

AI-Based Approaches for Software Tasks Effort Estimation: A Systematic Review of Methods and Trends

Bruno Budel Rossi^a and Lisandra Manzoni Fontoura^b

*Programa de Pós-Graduação em Ciência da Computação, Federal University of Santa Maria (UFSM), Brazil
{bbrossi, lisandra}@inf.ufsm.br*

Keywords: Task Effort Estimation, Machine Learning, Neural Networks, Large Language Models, Natural Language Processing.

Abstract: Accurate measurement of task effort in software projects is essential for effective management and project success in software engineering. Conventional methods often face limitations in both accuracy and their ability to adapt to the complexities of contemporary projects. This systematic analysis examines the use of ensemble learning methods and other artificial intelligence strategies for estimating task effort in software projects. The review focuses on methods that employ machine learning, neural networks, large language models, and natural language processing to improve the accuracy of effort estimation. The use of expert opinion is also discussed, along with the metrics utilized in task effort estimation. A total of 826 empirical and theoretical studies were analyzed using a comprehensive search across the ACM Digital Library, IEEE Digital Library, ScienceDirect, and Scopus databases, with 66 studies selected for further analysis. The results highlight the effectiveness, current trends, and benefits of these techniques, suggesting that adopting AI could lead to substantial improvements in effort estimation accuracy and more efficient software project management.

1 INTRODUCTION

Accurate task effort estimation is vital for successful software project management, as it directly influences scheduling, resource allocation, and project outcomes. Unlike general project effort estimation, which evaluates the total effort for an entire project, task effort estimation predicts the effort for individual tasks or subtasks. This detailed approach is crucial in agile and iterative environments where tasks are frequently reassigned and reprioritized (Ali and Gravino, 2019).

Traditional methods like COCOMO-II (Boehm et al., 2000), Use Case Points (UCP) (Karnier, 1993), and Function Point Analysis (FPA) (Albrecht and Gaffney, 1983) often lack accuracy and adaptability at the task level, leading to delays and cost overruns. These approaches aggregate effort estimates broadly, overlooking the nuances and variability specific to individual task estimations.


Advancements in AI have introduced ensemble learning, which combines multiple machine learning models to enhance prediction robustness (Rahman


et al., 2024). Techniques such as artificial neural networks (ANNs) and hybrid models integrating different algorithms have proven effective in improving the accuracy of task-level estimations (Bilgaiyan et al., 2019; Kumar and Srinivas, 2023). Additionally, deep learning models like LSTM and GRU have been explored for their ability to model sequential data and human factors in tasks, capturing the complexities of task effort estimation (Iordan, 2024).

This systematic review synthesizes recent literature on AI techniques specifically applied to task effort estimation in software projects. Our review stands out by exploring not only the techniques used to predict effort in software project tasks but also the role of expert opinion in this process. In addition, we provide a detailed analysis of the metrics used to evaluate model accuracy to provide a comprehensive overview of current practices and their efficacy in task-level estimation.

Our study provides an up-to-date overview of effort estimation practices, emphasizing task-focused approaches. Examining the strengths and limitations of AI-based techniques offers valuable insights to advance research and support adopting more accurate methods in software engineering.

The paper is organized as follows. Section 2 re-

^a  <https://orcid.org/0009-0000-8153-1564>

^b  <https://orcid.org/0000-0002-4669-1383>

views related work, Section 3 describes our methodology, Section 4 discusses the results, and Section 5 concludes the article with future directions.

2 RELATED WORKS

Several systematic reviews, such as Cabral et al. (2023), have explored Ensemble Effort Estimation (EEE), focusing on dynamic ensemble selection techniques. Our study specifically addresses the use of ensemble methods for task effort estimation, differing from broader reviews like Ali and Gravino (2019), which analyze ML techniques, including artificial neural networks (ANN) and support vector machines (SVM), across various contexts. Unlike Brar and Nandal (2022), who focus on ML techniques for general effort estimation, we concentrate on task-level estimations, emphasizing the unique challenges and advantages of ensemble models.

Usman et al. (2014) focus on effort estimation in agile development, using expert judgment, Planning Poker, and AI-based methods. Their focus on agile environments contrasts with our broader approach, which encompasses diverse software project contexts.

Other notable studies include Dehghan et al. (2016), who proposed a hybrid model combining textual techniques and task metadata, and Bilgaiyan et al. (2019), who investigated neural networks in agile environments.

These studies provide a foundation for understanding the application of ML and AI in software effort estimation. Our research stands out by focusing on ensemble models and exploring their potential to enhance the accuracy and efficiency of task effort estimation.

3 RESEARCH METHOD

This systematic review followed the guidelines proposed by Kitchenham (2004) and Kitchenham et al. (2009) to ensure comprehensive and unbiased coverage of the relevant literature, aiming to provide a complete overview of the use of ensemble models and other AI techniques for task effort estimation in software projects.

The Parsifal tool (<https://parsif.al>) was used to assist in the search and selection of studies, ensuring a rigorous and replicable methodology. The research was conducted in the ACM Digital Library, IEEE Digital Library, ScienceDirect, and Scopus databases, which are highly relevant sources for the field.

3.1 Planning

The PICOC technique was employed to define the scope of this systematic review, guiding the formulation of research questions and the search for relevant evidence. This approach structured the review by focusing on the population (software projects and development teams), intervention (AI techniques like machine learning and ensemble models), comparison (against traditional methods such as COCOMO and function point analysis), outcome (improvements in estimation accuracy and efficiency), and context (software development environments). PICOC ensured a comprehensive and well-defined framework for analyzing the impact of AI-based techniques on task effort estimation.

3.1.1 Research Questions

The following research questions guided the data extraction and analysis:

RQ1: Which AI-based techniques are most commonly used for task effort estimation in software projects, and why are they chosen?

RQ2: Are experts still necessary for task effort estimation or for training AI models? If so, how are they employed?

RQ3: What metrics or benchmarks are commonly used to evaluate the performance of ensemble learning models in task effort estimation?

These questions aim to clarify the current landscape of AI use in effort estimation and to identify best practices and gaps in the literature.

3.1.2 Search Strategy and Data Sources

To ensure the retrieval of relevant articles, we employed the following search string: (“software development” OR “software project”) AND (“ensemble learning” OR “artificial intelligence” OR “neural network” OR “deep learning” OR “large language model” OR “machine learning”) AND (“effort estimation”) AND year \geq 2014.

The review used widely recognized scientific databases to ensure the relevance and quality of the included studies. The selected databases were ACM Digital Library, IEEE Digital Library, ScienceDirect, and Scopus, providing a solid foundation for the systematic review.

3.2 Conducting

Figure 1 summarizes the process of selecting the articles, outlining the three steps taken and the number of articles analyzed and selected at each step.

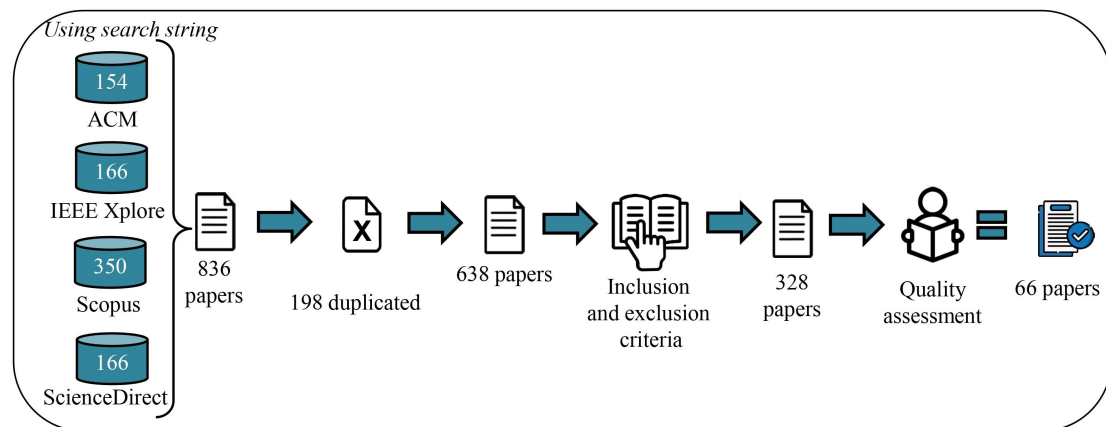


Figure 1: Article search protocol.

After eliminating duplicates using the Parsifal tool, 198 articles were removed, leaving 638 for detailed analysis. In the second step, based on the reading of the title and keywords, the inclusion and exclusion criteria were applied, resulting in 328 selected articles (see Section 3.2.1). In the final selection step, the articles were read in full, resulting in 66 articles being selected for analysis (see Section 3.2.2). Efforts included using institutional subscriptions and contacting authors to access publications, although four articles were excluded due to inaccessibility, ensuring transparency in the process.

3.2.1 Inclusion and Exclusion Criteria

The selection criteria ensured the quality and relevance of the articles:

Inclusion Criteria (IC). Studies published between 2014 and 2024 applying AI techniques to effort estimation in software tasks and empirical or theoretical articles explicitly addressing AI applications in effort estimation.

Exclusion Criteria (EC). Studies that do not address effort estimation or software development, those that do not use AI, publications before 2014, and inaccessible articles.

After applying these criteria, 328 articles were selected for the quality assessment phase, focusing on their title and keywords to evaluate their contributions to effort estimation in software projects using AI and ensemble models.

3.2.2 Quality Assessment Checklist

To ensure the quality of the included studies, we applied the following questions:

Title. Is the title related to the review theme?

Keywords. Are the keywords relevant to the theme?

Abstract. Does the abstract clearly present the study's objective, development, and results?

Context. Is the study set in the context of task effort estimation using AI techniques?

Each question was scored: Yes (5 points), Partially (3 points), or No (0 points). Studies scoring at least 14 points were included in the review. After this phase, 66 studies were selected, representing the complete set used to evaluate effort estimation methods.

4 RESULTS AND DISCUSSION

Section 4.1 analyzes AI-based techniques like Machine Learning, Large Language Models, NLP, and Neural Networks, highlighting their application and selection for improving effort estimation (RQ1). Section 4.2 discusses the continued role of experts in adjusting or validating AI predictions, particularly in complex or data-scarce contexts (RQ2). Finally, Section 4.3 explores metrics and benchmarks for evaluating model effectiveness, emphasizing practices for accurate predictions (RQ3).

4.1 AI-Based Techniques for Task Effort Estimation

To address **RQ1**, this section explores commonly used AI-based techniques for task effort estimation in software projects. The studies emphasize the frequent use of hybrid models combining various machine learning approaches, including neural networks (NN) and natural language processing (NLP), leveraging their strengths for more accurate task effort estimation. These models overcome individual limitations, providing robust, adaptable solutions for the complex realities of modern software projects, where

a single method often falls short of delivering accurate estimations.

4.1.1 Artificial Intelligence and Machine Learning

AI and ML are significant for effort estimation in software projects, with distinct approaches impacting estimation accuracy. AI without ML involves techniques like rule-based systems and Bayesian networks, effective in modeling uncertainties and inter-dependent variables. Dragicevic et al. (2017) showcased Bayesian networks' ability to handle incomplete data and probabilistic inferences, making them suitable for uncertain data scenarios.

However, AI without ML lacks dynamic learning, limiting its use in evolving data contexts, such as modern software projects. Bayesian networks model agile tasks effectively but struggle with new, unanticipated inputs during initial modeling phases.

Conversely, ML is a widely used approach, with 62 of the 66 reviewed articles applying techniques like SVM, KNN, and decision trees to enhance estimation accuracy. Studies like Ali and Gravino (2019) highlight the superior performance of ANN and SVM in modeling complex nonlinear relationships and handling smaller, noisier datasets.

A notable trend is combining AI and ML to leverage their strengths. Bilgaiyan et al. (2019) explored hybrid models combining feedforward neural networks with rule-based models, achieving greater accuracy in various software development stages. Techniques like ensemble learning, as demonstrated by Azzeh et al. (2018), further improve estimates by reducing bias across multiple ML models.

The reviewed studies indicate that ML methods, particularly when integrated with other techniques, generally produce better results than purely AI-based methods. Neural networks and SVM remain highly effective, while hybrid models such as Bayesian networks and ensemble learning are emerging as promising solutions for accurate effort estimation.

These evolving approaches in effort estimation offer robust solutions for handling the complexity and variability of modern software projects. As Sarro et al. (2022) noted, the integration of AI and ML enables continuous improvement in predictions, overcoming traditional methods' limitations.

4.1.2 Large Language Models

Alhamed and Storer (2022) explore the use of LLMs, such as GPT, in effort estimation for software maintenance, demonstrating their efficiency in processing large volumes of textual data. They propose a hybrid

framework combining LLM predictions with human expert insights to enhance estimation accuracy.

The study highlights that while LLMs excel in automated large-scale data analysis, human experts are crucial for interpreting contextual nuances, especially in complex, dynamic maintenance tasks. This hybrid approach significantly improves estimation accuracy and adaptability, as confirmed by practical case studies across various maintenance scenarios.

The integration of LLMs with human expertise reduces errors in high variability contexts where traditional methods fall short. The study underscores the evolving role of LLMs in software engineering, emphasizing the balance between automation and human intervention for more reliable and nuanced predictions.

4.1.3 Natural Language Processing

The use of NLP techniques in task effort estimation for software projects has become prominent, with nine of the 66 reviewed articles employing various NLP approaches to improve accuracy. Techniques like topic modeling, as used by Yasmin (2024) with Latent Dirichlet Allocation (LDA), help reduce textual complexity by focusing on relevant topics. However, LDA may struggle with capturing nuanced details of complex tasks, leading to less accurate estimates in certain contexts.

Semantic embedding techniques, a subset of machine learning approaches like SBERT and Word2Vec, have also been widely applied. Yalçiner et al. (2024) demonstrate SBERT's capability to generate rich semantic representations, enhancing task description comparisons despite its high computational demands. Conversely, Dan et al. (2024) highlight Word2Vec's efficiency in identifying task similarities with lower computational costs, though it may be less effective in understanding complex sentences compared to SBERT.

Hybrid models combining NLP with other techniques, as explored by Dehghan et al. (2016), integrate NLP methods with ensemble learning and task metadata to provide comprehensive task views and robust estimates. Despite their effectiveness, these models can increase computational complexity and reduce result explainability. NLP's ability to process unstructured data, like natural language task descriptions, is invaluable, but challenges like the need for high-quality training data and the interpretability of complex model results persist.

4.1.4 Neural Networks

Neural networks are widely adopted in effort estimation for software projects, with 44 reviewed articles utilizing these techniques. Their strength lies in capturing the complexity of project data, modeling non-linear relationships, and identifying hidden patterns often missed by traditional methods. However, the high training costs in terms of time and computational resources are notable challenges. Commonly used techniques include Deep Neural Networks (DNNs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), including Long Short-Term Memory (LSTM) networks, and Cascade Correlation Neural Networks (CCNN), each with specific strengths and limitations.

DNNs are praised for their predictive accuracy with extensive data, but require considerable tuning and large volumes of training data, as shown by Ali and Gravino (2019). Radial Basis Function Neural Networks (RBFNNs), discussed by Nassif et al. (2016), offer faster training and are effective for well-defined clusters, though they may not perform as well with complex tasks. Cascade Correlation Neural Networks provide adaptability by adjusting their structure during training, reducing training time, and showing lower overfitting tendencies.

RNNs, particularly LSTM networks, are effective in modeling sequential data and capturing long-term dependencies, crucial for evolving projects. Jordan (2024) demonstrated LSTM's superior performance when optimized with particle swarm optimization. Despite their effectiveness, RNNs and LSTMs demand significant computational power and face challenges like the vanishing gradient problem. CNNs, though traditionally used in image processing, have been adapted for time series and textual data, proving useful in specific scenarios but less common in effort estimation compared to RNNs and LSTMs.

Overall, neural networks are highly adaptable and effective for handling large volumes of unstructured data, capturing complex patterns in software projects of varying scope and complexity. However, their training is resource-intensive, and the "black box" nature of deeper networks complicates result interpretation, posing challenges for project managers who require transparency in predictive models.

4.2 Use of Expert Opinion

To address RQ2, twelve articles incorporating expert opinions were selected, highlighting their role in hybrid systems for model validation or adjustment. While ML algorithms excel in identifying patterns

from large datasets, they often lack the context needed to account for human or circumstantial factors in effort estimation. Alhamed and Storer (2022) propose a hybrid framework combining language models and expert opinion, enhancing estimation accuracy. Similarly, Meiliana et al. (2023) and Dan et al. (2024) illustrate the integration of expert input to calibrate and adjust automated models, ensuring predictions align with project-specific nuances.

Despite advancements in automation, expert reliance remains significant, particularly in complex or early-stage projects where historical data may be insufficient for AI models. Experts provide critical contextual insights that automated systems often miss. Yasmin (2024) demonstrate the importance of expert adjustment in cases of significant project variations or ambiguous AI model outputs. This expert intervention bridges gaps in automated models, especially when project variables deviate from historical norms.

The main advantage of expert involvement is their ability to contextualize predictions based on emerging variables, enhancing estimation accuracy. However, it can also introduce subjectivity and potential delays in decision-making, as discussed by Meiliana et al. (2023). Despite ongoing advancements in AI, expert opinion remains crucial, particularly in scenarios where data scarcity or complexity challenges AI efficacy. The trend is toward integrating expert adjustments with automated predictions to enhance accuracy and reliability, though reliance on experts is expected to decrease as AI models evolve.

4.3 Metrics Used to Evaluate Model Performance

This section covers metrics commonly used to evaluate model performance in task effort estimation (RQ3).

Mean Magnitude of Relative Error (MMRE). Highlighted by Bilgaiyan et al. (2019) and Satapathy et al. (2014), MMRE measures the difference between predicted and actual effort.

Median Magnitude of Relative Error (MdMRE). Used alongside MMRE, it mitigates the impact of outliers, as noted by Zakrani et al. (2019).

PRED(X): Evaluates the percentage of predictions within a given error range, typically 25%, as seen in Sarro et al. (2022).

Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). These metrics measure the mean absolute and squared errors, sensitive to huge prediction errors, frequently used in Phan and Janesari (2022) for neural network performance analysis.

Comparing traditional models with AI and ML benchmarks, such as neural networks and ensemble learning, is crucial for assessing improvements in estimation accuracy over conventional methods like CO-COMO II and UCP (Cabral et al., 2023).

5 CONCLUSIONS

The application of AI techniques, particularly ML, has become a dominant approach for effort estimation in software projects, as evidenced by the majority of the 66 analyzed studies. ML's versatility and effectiveness in handling complex data make it a preferred choice among researchers.

Conversely, the adoption of AI models without ML, such as Bayesian networks, remains limited, and LLM usage is still in its early stages. A significant trend observed is the integration of hybrid models, which combine various techniques to produce more robust and accurate results.

Expert opinions remain to play a crucial role in certain studies, particularly for calibrating and validating estimates in complex scenarios. The integration of human expertise with AI models ensures that predictions are tailored to the specific nuances of each project, enhancing accuracy.

Evaluation metrics such as MMRE, MdMRE, and PRED(X) are critical for assessing model accuracy and verifying improvements over traditional methods. These metrics are instrumental in demonstrating the effectiveness of AI-based models in improving effort estimation accuracy in software projects.

ACKNOWLEDGEMENTS

We thank the Brazilian Army and its Army Strategic Program ASTROS for the financial support through the SIS-ASTROS GMF project (898347/2020).

REFERENCES

- Abdelali, Z., Mustapha, H., and Abdelwahed, N. (2019). Investigating the use of random forest in software effort estimation. In *Procedia Computer Science*, volume 148, pages 343–352. Elsevier B.V.
- Ahmad, F. B. and Ibrahim, L. M. (2022). Software effort estimation based on long short term memory and stacked long short term memory. In *2022 8th International Conference on Contemporary Information Technology and Mathematics, ICCITM 2022*, pages 165–170. Institute of Electrical and Electronics Engineers Inc.
- Albrecht, A. J. and Gaffney, J. E. (1983). Software function, source lines of code, and development effort prediction: A software science validation. *IEEE Transactions on Software Engineering*, (6):639–648.
- Alhamed, M. and Storer, T. (2022). Evaluation of context-aware language models and experts for effort estimation of software maintenance issues. In *Proceedings - 2022 IEEE International Conference on Software Maintenance and Evolution, ICSME 2022*, pages 129–138. Institute of Electrical and Electronics Engineers Inc.
- Ali, A. and Gravino, C. (2019). A systematic literature review of software effort prediction using machine learning methods.
- Alsaadi, B. and Saeedi, K. (2024). Ensemble effort estimation for novice agile teams. *Information and Software Technology*, 170.
- Arachchi, K. J. W. and Amalraj, C. R. (2023). An agile project management supporting approach for estimating story points in user stories. In *Proceedings of ICITR 2023 - 8th International Conference on Information Technology Research: The Next Evolution in Digital Transformation*. Institute of Electrical and Electronics Engineers Inc.
- Azzeh, M., Nassif, A. B., Banitaan, S., and Lopez-Martin, C. (2018). Ensemble of learning project productivity in software effort based on use case points. In *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018*, pages 1427–1431. Institute of Electrical and Electronics Engineers Inc.
- Bilgaiyan, S., Mishra, S., and Das, M. (2019). Effort estimation in agile software development using experimental validation of neural network models. *International Journal of Information Technology (Singapore)*, 11:569–573.
- Boehm, B. W., Horowitz, E., Madachy, R., Reifer, D., Clark, B., Steece, B., Brown, A. W., Chulani, S., and Abts, C. (2000). *Software cost estimation with CO-COMO II*. Prentice Hall PTR.
- Brar, P. and Nandal, D. (2022). A systematic literature review of machine learning techniques for software effort estimation models. In *Proceedings - 2022 5th International Conference on Computational Intelligence and Communication Technologies, CCICT 2022*, pages 494–499. Institute of Electrical and Electronics Engineers Inc.
- Cabral, J. T. H. A., Oliveira, A. L., and da Silva, F. Q. (2023). Ensemble effort estimation: An updated and extended systematic literature review. *Journal of Systems and Software*, 195.
- Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T., Ghose, A., and Menzies, T. (2019). A deep learning model for estimating story points. *IEEE Transactions on Software Engineering*, 45:637–656.
- Dan, I., Rusu, C., and Oliver, O. (2024). An nlp approach to estimating effort in a work environment. In *Proceedings of the 2024 International Conference on Software Effort Estimation, Cluj-Napoca, Romania*. IEEE.
- Dantas, E., Costa, A., Vinicius, M., Perkusich, M.,

- Almeida, H., and Perkusich, A. (2019). An effort estimation support tool for agile software development: An empirical evaluation. In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE*, volume 2019-July, pages 82–87. Knowledge Systems Institute Graduate School.
- Dantas, E., Perkusich, M., Dilorenzo, E., Santos, D. F., Almeida, H., and Perkusich, A. (2018). Effort estimation in agile software development: An updated review. In *International Journal of Software Engineering and Knowledge Engineering, SEKE*, volume 28, pages 1811–1831. World Scientific Publishing Co. Pte Ltd.
- de A. Cabral, J. T. H., de A. Araujo, R., Nobrega, J. P., and de Oliveira, A. L. (2017). Heterogeneous ensemble dynamic selection for software development effort estimation. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 210–217.
- Dehghan, A., Blincoe, K., and Damian, D. (2016). A hybrid model for task completion effort estimation. In *SWAN 2016 - Proceedings of the 2nd International Workshop on Software Analytics, co-located with FSE 2016*, pages 22–28. Association for Computing Machinery, Inc.
- Dragicevic, S., Celar, S., and Turic, M. (2017). Bayesian network model for task effort estimation in agile software development. *Journal of Systems and Software*, 127:109–119.
- Goto, T. (2016). *ACIT-CSII-BCD 2016 : 4th International Conference on Applied Computing and Information Technology (ACIT 2016) ; 3rd International Conference on Computational Science/Intelligence and Applied Informatics (CSII 2016), 1st International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD 2016) : proceedings : 12-14 December 2016, Las Vegas, Nevada, USA*. Conference Publishing Services, IEEE Computer Society.
- Gultekin, M. and Kalipsiz, O. (2020). Story point-based effort estimation model with machine learning techniques. *International Journal of Software Engineering and Knowledge Engineering*, 30:43–66.
- Hoda, M. N. (2023). *Proceedings of the 17th INDIACom; 2023 10th International Conference on Computing for Sustainable Global Development (INDIACom) : (15-17 March 2023)*. India-Com.
- Hosni, M. (2023). On the value of combiners in heterogeneous ensemble effort estimation. In *International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K - Proceedings*, volume 1, pages 153–163. Science and Technology Publications, Lda.
- Idri, A., Hosni, M., and Abran, A. (2016). Improved estimation of software development effort using classical and fuzzy analogy ensembles. *Applied Soft Computing Journal*, 49:990–1019.
- Jordan, A. E. (2024). An optimized lstm neural network for accurate estimation of software development effort. *Mathematics*, 12.
- Iqbal, M., Ijaz, M., Mazhar, T., Shahzad, T., Abbas, Q., Ghadi, Y. Y., Ahmad, W., and Hamam, H. (2024). Exploring issues of story-based effort estimation in agile software development (asd). *Science of Computer Programming*, 236.
- Iwata, K., Nakashima, T., Anan, Y., and Ishii, N. (2017). Machine learning classification to effort estimation for embedded software development projects. *International Journal of Software Innovation*, 5:19–32.
- Karner, G. (1993). Metrics for object-oriented system development. Technical report, Objectory Systems Developer Report.
- Kassaymeh, S., Alweshah, M., Al-Betar, M. A., Hammouri, A. I., and Al-Ma'aitah, M. A. (2024). Software effort estimation modeling and fully connected artificial neural network optimization using soft computing techniques. *Cluster Computing*, 27:737–760.
- Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., and Linkman, S. (2009). Systematic literature reviews in software engineering – a systematic literature review. *Information and Software Technology*, 51(1):7–15. Special Section - Most Cited Articles in 2002 and Regular Research Papers.
- Kitchenham, B. A. (2004). Procedures for performing systematic reviews.
- Kumar, A. (2024). Recommendation of machine learning techniques for software effort estimation using multi-criteria decision making. *Journal of Universal Computer Science*, 30:221–241.
- Kumar, K. H. and Srinivas, K. (2023). An accurate analogy based software effort estimation using hybrid optimization and machine learning techniques. *Multimedia Tools and Applications*, 82:30463–30490.
- Kumar, S., Arora, M., Sakshi, and Chopra, S. (2022). A review of effort estimation in agile software development using machine learning techniques. In *4th International Conference on Inventive Research in Computing Applications, ICIRCA 2022 - Proceedings*, pages 416–422. Institute of Electrical and Electronics Engineers Inc.
- Lavingia, K., Patel, R., Patel, V., and Lavingia, A. (2024). Software effort estimation using machine learning algorithms. *Scalable Computing*, 25:1276–1285.
- Marapelli, B., Carie, A., and Islam, S. M. (2021). Software effort estimation with use case points using ensemble machine learning models. In *International Conference on Electrical, Computer, and Energy Technologies, ICECET 2021*. Institute of Electrical and Electronics Engineers Inc.
- Meiliana, Daniella, G., Wijaya, N., Putra, N. G. E., and Efata, R. (2023). Agile software development effort estimation based on product backlog items. In *Procedia Computer Science*, volume 227, pages 186–193. Elsevier B.V.
- Najm, A. and Zakrani, A. (2023). On the interpretability of a tree-based ensemble for predicting software effort. In *Colloquium in Information Science and Technology, CIST*, pages 129–134. Institute of Electrical and Electronics Engineers Inc.
- Nakamura, M. (2019). *Proceedings, 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed*

- Computing (SNPD 2019) : July 8-11, 2019, Toyama, Japan*. IEEE Computer Society.
- Nassif, A. B., Azzeh, M., Capretz, L. F., and Ho, D. (2016). Neural network models for software development effort estimation: a comparative study. *Neural Computing and Applications*, 27:2369–2381.
- Phan, H. and Jannesari, A. (2022). Heterogeneous graph neural networks for software effort estimation. In *International Symposium on Empirical Software Engineering and Measurement*, pages 103–113. IEEE Computer Society.
- Potolea, R. and Slavesescu, R. R. (2017). *Proceedings, 2017 IEEE 13th International Conference on Intelligent Computer Communication and Processing (ICCP) : Cluj-Napoca, Romania, September 7-9, 2017*. IEEE.
- Rahman, M., Sarwar, H., Kader, A., Goncalves, T., and Tin, T. T. (2024). Review and empirical analysis of machine learning-based software effort estimation. *IEEE Access*, 12:85661–85680.
- Ramessur, M. A. and Nagowah, S. D. (2021). A predictive model to estimate effort in a sprint using machine learning techniques. *International Journal of Information Technology (Singapore)*, 13:1101–1110.
- Rankovic, N. and Rankovic, D. (2024). Power of lstm and shap in the use case point approach for software effort and cost estimation. In *2024 IEEE 22nd World Symposium on Applied Machine Intelligence and Informatics, SAMI 2024 - Proceedings*, pages 59–64. Institute of Electrical and Electronics Engineers Inc.
- Sanchez, E. R., Maceda, H. C., and Santacruz, E. V. (2022). Software effort estimation for agile software development using a strategy based on k-nearest neighbors algorithm. In *2022 IEEE Mexican International Conference on Computer Science, ENC 2022 - Proceedings*. Institute of Electrical and Electronics Engineers Inc.
- Sarro, F., Moussa, R., Petrozziello, A., and Harman, M. (2022). Learning from mistakes: Machine learning enhanced human expert effort estimates. *IEEE Transactions on Software Engineering*, 48:1868–1882.
- Satapathy, S. M., Acharya, B. P., and Rath, S. K. (2014). Class point approach for software effort estimation using stochastic gradient boosting technique. *ACM SIGSOFT Software Engineering Notes*, 39:1–6.
- Sharma, A. and Chaudhary, N. (2022a). Analysis of software effort estimation based on story point and lines of code using machine learning. *International Journal of Computing and Digital Systems*, 12:131–140.
- Sharma, A. and Chaudhary, N. (2022b). Prediction of software effort by using non-linear power regression for heterogeneous projects based on use case points and lines of code. In *Procedia Computer Science*, volume 218, pages 1601–1611. Elsevier B.V.
- Song, L., Minku, L. L., and Yao, X. (2018). A novel automated approach for software effort estimation based on data augmentation. In *ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 468–479. Association for Computing Machinery, Inc.
- Sousa, A. O., Veloso, D. T., Goncalves, H. M., Faria, J. P., Mendes-Moreira, J., Graca, R., Gomes, D., Castro, R. N., and Henriques, P. C. (2023). Applying machine learning to estimate the effort and duration of individual tasks in software projects. *IEEE Access*, 11:89933–89946.
- Sudarmaningtyas, P. and Mohamed, R. (2021). A review article on software effort estimation in agile methodology.
- Sunda, N. and Sinha, R. R. (2023). Optimizing effort estimation in agile software development: Traditional vs. advanced ml methods. In *2023 International Conference on Communication, Security and Artificial Intelligence, ICCSAI 2023*, pages 487–494. Institute of Electrical and Electronics Engineers Inc.
- Usman, M., Mendes, E., Weidt, F., and Britto, R. (2014). Effort estimation in agile software development: A systematic literature review. In *ACM International Conference Proceeding Series*, pages 82–91. Association for Computing Machinery.
- Vyas, M. and Hemrajani, N. (2021). Predicting effort of agile software projects using linear regression, ridge regression and logistic regression. *International Journal on "Technical and Physical Problems of Engineering" (IJTPE) Issue*, 47:14–19.
- Yalçiner, B., Dinçer, K., Karaçor, A. G., and Efe, M. O. (2024). Enhancing agile story point estimation: Integrating deep learning, machine learning, and natural language processing with sbert and gradient boosted trees. *Applied Sciences*, 14(16):7305.
- Yasmin, A. (2024). Cost adjustment for software crowd-sourcing tasks using ensemble effort estimation and topic modeling. *Arabian Journal for Science and Engineering*, 49:12693–12728.
- Zakrani, A., Hain, M., and Idri, A. (2019). Improving software development effort estimation using support vector regression and feature selection. *IAES International Journal of Artificial Intelligence*, 8:399–410.