Evaluation

The proposed model demonstrates excellent performance with a test variance of 0.9918, test  $R^2$  of 0.9917, and a low test RMSE of 1.93, indicating high accuracy and minimal prediction error. Test Score of proposed model is given below:

- test var: 0.991779551136201
- test r2: 0.9917298477867595
- test rmse: 1.930231387985889

## 6 Conclusion

The study addresses the critical challenge of effort estimation in Agile environments, where volatile requirements and limited empirical tools often lead to inaccurate projections and project failures. To bridge this gap, the proposed system, Agilator, leverages advanced machine learning techniques, including an optimized Adaptive Neuro-Fuzzy Inference System (ANFIS), to deliver accurate and real-time effort predictions. By integrating features such as parameter optimization, robust data preprocessing, and real-time adaptability, Agilator effectively handles the complexities of Agile projects. Our model offers a comprehensive solution for IT managers to manage project size and complexity efficiently within a single framework and also addresses the limitations of empirical estimation tools. Future work may explore the enhancements in the capabilities of Agilator by incorporating evolutionary algorithms. Domain-specific datasets and real-time feedback mechanisms may be considered to increase model adaptability. The addition of capabilities like resource allocation and risk management could significantly place Agilator as a modern project management tool for Agile environments.

## References

1. Kumar, S., Arora, M., & Chopra, S. A Review of Effort Estimation in Agile Software Development using Machine Learning Techniques. In 2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA) (pp. 416-422). IEEE. (2022, September).
2. Assefa, Y., Berhanu, F., Tilahun, A., & Alemneh, E. (2022, November). Software Effort Estimation using Machine learning Algorithm. In 2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA) (pp. 163-168). IEEE.(2022)
3. Tawosi, Vali, Rebecca Moussa, and Federica Sarro. "Agile effort estimation: Have we solved the problem yet? Insights from a replication study." *IEEE Transactions on Software Engineering* 49, no. 4 (2022): 2677-2697.
4. Mehdi, Riyadh AK. "Predicting and Explaining Variations in Software Effort Estimation Using Adaptive Fuzzy-Neural Networks with Clustering." In *Proceedings of SAI Intelligent Systems Conference*, pp. 765-779. Cham: Springer Nature Switzerland, 2023.
5. H. Chen, B. Xu, and K. Zhong, "Enhancing Software Effort Estimation through Reinforcement Learning-based Project Management-Oriented Feature Selection," *arXiv preprint arXiv:2403.16749*, 2024. [Online]. Available: <https://arxiv.org/abs/2403.16749>.
6. Coelho Jr, Claudionor N., Hanchen Xiong, Tushar Karayil, Sree Koratala, Rex Shang, Jacob Bollinger, Mohamed Shabar, and Syam Nair. "Effort and Size Estimation in Software Projects with Large Language Model-based Intelligent Interfaces." *arXiv preprint arXiv:2402.07158* (2024).
7. V. Uc-Cetina, "Recent Advances in Software Effort Estimation using Machine Learning," *arXiv preprint arXiv:2303.03482*, 2023. [Online]. Available: <https://arxiv.org/abs/2303.03482>.
8. N. Tran, T. Tran, and N. Nguyen, "Leveraging AI for Enhanced Software Effort Estimation: A Comprehensive Study and Framework Proposal," *arXiv preprint arXiv:2402.05484*, 2024. [Online]. Available: <https://arxiv.org/abs/2402.05484>.

9. Meenakshi, and Meenakshi Pareek. "Software Effort Estimation Using Deep Learning: A Gentle Review." In International Conference on Sustainable and Innovative Solutions for Current Challenges in Engineering & Technology, pp. 351-364. Singapore: Springer Nature Singapore, 2023.
10. S. Sharma and S. Vijayvargiya, "Modeling of Software Project Effort Estimation: A Comparative Performance Evaluation of Optimized Soft Computing-Based Methods," International Journal of Information Technology, vol. 14, pp. 2487–2496, 2022. [Online]. Available: <https://link.springer.com/article/10.1007/s41870-022-00962-5>.
11. M. Pareek and M. Pareek, "Software Effort Estimation Using Deep Learning: A Gentle Review," in Artificial Intelligence and Sustainable Computing, Springer, 2024, pp. 351–364. [Online]. Available: [https://link.springer.com/chapter/10.1007/978-981-97-0327-2\\_26](https://link.springer.com/chapter/10.1007/978-981-97-0327-2_26).
12. K. Lavingia, R. Patel, V. Patel, and A. Lavingia, "Software Effort Estimation using Machine Learning Algorithms," Scalable Computing: Practice and Experience, vol. 25, no. 2, 2024. [Online]. Available: <https://scpe.org/index.php/scpe/article/view/2213>.
13. T. N. Tran, H. T. Tran, and Q. N. Nguyen, "Leveraging AI for Enhanced Software Effort Estimation: A Comprehensive Study and Framework Proposal," arXiv preprint arXiv:2402.05484, 2024. [Online]. Available: <https://arxiv.org/pdf/2402.05484>.
14. A.-E. Iordan, "An Optimized LSTM Neural Network for Accurate Estimation of Software Development Effort," Mathematics, vol. 12, no. 2, p. 200, 2024. [Online]. Available: <https://www.mdpi.com/2227-7390/12/2/200>.
15. M. Pareek and M. Pareek, "Software Effort Estimation Using Deep Learning: A Gentle Review," in Artificial Intelligence and Sustainable Computing, Springer, 2024, pp. 351–364. [Online]. Available: [https://link.springer.com/chapter/10.1007/978-981-97-0327-2\\_26](https://link.springer.com/chapter/10.1007/978-981-97-0327-2_26).
16. S. Sharma and S. Vijayvargiya, "Modeling of Software Project Effort Estimation: A Comparative Performance Evaluation of Optimized Soft Computing-Based Methods," International Journal of Information Technology, vol. 14, pp. 2487–2496, 2022. [Online]. Available: <https://link.springer.com/article/10.1007/s41870-022-00962-5>.
17. Kumar, Sandeep, Mohit Arora, and Shivali Chopra. "A Review of Effort Estimation in Agile Software Development using Machine Learning Techniques." In 2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 416-422. IEEE, 2022.
18. Assefa, Yibeltal, Fekerte Berhanu, Asnakech Tilahun, and Esubalew Alemneh. "Software Effort Estimation using Machine learning Algorithm." In 2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA), pp. 163-168. IEEE, 2022.
19. T. N. Tran, H. T. Tran, and Q. N. Nguyen, "Leveraging AI for Enhanced Software Effort Estimation: A Comprehensive Study and Framework Proposal," arXiv preprint arXiv:2402.05484, 2024. [Online]. Available: <https://arxiv.org/abs/2402.05484>.
20. A.-E. Iordan, "An Optimized LSTM Neural Network for Accurate Estimation of Software Development Effort," Mathematics, vol. 12, no. 2, p. 200, 2024. [Online]. Available: <https://www.mdpi.com/2227-7390/12/2/200>.
21. M. Pareek and M. Pareek, "Software Effort Estimation Using Deep Learning: A Gentle Review," in Artificial Intelligence and Sustainable Computing, Springer, 2024, pp. 351–364. [Online]. Available: [https://link.springer.com/chapter/10.1007/978-981-97-0327-2\\_26](https://link.springer.com/chapter/10.1007/978-981-97-0327-2_26).
22. S. Sharma and S. Vijayvargiya, "Modeling of Software Project Effort Estimation: A Comparative Performance Evaluation of Optimized Soft Computing-Based

- Methods,” *International Journal of Information Technology*, vol. 14, pp. 2487–2496, 2022. [Online]. Available: <https://link.springer.com/article/10.1007/s41870-022-00962-5>.
23. Kumar, Sandeep, Mohit Arora, and Shivali Chopra. "A Review of Effort Estimation in Agile Software Development using Machine Learning Techniques." In 2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 416-422. IEEE, 2022.
  24. Assefa, Yibeltal, Fekerte Berhanu, Asnakech Tilahun, and Esubalew Alemneh. "Software Effort Estimation using Machine learning Algorithm." In 2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA), pp. 163-168. IEEE, 2022.
  25. Tawosi, Vali, Rebecca Moussa, and Federica Sarro. "Agile effort estimation: Have we solved the problem yet? Insights from a replication study." *IEEE Transactions on Software Engineering* 49, no. 4 (2022): 2677-2697.
  26. Mehdi, Riyadh AK. "Predicting and Explaining Variations in Software Effort Estimation Using Adaptive Fuzzy-Neural Networks with Clustering." In *Proceedings of SAI Intelligent Systems Conference*, pp. 765-779. Cham: Springer Nature Switzerland, 2023.
  27. M. Azzeh, M. Elsheikh, and R. Al-Akhras, "Improving Software Effort Estimation Accuracy Using CNN and Fuzzy Clustering," *Journal of Systems and Software*, vol. 191, p. 111461, 2023.
  28. Assefa, Yibeltal, Fekerte Berhanu, Asnakech Tilahun, and Esubalew Alemneh. "Software Effort Estimation using Machine learning Algorithm." In 2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA), pp. 163-168. IEEE, 2022.
  29. H. Chen, B. Xu, and K. Zhong, "Enhancing Software Effort Estimation through Reinforcement Learning-based Project Management-Oriented Feature Selection," *arXiv preprint arXiv:2403.16749*, 2024. [Online]. Available: <https://arxiv.org/abs/2403.16749>.
  30. N. Tran, T. Tran, and N. Nguyen, "Leveraging AI for Enhanced Software Effort Estimation: A Comprehensive Study and Framework Proposal," *IEEE Xplore*, 2023. [Online]. Available: <https://ieeexplore.ieee.org/document/10420603>.

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

