khazanah informatika

Jurnal Ilmu Komputer dan Informatika

The Implementation of Machine Learning for Software Effort Estimation: A Literature Review

Eva Hariyanti[1*], Mirtha Aini Paradista[1], Maria Lauda Joel Goyayi[2], Arthalia[1], Detria Azka Shabirina[1], Endang Nurjanah[1], Oktavia Intifada Husna[1], Fakhrana Almas Syah Yahrani[1]

[1]Department of Information Systems
Universitas Airlangga
Surabaya, Indonesia
[2]Department of Business and IT Studies
Mzumbe University
Dar Es Salaam, Tanzania
*eva.hariyanti@fst.unair.ac.id

Abstract-Effort estimation is pivotal for the triumph of software development endeavors. The appropriate forecasting approach is vital for aligning software project effort estimation outcomes. This process aids in efficiently distributing resources, charting project strategies, and facilitating informed choices in IT Project Management. Machine learning, a facet of artificial intelligence (AI), is dedicated to crafting algorithms and models that empower computers to enhance their performance based on data and facilitate predictions or decision-making. This study discusses the implementation of machine learning in software development effort estimation by highlighting the advantages of ensemble techniques We collected 558 relevant papers on software effort estimation and machine learning techniques. After a quality review process, we identified 40 articles for in-depth review. The study result shows that using ensemble techniques in supervised and unsupervised learning can improve the accuracy of software effort estimation. Artificial Neural Networks, Regression, K-Nearest Neighbors, Decision Trees, Random Forest, and Bootstrap Aggregation are the most commonly used methods. The study also indicates that most articles use ensemble techniques for tuning parameters, selecting features, and weighting features. This study provides insights into implementing machine learning techniques to estimate software effort and highlights the advantages of ensemble techniques.

Keywords: artificial intelligence, ensemble technique, governance, IT project management, software effort estimation

Article info: *submitted September 21, 2023, revised March 19, 2024, accepted March 20, 2024*

1. Introduction

Effort estimation plays a crucial role in the success of software development projects. Effort estimation is required to provide transparent information about relevant business factors and their interrelationships to support decision-makers in understanding what might happen, why it might happen, and what can be done to prevent it. Determining the most suitable prediction method is essential to agree between the results obtained in project effort estimation [1]. In IT governance, effort estimation primarily falls under IT Project Management. Project management involves applying knowledge, skills, tools, and techniques to project activities to meet requirements [2]. Effort estimation supports effective resource allocation, project planning, and decision-making in IT Project Management.

Machine learning is a field of artificial intelligence (AI) that focuses on developing algorithms and models that enable computers to optimize performance criteria from data and make predictions or decisions without being explicitly programmed. Machine learning aids humans in finding solutions to various problems, such as computer vision, speech recognition, and robotics. Machine learning uses statistical theory to build mathematical models since its primary task is to make inferences from a sample [3]. Ensemble learning is one of the machine learning approaches used in software effort estimation processes [4]. Ensemble learning, or ensemble methods, are machine learning algorithms that build a collection of classifiers and then classify new data by taking a vote from their predictions. The original ensemble method is Bayesian averaging, but more recent ensemble algorithms include output coding that corrects errors, bagging, and boosting [5]. Ensemble techniques have been widely used to improve the results of machine learning, not only in cases of software effort estimation but also in other cases such as language identification as in the study [6].

In the era of big data, machine learning is considered a precious and efficient tool for processing large amounts of data, often likened to a 'work-horse' in tackling the challenges presented in this era [7]. There are three types of machine learning: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning algorithms, knowledge acquired from past and current data is utilized with labels to predict events. This approach begins with training a dataset, where machine learning develops functions that can infer and estimate output values [8]. Examples of supervised learning algorithms include neural networks, gradient boosting, support vector regression, decision tree regression, and others. Unsupervised learning refers to the network's ability to learn to discover patterns in input data without prior guidance or labels. In this process, the network learns to cluster or identify numerical patterns within a set of input data without a specific goal of reproducing the entire numerical arrangement [9]. Clustering is one example of an unsupervised learning algorithm since it is typically used to divide images into two groups or clusters based on inherent characteristics within the images, such as color, size, shape, and so on. Reinforcement learning is learning to map situations into actions that can yield maximum numerical rewards [10]. This algorithm enables computers to learn independently from their environment through agents. As a result, computers can conduct their exploration through interactions with the environment.

Software effort estimation estimates the human resources, time, and cost required to develop or maintain software. Its goal is to provide accurate estimates of the effort needed for a software project to succeed. In some cases, software effort estimation involves the use of machine learning. There are two machine learning estimation techniques in its implementation: solo and ensemble. According to [11]controlling, and delivering a successful software project within budget and schedule. The overestimation and underestimation both are the key challenges for future software development, henceforth there is a continuous need for accuracy in software effort estimation. The researchers and practitioners are striving to identify which machine learning estimation technique gives more accurate results based on evaluation measures, datasets and other relevant attributes. The authors of related research are generally not aware of previously published results of machine learning effort estimation techniques. The main aim of this study is to assist the researchers to know which machine learning technique yields the promising effort estimation accuracy prediction in software development. In this article, the performance of the machine learning ensemble and solo techniques are investigated on publicly and non-publicly domain datasets based on the two most commonly used accuracy evaluation metrics. We used the systematic literature review methodology proposed by Kitchenham and Charters. This includes searching for the most relevant papers, applying quality assessment (QA, the scientific literature used ensemble and solo techniques in estimating software effort.

Research [12] proposed using Optimal Tree Ensemble (OTE) to predict software development efforts by combining top-ranked trees one by one from Random Forests. The OTE model outperformed other techniques such as regression trees, random forests, RBF neural networks, support vector regression, and multiple linear regression measured with mean magnitude relative error (MMRE), MdMRE, and Pred(l) accuracy matrices using five well-known datasets: ISBSG R8, COCOMO, Tukutuku, Desharnais, and Albrecht.

Research [13] systematically reviewed recent studies that utilized and discussed software effort estimation models using machine learning techniques. The research showed that Artificial Neural Networks (ANN) as a machine learning model, NASA as the dataset, and the Mean Magnitude Relative Error (MMRE) were the most commonly used accuracy measures in specific studies. ANN and Support Vector Machine (SVM) were two techniques that outperformed other machine learning techniques. Regression techniques were the most commonly used non-machine learning methods, among others. Additionally, the combination of SVM and regression yielded better predictions when compared to other machine learning and non-machine learning techniques.

A study [14] cost, and duration of their projects. As evident by Standish group chaos manifesto that approx. 43{\\%} of the projects are often delivered late and entered crises because of overbudget and less required functions. Improper and inaccurate estimation of software projects leads to a failure, and therefore it must be considered in true letter and spirit. When Agile principle-based process models (e.g., Scrum reveals that algorithms optimized with Neural Networks are the best machine learning technique, measured by the MMRE accuracy metric of 2.93%. Ensemble estimation techniques in this study outperform single estimation techniques. Furthermore, estimation accuracy can be improved by combining them with metaheuristic algorithms. A study [15] shows that the Gradient Boost Regressor (GBR) model is the best ensemble model for estimating effort. According to the research, the GBR technique achieves a regression score (R2 Score) of 99%. This indicates that GBR is the most accurate technique compared to other ensemble techniques tested in the study, such as the Random Forest Regressor (RFR). Study [11]controlling, and delivering a successful software project within budget and schedule. The overestimation and underestimation both are the key challenges for future software development, henceforth there is a continuous need for accuracy in software effort estimation. The researchers and practitioners are striving to identify which machine learning estimation technique gives more accurate results based on evaluation measures, datasets and other relevant

attributes. The authors of related research are generally not aware of previously published results of machine learning effort estimation techniques. The main aim of this study is to assist the researchers to know which machine learning technique yields the promising effort estimation accuracy prediction in software development. In this article, the performance of the machine learning ensemble and solo techniques are investigated on publicly and non-publicly domain datasets based on the two most commonly used accuracy evaluation metrics. We used the systematic literature review methodology proposed by Kitchenham and Charters. This includes searching for the most relevant papers, applying quality assessment (QA discusses the accuracy performance of software estimation in 35 selected studies, including 17 ensemble and 18 solo techniques, using MMRE and PRED(25) as evaluation criteria. Two datasets were used: Non-Public Domain (NPD) and Public Domain (PD). According to the research, ensemble techniques produce more accurate effort estimates than solo techniques. This is because ensemble techniques combine appropriate rules and methods to predict software effort estimates. The study [16]especially in the financial resources available for the project as well as the time required to complete it. As a result of this, the research community has developed different methods for estimating effort in software projects in the hope of achieving high levels of accuracy and efficiency in the use of available resources. Among those methods that have proven to be accurate in estimating the effort of software projects is the use of machine learning (ML investigates the benefits of using the stacking ensemble method in software effort estimation. The International Software Benchmarking Standards Group (ISBSG) dataset is used in this research. A comparison is made between individual methods (M5P method) and the stacking ensemble method, which is a combination of different regression models, including Gradient Boosting Regression (GBReg), Linear Support Vector Regression (LinearSVR), and Random Forest Regression (RFR). The results of this study show that the ensemble method improves accuracy with a Mean Absolute Error (MAE) value of 0.0383, while the M5P method demonstrates an MAE value of 0.0612.

Several previous literature review studies have discussed the comparison of machine learning techniques. While this study examines the implementation of the most commonly used ensemble machine learning techniques in software effort estimation. The objectives and novelty of this paper are to provide an overview of the utility of each type of machine learning and the benefits of ensemble techniques in software effort estimation, which can be used as a reference to determine the appropriate method.

The rest of the paper is organized as follows. Section 2 describes the research method. Section 3 reports the results. Section 4 discusses analysis and provisions for future work. Finally, Section 5 presents our conclusions.

2. Methods

We adopted the PRISMA method as a guideline in writing this SLR, as used in [17]. Our SLR comprises three stages: review planning, conducting, and reporting.

a. Review planning

In the planning stage of our review, we determine the objectives and research questions, establish the search strategy, and define inclusion and exclusion criteria.

1) Objectives and Research Question
   This study investigates the implementation of ensemble machine-learning techniques in software effort estimation. We formulated a research question regarding how the implementation of machine learning techniques in software effort estimation.
2) Search Strategy
   We gathered relevant studies by defining search keywords, electronic databases, search parameters, and reporting formats. The keywords we used were software effort estimation and machine learning techniques. We compiled alternative search keywords: (software effort estimation OR effort estimation) AND (machine learning OR machine learning techniques OR machine learning methods). The search for relevant studies in electronic databases was restricted to the last five years (2018-2023) and included research articles published in English-language journals. We examined electronic databases: SCOPUS, ScienceDirect, Web of Science, Google Scholar, and IEEE Xplore.
3) Inclusion and Exclusion Criteria
   Table I shows the inclusion and exclusion criteria we use to determine relevant studies.

Table 1. Inclusion and Exclusion Criteria

Inclusion Criteria	Exclusion Criteria
1. Using machine learning techniques for software effort estimation. 2. Using a Hybrid Model of Combination Techniques for Software Effort Estimation 3. A comparative study of the software effort estimation model. 4. Use at least two public datasets for software engineering.	1. Inappropriate title 2. Inappropriate abstract/ keywords 3. Focus on something other than estimating software effort. Only do software size estimates, schedules, and times. (No effort estimation). 4. Duplicate paper 5. Older than January 2018

b. Conducting the review

At this stage, we search and select articles based on our determined quality criteria.

1) Study Search and Selection

We searched for relevant studies in electronic databases using alternative keywords and recorded the findings in a spreadsheet. Some databases allowed downloading XLSX files, which facilitated the review process. The research selection process involved checking keywords, duplicate studies, titles, and abstracts. The findings were filtered using inclusion and exclusion criteria, followed by quality assessment.

The electronic database search yielded the following results: 1065 studies from SCOPUS, 452 studies from ScienceDirect, 243 studies from Web of Science, 93 studies from IEEE Xplore, and 17,600 studies from Google Scholar. The initial selection process involved removing duplicate studies, resulting in 558 studies for further consideration, which were documented in the spreadsheet to aid in the selection process. Relevant studies were chosen based on their titles and abstracts, resulting in 288 and 96 relevant studies, respectively. Applying inclusion and exclusion criteria, we obtained 41 relevant studies to ensure they met the requirements for conducting a Systematic Literature Review (SLR). All acquired studies were assessed for quality based on the criteria, resulting in 40 relevant studies. These studies were used as the primary research material. The process of searching and selecting studies can be seen in Figure 1.

2) Quality Assessment

This study employs a quality assessment to evaluate the quality of the primary research. All selected research will be assessed based on the inclusion and exclusion criteria in Table 2.

Four assessment criteria are used for evaluation:

1. (QA1) assesses the purpose of the study.
2. (QA2) assesses the detailed description of the proposed solution in the study.
3. (QA3) assesses the validation of research solutions.
4. (QA4) assesses the continuity between models, interpretations, and conclusions.

Furthermore, all studies will be given scores according to the assessment criteria. Studies with an assessment score of ≧ 3 will be used for the main study.

3) Data Extraction and Synthesis

We collect the data by formulating relevant information in research goals, methods, datasets, results, discussions, and conclusions.

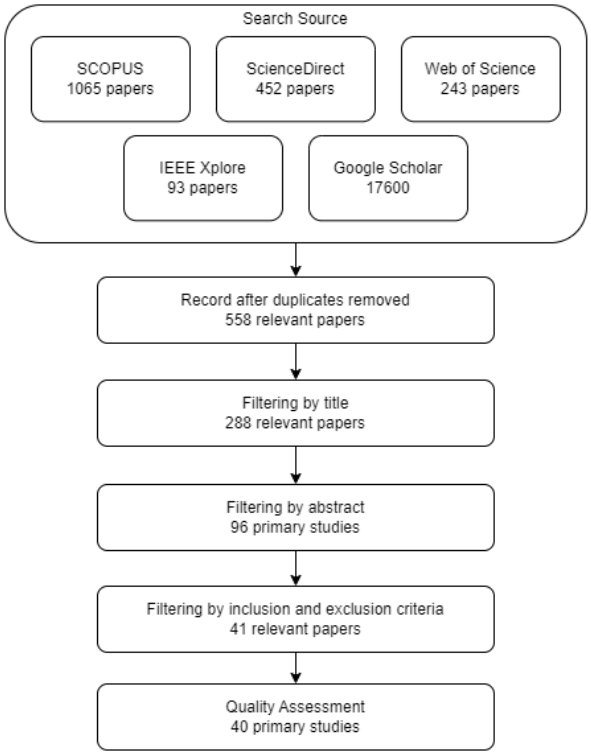

Figure 1. ***Study Search and Selection Process.***

c. Reporting the review

The review results were reported by describing the summary of the studies and answering RQ. The description of RQ was based on the data extraction results.

3. Results

This section explains the review findings and comprehensive discussions related to RQ.

We involved 40 studies on software effort estimation using various techniques that have passed the quality assessment stage with a score of □ 3, listed in Table 3.

We categorize fundamental machine learning techniques employed or examined in each study. Table 4 and Figure 2 present the results of the research categorization based on basic machine-learning techniques. It can be seen that most machine learning techniques used for software effort estimation are in supervised learning.

Table 2. Quality Assessment Criteria

Criteria	Score	Description
(QA1) Is there a clear statement regarding the purpose of the research?	-1	There is no statement regarding the purpose of the research.
	0	The statement regarding the purpose of the research is not clear.
	1	There is a clear statement regarding the purpose of the research.
(QA2) Is there a detailed explanation of the proposed solution in the research?	-1	There is no detailed explanation of the proposed solution.
	0	The explanation of the proposed solution is not clear, reading from other resources is needed.
	1	There is a detailed explanation of the proposed solution.
(QA3) Has the proposed solution been validated?	-1	The proposed solution has not been validated.
	0	Not all of the proposed solution has been validated.
	1	The proposed solution has been validated.
(QA4) Is there continuity between the research's models, interpretations, and conclusions?	-1	There is no continuity between models, interpretations, and conclusions in the research.
	0	There is continuity, but not all three.
	1	There is continuity between models, interpretations, and conclusions in the research.

Table 3. Article Quality Scores

Research	(QA1)	(QA2)	(QA3)	(QA4)	Score
[15]	1	1	1	1	4
[16]	1	1	1	1	4
[18]	1	1	1	1	4
[19]	1	1	1	1	4
[20]	1	1	1	1	4
[21]	0	1	1	1	3
[22]	1	1	1	1	4
[23]	1	1	1	1	4
[24]	1	1	1	1	4
[25]	1	1	1	1	4
[26]	0	1	1	1	3
[27]	1	1	1	1	4
[28]	1	1	1	1	4
[29]	1	1	1	1	4
[30]	1	1	1	1	4
[31]	1	1	1	1	4
[32]	1	1	1	1	4
[33]	1	1	1	1	4
[34]	1	1	1	1	4
[35]	1	1	1	1	4
[36]	1	1	1	1	4
[37]	1	1	1	1	4
[38]	1	1	1	1	4
[39]	1	1	1	1	4
[40]	1	1	1	1	4
[41]	1	1	1	1	4
[42]	1	1	1	1	4
[43]	0	1	1	1	3
[44]	1	1	1	1	4
[45]	1	1	1	1	4

Research	(QA1)	(QA2)	(QA3)	(QA4)	Score
[46]	1	1	1	1	4
[47]	0	1	1	1	3
[48]	1	1	1	1	4
[49]	1	1	1	1	4
[50]	1	1	1	1	4
[51]	1	1	1	1	4
[52]	1	1	1	1	4
[53]	1	1	1	1	4
[54]	1	1	1	1	4
[55]	1	1	1	1	4

Table 4. Categorization of Articles Based on Types of Machine Learning

Machine Learning	Articles
Supervised Learning	[15], [18], [20], [21], [22], [23]
Unsupervised Learning	[18], [23]
Reinforcement Learning	[16]

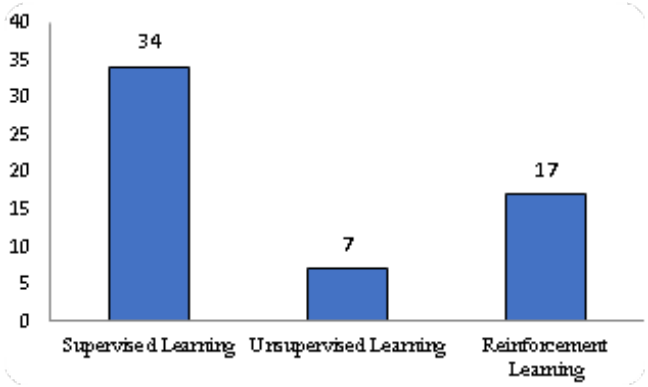

Figure 2. The Number of Articles Based on Machine Learning Technique Categorization

Furthermore, we reviewed ensemble techniques commonly employed in supervised and unsupervised learning to examine the purposes behind their usage. We found that ensemble techniques are used for six purposes: determining training data, reducing iterations, weighting features, selecting features, tuning parameters, and reducing sensitivity to data issues such as missing data, variance data, noise, and outliers.

Table 5 shows the articles that utilize ensemble techniques for these six purposes. In Figure 3, it can be seen that the top three purposes of using ensemble techniques are for tuning parameters, selecting features, and weighting features.

Table 5. Categorization of Articles Based on the Purposes of Ensemble Techniques

Ensemble Purpose	Articles
Tuning Parameters	[15], [16], [21], [23], [25], [26], [28], [29], [32], [33], [34], [43], [44], [45], [46], [49], [50], [52], [54]
Selecting Features	[18], [27], [28], [32], [33], [36], [38], [44], [46], [48], [53]
Weighting Features	[19], [20], [30], [31], [35], [36], [37], [46], [51], [55]
Reducing Sensitivity to Data Issues	[24], [26], [27], [28], [39], [40], [41], [42], [46]
Reducing Iterations	[43], [47]
Determining Training Data	[22]

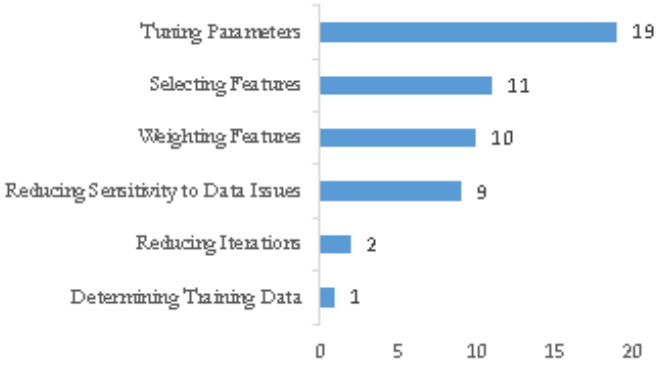

Figure 3. The Number of Articles Based on The Purposes of Using Ensemble Technique

4. Discussion

In supervised learning methods, commonly used techniques include Neural Networks, Regression, K-Nearest Neighbor (KNN), Classification and Regression Tree, and Support Vector Machine (SVM). These supervised learning techniques are often combined with Taguchi Orthogonal Array, bio-inspired algorithms, heterogeneous ensembles, ROME (Rapid Optimizing Methods for Estimation), and metaheuristic algorithms to improve accuracy. Ensemble techniques can enhance the accuracy of prediction models. For instance, research [20] shows that more complex NN architectures have a higher convergence rate. In this study, the best ensemble technique is ANN-L36 (ANN based on Taguchi Orthogonal Array L) because it requires only a few iterations with a lower MMRE than other ensemble techniques.

Furthermore, based on research [18], the average accuracy of the ensemble method Support Vector Regression (SVR) (with bio-inspired algorithms can also predict software effort estimation better than the SVR model itself after being evaluated with MMRE and Pred [25]. These results are attributed to the bio-inspired algorithms that assist in selecting relevant features in the training data. Research [32] has demonstrated that heterogeneous ensemble methods are more accurate than non-ensemble techniques. The findings of this research indicate that a combination of three classification techniques consisting of K-Nearest Neighbor (KNN), Support Vector Regression (SVR), and Decision Tree, named KSD, has 6.26% higher accuracy compared to a regular Decision Tree when evaluated using SA. However, some studies indicate that using ensemble techniques results in lower accuracy than non-ensemble techniques. For example, in research [39], an ensemble technique comprising SVM, Multi-Layer Perceptron (MLP), and Generalized Linear Models (GLM) had 7.47% lower accuracy after evaluation with PRED (0.25) compared to SVM as a solo technique, attributed to algorithm overhead. Based on these research findings, it is evident that the use of ensemble techniques in supervised machine learning can vastly improve the accuracy of software effort estimation prediction when compared to non-ensemble techniques.

In unsupervised learning, commonly used techniques include Random Forest and Bootstrap Aggregating. Research [42] indicates that ensemble learning with bootstrap aggregation principles like Bagging and RF provides the best overall performance. This is because the optimization process involves using each algorithm's grid search techniques for hyperparameters and data preprocessors. Research [18] shows that ensemble methods using random forest algorithms inspired by biology improve software effort estimation predictions better than single random forest models. This is because bio-inspired algorithms assist in selecting relevant features within the training data.

In reinforcement learning, machine-learning techniques commonly used include case-based reasoning, genetic algorithms, and fuzzy learning. The combination of CBR and GA has improved the accuracy of software effort estimation models. Research [16] especially in the financial resources available for the project as well as the time required to complete it. As a result of this, the research community has developed different methods for estimating effort in software projects in the hope of achieving high levels of accuracy and efficiency in the use of available resources. Among those methods that have proven to be accurate in estimating the effort of software projects is the use of machine learning (ML demonstrates that applying genetic algorithms to CBR in the CBR-GA model can increase win values and reduce loss values compared to the classic CBR method. This occurs because genetic algorithms can help determine more optimal combinations of weights, distances, and k-values. According to research [16] especially in the financial resources available for the project as well as the time required to complete it. As a result of this, the research community has developed different methods for estimating effort in software projects in the hope of achieving high levels of accuracy and efficiency in the use of available resources. Among those methods that have proven to be accurate in estimating the effort of software projects is the use of machine learning (ML, GA also aids in determining parameters in feature selection. Fuzzy

logic is widely combined with Model-Based Expectation-Maximization (EM) clustering and Firefly optimization. According to research [54] it is not frequently possible to antedate the exact guesses in the estimation of software development effort. There are many techniques used for effort estimation. But we cannot confirm that one particular method alone gives good accuracy in estimates. In this expose, a hybrid process is gracefully boosted for the estimation of the effort of software project. The innovative process is unknown; but consolidation of the fuzzy analogy by the side of the firefly and the Expectation-Maximization (EM, the Firefly algorithm provides an optimal set of rules to enhance accuracy in the estimation process.

In this study, we also identified several future works that need to be addressed. First is how machine learning methods can address imbalanced data issues in estimating software effort. Second, how to identify the most influential ensemble methods in improving the accuracy of software effort estimation models. Third, there is a need for an analysis of factors influencing the implementation of ensemble methods in enhancing the accuracy of effort estimation models

5. Conclusion

The study results indicate that ensemble techniques in supervised learning can enhance the accuracy of software effort estimation predictions compared to non-ensemble techniques. Ensemble techniques are predominantly used for tuning parameters, selecting features, and weighting features. Supervised learning methods are typically employed to select relevant features in training data. Unsupervised learning methods in software effort estimation are widely used for feature selection, hyperparameter tuning, and data preprocessing to optimize the model, one of which is by applying grid search techniques. Reinforcement learning is used to determine feature selection parameter values to ascertain the relevance and contribution of data to software effort estimation. The most common machine learning methods used in research on software effort estimation are Artificial Neural Networks, Regression, K-Nearest Neighbors, Decision Trees, Random Forest, and Bootstrap Aggregation.

We have identified several future works that need to be addressed, which relates to overcoming data imbalance problems, identifying ensemble methods that are most influential in improving the accuracy of estimation models, and analyzing factors that influence the implementation of ensemble methods

References

[1] A. Trendowicz and R. Jeffery, *Software project effort estimation: Foundations and best practice guidelines for success,* 2014. [Online]. Available: http://link.springer.com/content/pdf/10.1007/978-3-319-03629-8.pdf

[2] S. Kathy, *Information Technology Project Management*, p. 643, 2016.

[3] E. Alpadyn, *Introduction to Machine Learning,* The MIT Press, 2014.

[4] O. Malgonde and K. Chari, *An ensemble-based model for predicting agile software development effort,* vol. 24, no. 2. 2019. doi: 10.1007/s10664-018-9647-0.

[5] T. G. Dietterich, "Ensemble methods in machine learning", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1857 LNCS, pp. 1–15, 2000, doi: 10.1007/3-540-45014-9_1.

[6] A. Abdiansah and M. Q. Rizqie, "Automatic Language Identification For Indonesia-Malaysian Language Using Machine Learning", *Khazanah Informatika : Jurnal Ilmu Komputer dan Informatika,* vol. 9, no. 2, pp. 104–110, 2023.

[7] I. El Naqa and M. J. Murphy, *Machine Learning in Radiation Oncology,* pp. 3–11, 2015, doi: 10.1007/978-3-319-18305-3.

[8] R. Saravanan and P. Sujatha, "A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification", *Proceedings of the 2nd International Conference on Intelligent Computing and Control Systems, ICICCS 2018,* no. Iciccs, pp. 945–949, 2018, doi: 10.1109/ICCONS.2018.8663155.

[9] H. U. Dike, Y. Zhou, K. K. Deveerasetty, and Q. Wu, "Unsupervised Learning Based On Artificial Neural Network: A Review", *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS 2018)*, pp. 322–327, 2018, doi: 10.1109/CBS.2018.8612259.

[10] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction,* The MIT Press, 2020.

[11] Y. Mahmood, N. Kama, A. Azmi, A. S. Khan, and M. Ali, "Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation", *Software: Practice and Experience,* vol. 52, no. 1, pp. 39–65, 2022, doi: 10.1002/spe.3009.

[12] A. Zakrani, A. Idri, and M. Hain, *Software Effort Estimation Using an Optimal Trees Ensemble: An Empirical Comparative Study,* vol. 146. Springer International Publishing, 2020. doi: 10.1007/978-3-030-21005-2_7.

[13] A. Ali and C. Gravino, "A systematic literature review of software effort prediction using machine learning methods", *Journal of Software: Evolution and Process,* vol. 31, no. 10, pp. 1–25, 2019, doi:

10.1002/smr.2211.

[14] M. Arora, S. Verma, Kavita, and S. Chopra, "A Systematic Literature Review of Machine Learning Estimation Approaches in Scrum Projects", *Advances in Intelligent Systems and Computing,* vol. 1040, no. January, pp. 573–586, 2020, doi: 10.1007/978-981-15-1451-7_59.

[15] Y. Alqasrawi, M. Azzeh, and Y. Elsheikh, "Locally weighted regression with different kernel smoothers for software effort estimation", *Science of Computer Programming,* vol. 214, p. 102744, 2022, doi: 10.1016/j.scico.2021.102744.

[16] S. Hameed, Y. Elsheikh, and M. Azzeh, "An optimized case-based software project effort estimation using genetic algorithm", *Information and Software Technology,* vol. 153, no. October 2022, p. 107088, 2023, doi: 10.1016/j.infsof.2022.107088.

[17] E. Hariyanti *et al*., "Implementations of Artificial Intelligence in Various Domains of IT Governance: A Systematic Literature Review", *Journal of Information Systems Engineering and Business Intelligence,* vol. 9, no. 2, pp. 305–319, 2023.

[18] A. Ali and C. Gravino, "Improving software effort estimation using bio-inspired algorithms to select relevant features: An empirical study", *Science of Computer Programming,* vol. 205, p. 102621, 2021, doi: 10.1016/j.scico.2021.102621.

[19] T. R. Benala and R. Mall, "DABE: Differential evolution in analogy-based software development effort estimation", *Swarm and Evolutionary Computation,* vol. 38, no. May 2016, pp. 158–172, 2018, doi: 10.1016/j.swevo.2017.07.009.

[20] D. Rankovic, N. Rankovic, M. Ivanovic, and L. Lazic, "Convergence rate of Artificial Neural Networks for estimation in software development projects", *Information and Software Technology,* vol. 138, p. 106627, 2021, doi: 10.1016/j.infsof.2021.106627.

[21] C. Shekhar Yadav, R. Singh, S. Satpathy, S. Baghavathi Priya, B. T. Geetha, and V. Goyal, "Energy efficient and optimized genetic algorithm for software effort estimator using double hidden layer bi-directional associative memory", *Sustainable Energy Technologies and Assessments,* vol. 56, no. January, p. 102986, 2023, doi: 10.1016/j.seta.2022.102986.

[22] V. Nguyen, B. Boehm, and L. G. Huang, "Determining relevant training data for effort estimation using Window-based COCOMO calibration", *Journal of Systems and Software,* vol. 147, pp. 124–146, 2019, doi: 10.1016/j.jss.2018.10.019.

[23] T. Xia, R. Shu, X. Shen, and T. Menzies, "Sequential Model Optimization for Software Effort Estimation", *IEEE Transactions on Software Engineering,* vol. 48, no. 6, pp. 1994–2009, 2022, doi: 10.1109/TSE.2020.3047072.

[24] B. Marapelli, "Software Development Effort Duration and Cost Estimation using Linear Regression and K-Nearest Neighbors Machine Learning Algorithms", *International Journal of Innovative Technology and Exploring Engineering (IJITEE),* vol. 9, no. 2, pp. 1043–1047, 2019, doi: 10.35940/ijitee.k2306.129219.

[25] K. H. Kumar and K. Srinivas, "An accurate analogy based software effort estimation using hybrid optimization and machine learning techniques", *Multimedia Tools and Applications,* vol. 82, no. 20, pp. 30463–30490, 2023, doi: 10.1007/s11042-023-14522-x.

[26] A. Idri, I. Abnane, and A. Abran, "Support vector regression-based imputation in analogy-based software development effort estimation", *Journal of Software: Evolution and Process,* vol. 30, no. 12, pp. 1–23, 2018, doi: 10.1002/smr.2114.

[27] A. G. Priya Varshini, K. Anitha Kumari, and V. Varadarajan, "Estimating software development efforts using a random forest-based stacked ensemble approach", *Electronics,* vol. 10, no. 10, pp. 1–21, 2021, doi: 10.3390/electronics10101195.

[28] H. D. P. De Carvalho, R. Fagundes, and W. Santos, "Extreme Learning Machine Applied to Software Development Effort Estimation", *IEEE Access,* vol. 9, pp. 92676–92687, 2021, doi: 10.1109/ACCESS.2021.3091313.

[29] R. Marco, S. S. S. Ahmad, and S. Ahmad, "Bayesian Hyperparameter Optimization and Ensemble Learning for Machine Learning Models on Software Effort Estimation", *International Journal of Advanced Science and Computer Applications (IJASCA),* vol. 13, no. 3, pp. 419–429, 2022, doi: 10.14569/IJACSA.2022.0130351.

[30] M. A. Shah, D. N. A. Jawawi, M. A. Isa, M. Younas, A. Abdelmaboud, and F. Sholichin, "Ensembling Artificial Bee Colony with Analogy-Based Estimation to Improve Software Development Effort Prediction", *IEEE Access,* vol. 8, pp. 58402–58415, 2020, doi: 10.1109/ACCESS.2020.2980236.

[31] M. S. Khan, F. Jabeen, S. Ghouzali, Z. Rehman, S. Naz, and W. Abdul, "Metaheuristic Algorithms in Optimizing Deep Neural Network Model for Software Effort Estimation", *IEEE Access,* vol. 9, pp. 60309–60327, 2021, doi: 10.1109/ACCESS.2021.3072380.

[32] I. Abnane, A. Idri, I. Chlioui, and A. Abran, "Evaluating ensemble imputation in software

effort estimation", *Empirical Software Engineering,* vol. 28, no. 2, pp. 1–37, 2023, doi: 10.1007/s10664-022-10260-0.

[33] P. Phannachitta, "On an optimal analogy-based software effort estimation", *Information and Software Technology,* vol. 125, no. April, p. 106330, 2020, doi: 10.1016/j.infsof.2020.106330.

[34] A. Kaushik, N. Singal, and M. Prasad, "Incorporating whale optimization algorithm with deep belief network for software development effort estimation", *International Journal of Systems Assurance Engineering and Management,* vol. 13, no. 4, pp. 1637–1651, 2022, doi: 10.1007/s13198-021-01519-8.

[35] J. Rashid, S. Kanwal, M. W. Nisar, J. Kim, and A. Hussain, "An Artificial Neural Network-Based Model for Effective Software Development Effort Estimation", *Computer Systems Science and Engineering,* vol. 44, no. 2, pp. 1309–1324, 2023, doi: 10.32604/csse.2023.026018.

[36] S. K. Sehra, Y. S. Brar, N. Kaur, and S. S. Sehra, "Software effort estimation using FAHP and weighted kernel LSSVM machine," *Soft Computing - A Fusion of Foundations, Methodologies & Applications,* vol. 23, no. 21, pp. 10881–10900, 2019, doi: 10.1007/s00500-018-3639-2.

[37] H. Azath, M. Mohanapriya, and S. Rajalakshmi, "Software Effort Estimation Using Modified Fuzzy C Means Clustering and Hybrid ABC-MCS Optimization in Neural Network", *Journal of Intelligent Systems,* vol. 29, no. 1, pp. 251–263, 2020, doi: 10.1515/jisys-2017-0121.

[38] A. Latif, L. A. Fitriana, and M. R. Firdaus, "Comparative Analysis of Software Effort Estimation Using Data Mining Technique and Feature Selection", *JITK (Jurnal Ilmu Pengetahuan dan Teknologi Komputer),* vol. 6, no. 2, pp. 167–174, 2021, doi: 10.33480/jitk.v6i2.1968.

[39] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms", *Journal of Systems and Software,* vol. 137, pp. 184–196, 2018, doi: 10.1016/j.jss.2017.11.066.

[40] K. Dutta, V. Gupta, and V. S. Dave, "Analysis and Comparison of Neural Network Models for Software Development Effort Estimation", *Research Anthology on Agile Software, Software Development, and Testing,* vol. 1, pp. 165–193, 2021, doi: 10.4018/978-1-6684-3702-5.ch009.

[41] S. S. Ali, J. Ren, K. Zhang, J. Wu, and C. Liu, "Heterogeneous Ensemble Model to Optimize Software Effort Estimation Accuracy", *IEEE Access,* vol. 11, no. February, pp. 27759–27792, 2023, doi: 10.1109/ACCESS.2023.3256533.

[42] P. Phannachitta and K. Matsumoto, "Model-based software effort estimation - A robust comparison of 14 algorithms widely used in the data science community", *International Journal of Innovative Computing, Information and Control (IJICIC),* vol. 15, no. 2, pp. 569–589, 2019, doi: 10.24507/ijicic.15.02.569.

[43] S. Kumari and S. Pushkar, "Cuckoo search based hybrid models for improving the accuracy of software effort estimation", *Microsystem Technologies,* vol. 24, no. 12, pp. 4767–4774, 2018, doi: 10.1007/s00542-018-3871-9.

[44] M. M. Öztürk, "A tuned feed-forward deep neural network algorithm for effort estimation", *Journal of Experimental & Theoretical Artificial Intelligence,* vol. 34, no. 2, pp. 235–259, 2022, doi: 10.1080/0952813X.2021.1871664.

[45] N. Rankovic, D. Rankovic, M. Ivanovic, and L. Lazic, "A New Approach to Software Effort Estimation Using Different Artificial Neural Network Architectures and Taguchi Orthogonal Arrays", *IEEE Access,* vol. 9, pp. 26926–26936, 2021, doi: 10.1109/ACCESS.2021.3057807.

[46] 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, no. 5, pp. 2487–2496, 2022, doi: 10.1007/s41870-022-00962-5.

[47] N. Rankovic, D. Rankovic, M. Ivanovic, and L. Lazic, "Improved effort and cost estimation model using artificial neural networks and taguchi method with different activation functions", *Entropy,* vol. 23, no. 7, 2021, doi: 10.3390/e23070854.

[48] Q. Liu, J. Xiao, and H. Zhu, "Feature selection for software effort estimation with localized neighborhood mutual information", *Cluster Computing,* vol. 22, no. 1, pp. 6953–6961, 2019, doi: 10.1007/s10586-018-1884-x.

[49] W. Rhmann, B. Pandey, and G. A. Ansari, "Software effort estimation using ensemble of hybrid search-based algorithms based on metaheuristic algorithms", *Innovations in Systems and Software Engineering,* vol. 18, no. 2, pp. 309–319, 2022, doi: 10.1007/s11334-020-00377-0.

[50] P. Suresh Kumar, H. S. Behera, J. Nayak, and B. Naik, "A pragmatic ensemble learning approach for effective software effort estimation", *Innovations in Systems and Software Engineering,* vol. 18, no. 2, pp. 283–299, 2022, doi: 10.1007/s11334-020-00379-y.

[51] A. Jain and A. Bansa, "Effort Estimation using Neural Network and Metaheuristic Optimizer", *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO 2022),* pp. 1–5, 2022, doi: 10.1109/ICRITO56286.2022.9965014.

[52] V. Resmi and S. Vijayalakshmi, "Analogy-based approaches to improve software project effort estimation accuracy", *Journal of Intelligent Systems,* vol. 29, no. 1, pp. 1468–1479, 2020, doi: 10.1515/jisys-2019-0023.

[53] M. Hosni, A. Idri, and A. Abran, "Evaluating filter fuzzy analogy homogenous ensembles for software development effort estimation", *Journal of Software: Evolution and Process,* vol. 31, no. 2, pp. 1–26, 2019, doi: 10.1002/smr.2117.

[54] V. Resmi, S. Vijayalakshmi, and R. S. Chandrabose, "An effective software project effort estimation system using optimal firefly algorithm", *Cluster Computing,* vol. 22, no. s5, pp. 11329–11338, 2019, doi: 10.1007/s10586-017-1388-0.

[55] Z. Shahpar, V. K. Bardsiri, and A. K. Bardsiri, "Polynomial analogy-based software development effort estimation using combined particle swarm optimization and simulated annealing", *Concurrency and Computation: Practice and Experience,* vol. 33, no. 20, 2021, doi: 10.1002/cpe.6358.