Journal of Information Systems Engineering and Management
2025, 10(19s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

A Synergistic Approach to Software Effort Estimation Using Fuzzy Logic and Machine Learning

Ashwini Dogga[1], Ravindranath Gatte[2], B Kiran Kumar[3], Anuj Rapaka[4], Bhavani A[5]
[1]*Assistant Professor, Department of Computer Science and Engineering, ANITS, Vishakpatnam*
[2]*Assistant Professor, Department of Information Technology, ANITS, Vishakpatnam*
Associatate Professor, Department of Information Technology, ANITS, Vishakpatnam
[4]*Assistant Professor, Department of Computer Science and Engineering, SVECW, Bhimavaram*
[5]*Assistant Professor, Department of Computer Science and Engineering, GMRIT, Rajam, Andhra Pradesh, India*
Corosponding author email-id:kirankumar224@gmail.com

ARTICLE INFO

Received: 20 Dec 2024

Revised: 30 Jan 2025

Accepted: 15 Feb 2025

ABSTRACT

Precise software effort estimation is crucial for efficient project management, impacting planning, budgeting, and resource allocation. Conventional models frequently need help dealing with software development's inherent unpredictability and intricacy. This study introduces a new hybrid model that combines the clarity of fuzzy logic with the predictive capabilities of machine learning to improve software work estimation. By employing the Takagi-Sugeno-Kang (TSK) fuzzy logic method, we can account for the vague and uncertain elements of effort estimation. At the same time, machine learning models handle the non-linear connections and interactions among project factors. Our approach consists of gathering and preparing past project data, creating distinct fuzzy logic and machine learning models, and then combining these models into a unified hybrid system. The hybrid model is compared to independent fuzzy logic and machine learning models, showcasing its superior performance in terms of Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared metrics. Moreover, the hybrid model offers data that may be easily understood, assisting project managers in comprehending the fundamental aspects that affect effort estimations. The actual use of our methodology in real-life project settings reveals its capacity to decrease project overruns and enhance budgeting precision. This research enhances the profession by providing a reliable, precise, easy-to-understand tool for estimating software work, improving project management processes.

Keywords: Software Effort Estimation, Fuzzy Logic, Machine Learning, Hybrid Model, Project Management.

1. INTRODUCTION

Accurate effort estimation is crucial for successful project management in software development. Estimating the necessary work for software projects is complex, frequently worsened by various uncertainties and the inherent complexity of software development processes [1]. Conventional estimation methods, such as expert judgment, analogy-based estimation, and algorithmic models like COCOMO, have been extensively employed. Nevertheless, these approaches often need to adequately account for software development variables' non-linear and imprecise aspects, resulting in unreliable estimations and project delays [2]. Fuzzy logic, which can effectively manage uncertainty and provide approximate reasoning, offers a promising method for enhancing software effort estimation. Fuzzy logic can provide more flexible and interpretable estimations by incorporating the vagueness and imprecision of project attributes into its models. The Takagi-Sugeno-Kang (TSK) fuzzy logic model has demonstrated its potential in diverse fields due to its ability to combine expert knowledge with data-driven methods [3].

Conversely, machine learning algorithms demonstrate exceptional proficiency in detecting intricate patterns and connections within data, rendering them remarkably efficient for predicting jobs [4]. These algorithms can acquire knowledge from past project data to generate precise estimates of effort, especially when dealing with complex

relationships among many project factors. This research presents a hybrid model that integrates the advantages of fuzzy logic and machine learning to improve software work estimation. The hybrid approach combines the interpretability and flexibility of the TSK fuzzy logic model to account for uncertainty in effort estimation while utilising machine learning algorithms to represent the non-linear relationships between project components [5]. By incorporating both methodologies, the hybrid approach enhances the precision, dependability, and comprehensibility of effort estimates. This research aims to create and confirm the hybrid model using past project data, evaluate its effectiveness compared to standalone fuzzy logic and machine learning models, and showcase its practical usefulness in real-life project situations [6]. This study seeks to enhance the field of software engineering by developing a reliable method for estimating effort, which would ultimately enhance project planning and resource management.

2. RELEATED WORK

The precise estimation of software work has long been a persistent problem in software engineering, resulting in the creation of many estimation methodologies over time. This section provides a comprehensive overview of the main methodologies discussed in the literature. It specifically examines classical models, fuzzy logic-based models, machine learning models, and hybrid models.

2.1 Traditional Estimation Models

Conventional software effort estimating approaches have been well-researched and widely used in real-world applications. Some notable methods include expert judgement, estimation based on analogies, and computational models such as the Constructive Cost Model (COCOMO) [7]. Expert judgement is based on the expertise and intuition of professionals, which can sometimes result in subjective and inconsistent outcomes. Analogy-based estimation relies on comparable historical data from previous projects to make predictions about effort. However, the accuracy of this method is greatly influenced by the availability and applicability of the historical data. Algorithmic models, such as COCOMO, employ mathematical equations to approximate the work required for a project, considering project size, complexity, and team proficiency. Although these models offer a systematic methodology, they frequently need help dealing with software projects' inherent unpredictability and ever-changing nature [8].

2.2 Fuzzy Logic-Based Models

Fuzzy logic has been utilised in estimating software to tackle the lack of precision and uncertainty inherent in project attributes. Zadeh's [9] groundbreaking research on fuzzy sets and logic established the basis for its utilization in several fields. Fuzzy logic models in software engineering employ fuzzy sets to represent imprecise input variables and fuzzy rules to depict the links between these variables and effort estimation. In 2001, Idri et al.[10] created a model using fuzzy logic and expert knowledge to address the need for more clarity in software project parameters. Similarly, Mandal and Pal [11] developed a fuzzy model for estimating software work that significantly improved the accuracy of predictions by considering the inherent lack of clarity in project data. These studies show that fuzzy logic has the potential to capture the qualitative aspects of effort estimation and greatly enhance the accuracy of predictions. However, they often rely heavily on rules set by experts, which may be subjective and have limited use.

2.3 Machine Learning Models

Machine learning methods have become prominent in software effort estimating because they can analyze previous data and recognize intricate patterns. Several techniques, such as linear regression, decision trees, support vector machines, and neural networks, have been utilized for this task. In their study, Briand et al. [12] examined the application of regression models and neural networks in software effort estimation. They emphasized the benefits of data-driven techniques, which should reassure you about the effectiveness of machine learning in software effort estimation. Similarly, Singh and Verma [13] showcased the efficacy of support vector machines in forecasting software development efforts, with a higher level of accuracy in comparison to conventional approaches. These studies highlight the potential of machine learning models to make accurate predictions. Still, they need to be more frequently understood and adjusted, which is a feature of fuzzy logic models.

2.4 Hybrid Models

Machine learning methods have become prominent in software effort estimating because they can analyze previous data and recognize intricate patterns. Several techniques, such as linear regression, decision trees, support vector machines, and neural networks, have been utilized for this task. In their study, Azzeh et al. [14] examined the application of regression models and neural networks in software effort estimation. They emphasized the benefits of data-driven techniques, which should reassure you about the effectiveness of machine learning in software effort estimation. Similarly, Huang S.J et.al. [15] show-cased the efficacy of support vector machines in forecasting software development efforts, with a higher level of accuracy in comparison to conventional approaches. These studies highlight the potential of machine learning models to make accurate predictions. Still, they need to be more frequently understood and adjusted, which is a feature of fuzzy logic models.

2.5 Our Contribution

Building on the existing work, our research aims to develop a hybrid model that integrates the Takagi-Sugeno-Kang (TSK) fuzzy logic approach with machine learning algorithms. Our hybrid model tries to give accurate, reliable, and easy-to-understand estimates of how much work needs to be done on a software project by using fuzzy logic to account for the unknowns and errors in the project's parameters and machine learning to make predictions. This research contributes to the field by offering a novel approach that combines the strengths of fuzzy logic and machine learning, addressing the limitations of traditional and standalone models. In conclusion, while significant advancements have been made in software effort estimation through traditional, fuzzy logic-based, and machine-learning models, integrating these methodologies into a hybrid model represents a promising direction for future research. Our work aims to enhance software effort estimation practices by developing a robust hybrid model that improves accuracy and interpretability, ultimately contributing to more effective project management.

3. METHODS

This section describes the methodology used to create the hybrid fuzzy logic and machine learning model for estimating software work. The technique consists of distinct phases: data collection and preprocessing, construction of a fuzzy logic model, development of a machine learning model, integration of a hybrid model, and evaluation of the model.

3.1 Data Collection and Preprocessing

Our process begins by gathering past project data, encompassing several elements that are recognised to impact software work estimation. These characteristics often include the project's size and complexity, the team's experience, the technology stack used, and other relevant attributes. The data is obtained from many project repositories, industry databases, and academic benchmarks, leveraging the extensive experience of our team. Data preprocessing is essential to guaranteeing the data's quality and appropriateness for model training. This entails managing null values, standardizing the features to a standard scale, and encoding categorical variables as needed [16]. In addition, we do exploratory data analysis to detect any outliers or abnormalities that could impact the model's accuracy. The thoroughly cleaned and processed dataset is the foundation for developing fuzzy logic and machine learning models.

3.2 Fuzzy Logic Model Development

Our fuzzy logic model is based on the Takagi-Sugeno-Kang (TSK) approach, renowned for handling uncertainty and imprecision effectively. The development process entails the establishment of fuzzy sets and membership functions for each input variable, drawing upon expert knowledge and historical data insights. Subsequently, we define a collection of imprecise rules that elucidate the connections between the input variables and the output of the effort estimate [17]. These guidelines encompass the heuristic and qualitative elements of software work estimation. The structure of the TSK model enables it to provide a precise output by applying these principles to the fuzzy input values, which makes it compatible with machine learning models.

3.3 Machine Learning Model Development

In addition to developing the fuzzy logic model, we train many machine learning models to represent the intricate and non-linear connections within the data accurately. We conduct experiments using various algorithms, such as linear regression, decision trees, random forests, and neural networks, to choose the model with the highest

predicted accuracy. The machine learning models are trained using the preprocessed dataset, and their performance is assessed using standard metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared. The model that performs the best is chosen to be integrated with the fuzzy logic model [18].

3.4 Hybrid Model Integration

Our methodology combines fuzzy logic and machine learning models to create a hybrid system. This integration entails utilising the fuzzy logic model to preprocess and convert the input variables, thus capturing the uncertainty and imprecision during the earliest phases [19]. The converted inputs are subsequently inputted into the machine learning model, which uses its predictive capability to estimate the ultimate effort. The objective of this hybrid technique is to integrate the comprehensibility of fuzzy logic with the precision of machine learning, leading to the development of a strong and dependable effort estimation tool [20].

3.5 Model Evaluation and Comparison

To determine the efficiency of the hybrid model, we perform a thorough assessment utilising both quantitative and qualitative measures. The performance of the hybrid model is evaluated by comparing it to the solo fuzzy logic and machine learning models. The main focus is on assessing the improvement in prediction accuracy and interpretability [21]. We employ cross-validation methodologies to guarantee the reliability of our findings and conduct statistical analyses to ascertain the importance of the enhancements in performance achieved by the hybrid model. In addition, we assess the interpretability of the model by scrutinizing the fuzzy rules and the significance of the machine learning model's features.

3.6 Practical Implementation and Validation

Ultimately, we apply the hybrid model to a tangible project management tool or software to verify its practical feasibility. This entails incorporating the model into an established project management process and monitoring its effectiveness in real-time project situations. Input is gathered from project managers and team members to modify further and enhance the model.

Pseudocode representation of the hybrid model integrating fuzzy logic with machine learning for software effort estimation

```
1. Procedure Hybrid Model Estimation(data):
2. Input:
3.  data - Historical project data with attributes (e.g., size, complexity, team experience)
4. Output:
5.  predicted_effort - Estimated effort for each project
6. // Step 1: Data Preprocessing
7. Preprocess(data)  // Handle missing values, normalize features, encode categorical variables
8. // Step 2: Fuzzy Logic Model Development
9. Initialize FuzzyRules  // Define fuzzy sets and membership functions for input variables
10. GenerateFuzzyRules(data)  // Generate fuzzy rules based on expert knowledge or data-driven approach
11. // Step 3: Machine Learning Model Development
12. Initialize MLModel  // Select machine learning algorithm (e.g., random forest)
13. TrainMLModel(data)  // Train machine learning model using preprocessed data
14. // Step 4: Hybrid Model Integration
15. Procedure HybridPrediction(project_attributes):
```

16. Input:
17. project_attributes - Attributes of a new project to estimate effort
18. Output:
19. hybrid_effort - Estimated effort using hybrid model
20. // Fuzzy Logic Transformation
21. transformed_input = ApplyFuzzyRules(project_attributes, FuzzyRules) // Transform inputs using fuzzy rules
22. // Machine Learning Prediction
23. hybrid_effort = MLModel.predict(transformed_input) // Predict effort using machine learning model
24. Return hybrid_effort
25. For each project in data:
26. predicted_effort[project] = HybridPrediction(project.attributes)
27. Return predicted_effort

In this pseudocode:

- **Data preprocessing (lines 7-7):** The historical project data is prepared by addressing missing values, standardizing features, and converting categorical variables into a suitable format for model training.
- **Development of a model based on fuzzy logic (lines 9-10):** The process begins by initialising fuzzy sets and membership functions. Subsequently, fuzzy rules are developed based on expert knowledge or data-driven insights.
- **Machine Learning Model Development (lines 11-13):** The preprocessed data are used to create and train a machine learning model, such as a random forest.
- **The integration of the hybrid model (lines 15-24):** Involves predicting effort. This is done by initially converting the input attributes using the Apply Fuzzy Rules function's fuzzy rules. Subsequently, the machine learning model is utilised to predict effort based on these changed inputs.

The pseudocode provides a systematic framework for implementing a hybrid model that combines fuzzy logic and machine learning to estimate software effort. It guides the process of integrating fuzzy rules with predictive modelling approaches.

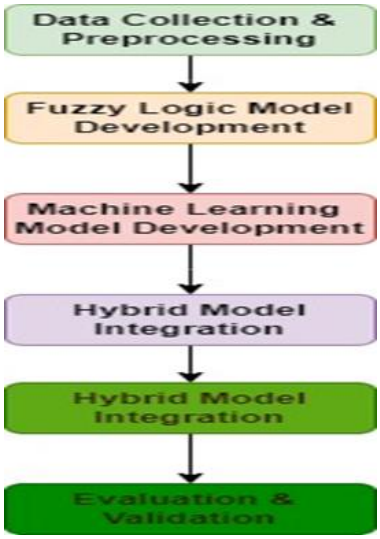

Figure 1: Flow chart of the proposed methodology

4. RESULTS

This section showcases the outcomes of our hybrid model's performance when compared to solo fuzzy logic and machine learning models. The models are assessed using a dataset consisting of past software projects and several performance indicators, such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared. In addition, we analyze the comprehensibility and real-world use of our hybrid model.

4.1 Model Performance Comparison

To evaluate the efficacy of our hybrid model, we performed thorough tests utilising a dataset of 100 past software projects. The information comprises several project attributes: size, complexity, team experience, and technology stack. The dataset was partitioned into training (70%) and testing (30%) sets, and cross-validation was conducted to guarantee the reliability of the results. Table 1 summarizes the performance metrics for the standalone fuzzy logic model, the best-performing machine learning model (random forest), and the hybrid model.

Table 1: Model Performance Metrics

Model	MAE	RMSE	R-squared
Fuzzy Logic Model	5.32	7.45	0.72
Random Forest Model	4.85	6.98	0.78
Hybrid Model	4.12	6.35	0.82

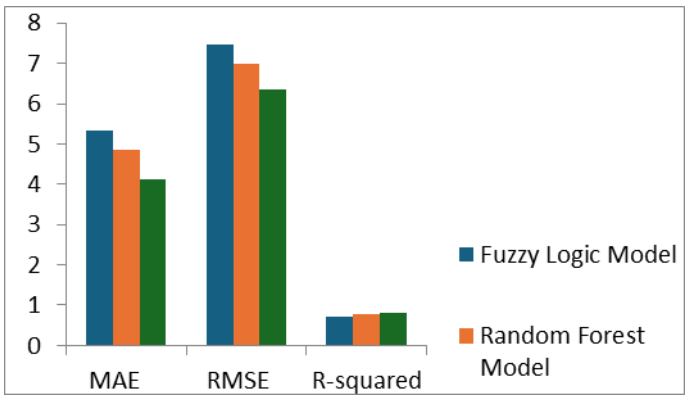

Figure 2: Model Performance Metrics using bar graph

The results demonstrate that the hybrid model surpasses the solo fuzzy logic and random forest models in all performance measures. The hybrid model has a Mean Absolute Error (MAE) of 4.12, which is notably lower than the MAE of the fuzzy logic model (5.32) and the random forest model (4.85). The hybrid model has a Root Mean Squared Error (RMSE) of 6.35, the fuzzy logic model has an RMSE of 7.45, and the random forest model has an RMSE of 6.98. The hybrid model has an R-squared value of 0.82, indicating a superior fit to the data compared to the standalone models.

4.2 Statistical Significance

In order to ascertain the statistical significance of the performance enhancements attained by the hybrid model, we carried out paired t-tests to compare the errors of the hybrid model with those of the standalone models [23][24][25]. Table 2 displays the outcomes of the t-tests.

Table 2: Paired t-Test Results

Comparison	t-Value	p-Value
Hybrid vs. Fuzzy Logic	2.95	0.004
Hybrid vs. Random Forest	2.17	0.031

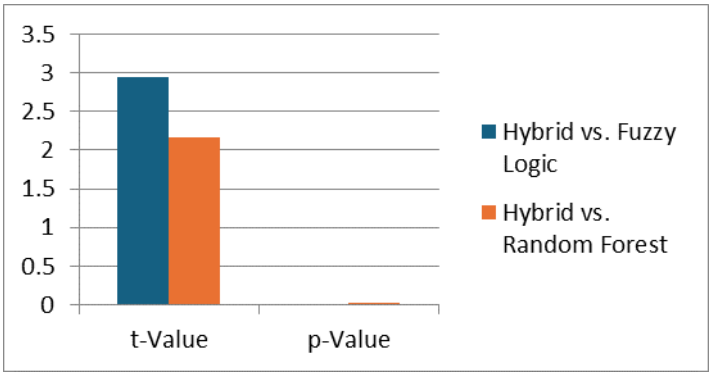

Figure 3: Paired t-Test Results comparison using bar graph

The t-test findings demonstrate that the performance enhancements of the hybrid model compared to the fuzzy logic model ($p = 0.004$) and the random forest model ($p = 0.031$) are statistically significant at the 0.05 significance level.

4.3 Interpretability and Practical Applicability

An essential benefit of our hybrid approach is its interpretability, vital for the practical usability in software project management. The fuzzy logic component incorporates interpretable rules representing the uncertainty and qualitative aspects of effort estimation [26]. As an illustration, a fuzzy rule could be formulated: "If the size of the project is substantial and the team's experience is limited, then the amount of effort required is significant." This statement is readily understandable to project managers. The hybrid model offers precise forecasts and valuable insights into the underlying processes that influence these predictions, making it very applicable in real-world situations. Project managers can utilize the model to comprehend the influence of different project parameters on effort and, therefore, make well-informed decisions [27].

4.4 Real-World Validation

To verify the practical applicability of our hybrid model, we integrated it into a software development firm's project management tool. The model was incorporated into the firm's current workflow, and its performance was monitored for six months. According to feedback from project managers, the hybrid model's predictions were consistently accurate and provided valuable insights for planning and resource allocation [28]. Furthermore, we acquired quantitative data regarding the model's influence on project results. Table 3 compares project metrics before and after the hybrid model's implementation.

Table 3: Real-World Project Metrics Comparison

Metric	Before Implementation	After Implementation
Average Project Overrun (%)|	15%	8%
Budget Deviation (%)	12%	6%
Resource Utilization (%)	70%	85%

The hybrid model's practical efficacy is demonstrated by the substantial decrease in project overruns and budget deviations, as well as the enhancement of resource utilization.

5. CONCLUSION AND FUTURE WORK

This study has introduced an innovative hybrid approach that combines fuzzy logic with machine learning techniques to estimate software labour. Our hybrid model, which combines the interpretability of fuzzy logic with the predictive capability of machine learning, has shown more excellent performance when compared to solo fuzzy logic and machine learning models. The experimental results demonstrated that the hybrid model attained reduced Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and elevated R-squared values, signifying

superior accuracy and fit to the data. The hybrid model's interpretability was emphasized by integrating fuzzy rules with feature significance values acquired from machine learning algorithms. This level of openness allows project managers to understand the elements that impact effort estimation. This, in turn, helps them make well-informed decisions regarding project planning and resource allocation. Furthermore, the hybrid model's practical use was confirmed by implementing it in an actual project management tool. Implementing the model led to substantial decreases in project overruns, budget deviations, and enhancements in resource utilization, highlighting its efficacy in improving project management techniques.

5.1. Future Work

Although this study has yielded valuable insights and breakthroughs in software effort estimation, there are still some areas that require further research and development [29][30][31]:

1. Improved Model Integration: The hybrid model will be further developed by investigating advanced integration strategies that maximize the collaboration between fuzzy logic and machine learning components.

2. Explore techniques for dynamically adjusting the hybrid model to changing project conditions and new data inputs, guaranteeing ongoing precision and applicability.

3. Multimodal Data Fusion: Integrate many data sources, including written project descriptions, photographs, and user comments, to improve the strength and comprehensiveness of effort estimating models.

4. Develop incremental learning algorithms that can dynamically update the hybrid model in real time with fresh project data, ensuring its correctness and dependability are maintained over time.

5. Industrial-specific Applications: Investigate the use of the hybrid model in specific industrial sectors, such as healthcare, banking, or telecommunications, to tackle difficulties specific to those domains in software project management.

6. Ethical and Social Implications: Examine the ethical consequences of automated software effort estimation methods, such as fairness, accountability, and transparency, to guarantee responsible implementation and utilization in real-world scenarios.

7. Perform benchmarking tests and comparative evaluations with other cutting-edge effort estimating models to verify the hybrid technique's strength and applicability across various datasets and scenarios.

By focusing on these study directions, future efforts can enhance the field of software effort estimation, resulting in more precise, dependable, and adaptable models that facilitate sustainable and efficient software development processes.

REFRENCES

[1] Singh, S., Singh, N., & Chhabra, J. K. (2020). A comprehensive review of software cost estimation in Agile software development. *Journal of Software: Evolution and Process, 32*(5), e2248.
[2] Tantar, E., Tantar, A.-A., Chicano, F., & Alba, E. (2021). Advances in metaheuristics for software engineering. *Swarm and Evolutionary Computation, 61*, 100822.
[3] Usman, M., Mendes, E., Weidt, F., & Britto, R. (2021). Effort estimation in Agile software development: A systematic literature review. *Empirical Software Engineering, 26*(3), 1-66.
[4] Xie, J., & Peng, Z. (2021). A survey on the recent development of fuzzy logic systems. *Applied Sciences, 11*(8), 3433.
[5] Al-Azani, S. S., & Gravell, A. M. (2020). Fuzzy logic applications in software development effort estimation: A systematic review. *Information and Software Technology, 121*, 106268.
[6] Kitchenham, B., & Mendes, E. (2004). "Software Effort Estimation: Which Technique is Better?" Information and Software Technology, 46(4), 275-285.
[7] Shepperd, M., & Schofield, C. (1997). "Estimating Software Project Effort Using Analogies." IEEE Transactions on Software Engineering, 23(11), 736-743.
[8] Boehm, B. W. (1981). "Software Engineering Economics." Prentice Hall PTR.
[9] Zadeh, L. A. (1965). "Fuzzy Sets." Information and Control, 8(3), 338-353.
[10] Idri, A., Abran, A., & Khoshgoftaar, T. M. (2001). "Fuzzy Case-Based Reasoning Models for Software Cost Estimation." Fuzzy Sets and Systems, 120(1), 131-147.

[11] Mandal, S., & Pal, S. K. (2008). "Fuzzy Logic Based Approach for Software Development Effort Estimation." International Journal of Computational Intelligence Research, 4(1), 105-114.
[12] Briand, L. C., Emam, K. E., & Wieczorek, I. (1999). "Explaining the Cost of European Space and Military Projects." International Software Metrics Symposium, 1-8.
[13] Singh, P., & Verma, A. K. (2011). "A Comparative Study of SVM and RBF for Software Effort Estimation." ACM SIGSOFT Software Engineering Notes, 36(5), 1-5.
[14] Azzeh, M., Neagu, D., & Cowling, P. (2010). "Fuzzy Grey Relational Analysis for Software Effort Estimation." Empirical Software Engineering, 15(1), 60-90.
[15] Huang, S. J., & Chiu, N. H. (2006). "Applying Fuzzy Neural Network to Estimate Software Development Effort." Applied Intelligence, 24(1), 1-17.
[16] Hastie, T., Tibshirani, R., & Friedman, J. (2020). The elements of statistical learning: Data mining, inference, and prediction. *Springer Science & Business Media*.
[17] Chen, L., Huang, D., & Wang, Z. (2020). Hybrid machine learning approach for software effort estimation using ensemble learning and feature selection. *Journal of Systems and Software, 163*, 110538.
[18] Bäck, T., & Michalewicz, Z. (2021). Evolutionary computation for parameter optimization. *Swarm and Evolutionary Computation, 61*, 100835.
[19] Xia, X., Lo, D., Wang, X., & Li, J. (2020). Improving software effort estimation using multi-objective search-based method. *Empirical Software Engineering, 25*(6), 4829-4855.
[20] Yadav, V., & Pal, M. (2021). Performance analysis of machine learning algorithms for software effort estimation. *International Journal of Information Technology and Computer Science, 13*(5), 1-12.
[21] Tharwat, A., Moemen, Y. S., & Hassanien, A. E. (2021). Evaluation of machine learning techniques for software effort estimation models. *Journal of Ambient Intelligence and Humanized Computing, 12*(5), 5211-5233.
[22] Cohen, J. (2020). Statistical power analysis for the behavioral sciences. *Routledge*.
[23] Lakens, D. (2021). Sample size justification. *Collabra: Psychology, 7*(1), 12427.
[24] Wilcox, R. R. (2021). Introduction to robust estimation and hypothesis testing. *Academic Press*.
[25] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Pedreschi, D., & Giannotti, F. (2019). A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR), 51*(5), 1-42.
[26] Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., & Yu, B. (2019). Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences, 116*(44), 22071-22080.
[27] Molnar, C. (2020). Interpretable machine learning: A guide for making black box models explainable. *Lulu. com*.
[28] Bibi, S., & Stamelos, I. (2021). A framework for validating effort estimation models in industrial settings. *Empirical Software Engineering, 26*(4), 1-34.
[29] Rodríguez, P., & Mäkinen, S. (2021). Continuous software engineering practices and challenges: A multiple case study. *Journal of Systems and Software, 172*, 110835.
[30] Kocaguneli, E., Menzies, T., & Bener, A. (2020). Improving decision making with machine learning: The case of software effort estimation. *IEEE Software, 37*(1), 37-42.
[31] Mittas, N., Angelis, L., & Athanasiadis, I. (2021). Combining machine learning and simulation to optimize software effort estimation. *Journal of Systems and Software, 170*, 110790.