A Review on Machine Learning Techniques for Software Effort Estimation

Rishika Pali
Department of Computer Science,
Shri Mathuradas Mohota College of Science,
Nagpur, Maharashtra,India 440024

Ms. Rishika Pali,
Assistant Professor, Department of Computer Science,
Shri Mathuradas Mohota College of Science,
Nagpur, Maharashtra, India 440024**,**

Abstract
Software effort estimation (SEE) is an essential element of software development, particularly in industrial software systems. It plays a key role in project planning, resource distribution, and cost control. Accurate prediction of the effort and cost associated with software development requires substantial expertise. Precise estimations enable efficient projects arescheduling deliveredand on timeworkforce and withinmanagement, budget. However, ensuring both overestimation and underestimation can lead to significant project risks. Machine learning algorithms have proven to be effective in uncovering patterns and correlations within extensive datasets, allowing for more accurate predictions that reflect the complexity of software projects. This review gives a consolidated idea about recent machine learning approaches used to enhance the efficacy of SEE. Using machine learning techniques, software effort estimation by leveraging empirical data.

Keywords: Software Effort Estimation, Machine Learning, Al- gorithm

I. Introduction

Software project management is based on estimation of software effort.[1] Estimation techniques are used in software project management to forecast time, effort, and cost, which are usually essential for the development process. It is still one of the trickiest parts of managing a software project. The intangible nature of computer software and the estimation's inherent wide margin of error make software estimation particularly challenging.[2]

A significant obstacle in the software sector is accurately estimating the development effort, which is essential for resource planning, risk assessment, and progress monitoring. Estimates that are not correct can have unfavorable effects; for instance, overestimating can waste resources, while underestimating might result in projects being approved that ultimately cost more than they were originally budgeted for.[3] In order to improve software effort estimation and overcome these obstacles, researchers and practitioners have resorted to Artificial Intelligence (AI) techniques, including Machine Learning. Large datasets have demonstrated remarkable patterns and relationships that machine learning algorithms may find, enabling more accurate forecasts that take into account the intrinsic complexity of software projects.[3] This paper aims to review some Machine Learning techniques which can be proven very effective for the prediction of Software Effort Estimation. The paper is structured as follows, the first section gives a detailed introduction, the second section is a brief detail on software effort estimation, the third section is a brief about traditional approach followed by the fourth section where the machine learning approach has been discussed. The fifth section is a year- wise review of various machine learning areas and its analysis is described in the sixth section. The brief conclusion of this paper is stated in the last section.

II. Software Effort Estimation in Detail *A.*

What is Software Effort Estimation?

The technique of estimating how many human re- sources—measured in person-hours or person-months—will be required to finish any software development project is known as effort estimation in software engineering. Because it assists stakeholders in planning and budgeting for the project as well as in making informed decisions about resource allocation and project boundaries, it is an essential component of project management. [4]

B. What is Effort Estimation in Software Development?

One step that is essential to the software development life cycle is effort estimation, which is crucial for figuring out how many hours are anticipated to be needed to do particular tasks in a software development project. To put it simply, effort estimation in software development is the process of calculating how many person-hours or person-days of labor are required to do a certain task or the project as a whole. This covers every stage of the Software Development Life Cycle (SDLC), including requirements collection, specification preparation, software design, coding, testing, and maintenance. An essential part of project management is effort estimation, which makes it possible to forecast how long a project will take to complete and guarantees that the quality and costs will meet client expectations. Expert-based and model-based methods are two popular and well-known categories of estimate techniques. While the latter are based on mathematical models, the former depends on human experience, judgment, and knowledge. COCOMO, Function Point Analysis, and Use Case Points are a few examples of these techniques. [5]

III. Traditional Methods for Software effort estimation:

There are several methods for estimating effort in software engineering, and each has pros and cons. Some of the more widely used methods are:

A. *Expert judgment:*

This method is based on the expertise and experience of people who have previously worked on related projects. It can provide a broad concept of the necessary effort and is commonly used as the first step in effort estimation. Nevertheless, it is susceptible to prejudices and could be impacted by an individual's specific experiences.

B. *Delphi Technique:*

Project estimates are anonymously provided by a group of specialists. A second round of estimations is then conducted after their results have been combined and distributed to the group. Until an agreement is found, this process keeps on.

C. *Function Point Analysis (FPA):* This approach focuses on assessing the software's features and functions (such as inputs, outputs, files, and user interactions) in order to determine its size and complexity. After that, function points are mapped to effort using past performance or industry norms.

D. *Use Case Points (UCP):* With this approach, the software system's use cases are used to estimate effort. The overall effort is determined by taking into account the weights (complexity) allocated to each use case as well as additional elements such as technical complexity.

E. *COCOMO (Constructive Cost Model):*

The Constructive Cost Model, or Cocomo, is a parametric model that determines effort depending on the project's size and complexity.

F. *LOC (Lines of Code) Estimation:*

The number of lines of code that must be written is used to evaluate effort in this method. Using a productivity factor (such as hours per line of code), the effort is computed.

G. *Analogous estimation:*

This method forecasts the amount of work required for the present project by using data from earlier, comparable

initiatives. It is a quick and easy procedure, but because the projects might not be precisely the same, its accuracy may be limited.

H. *Three-point estimation:*

This method generates a range of possible values by combining the most likely, optimistic, and pessimistic estimations of effort. It is a quite complicated procedure, but it aids in producing a range of likely effort estimates.

I. *Parametric estimation:*

This technique calculates the amount of work needed for a project using mathematical models. The project's size, complexity, and other factors determine it.[4]

IV. Machine Learning Approach for Software Effort Estimation

One of the most important and difficult project management tasks is estimating or forecasting the software development effort. Software project planning and management are impossi- ble without precise estimations. Effort prediction models have shown promise in predicting the amount of development effort needed for a software assignment, in contrast to the industry's inaccurate forecasts. A software project's time/duration and cost are crucial elements that are impacted by things like the rapid evolution of software application technology and shifting customer needs. Furthermore, software projects are conceptual in nature, which means that effort cannot be estimated before the project's work really starts, in contrast to other project types. Experts have been trying to increase the precision of development effort estimates for decades in order to efficiently plan and manage software projects. From crude assumptions to more advanced techniques, the estimation process has changed over time. Project managers find software effort estimating (SEE) to be particularly difficult because of the software industry's dynamic environment. For the project to be completed successfully, preliminary planning is essential. It entails an accurate estimation of the resources needed to finish the project on schedule. The more accurately resources (or efforts) are estimated, the more likely it is that a project will be finished within the given restrictions. Many SEE models based on Machine Learning (ML) have been developed in the past 30 years.[2]

The importance of machine learning techniques in research is becoming more widely acknowledged. The results of machine learning techniques are reliably reliable and are frequently used with confidence in a variety of investigations. These methods can automatically adapt to new project characteristics after learning from past project data. Here are a few popular machine learning methods for estimating software work.[2]

A. *Linear Regression*

One of the most basic machine learning methods for estimating software effort is linear regression. It makes the assumption that the input characteristics—like project size, complexity, and team experience—and the output—effort needed—have a linear connection.

B. *Decision Trees*

Decision trees generate a tree structure by dividing the data into smaller groups according to feature values. The expected effort is represented by the leaves, while each branch indicates a choice depending on a feature.

C. *Random Forests*

Several decision trees are combined in the ensemble learning technique known as random forests. A random subset of the data is used to train each tree, and the average of all the trees' predictions yields the final estimate.

D. *Support Vector Machines (SVM)*

Support Supervised learning methods called vector machines look for the optimum hyperplane in feature space to divide the input points. SVMs can also be applied to regression (SVR), a process in which the objective is to use input data to predict a continuous value (effort).

E. *K-Nearest Neighbors (K-NN)*

K-NN is a non-parametric technique that uses the average of the K closest points in the feature space to forecast a new data point. It is applied to jobs involving both regression and classification.

F. *Artificial Neural Networks (ANN)*

The layers of nodes (neurons) in artificial neural networks (ANNs) are connected by weights and are modelled after biological neural networks. In particular, they are helpful for identifying intricate, non-linear patterns in data. As a subset of ANN, deep learning models complex patterns by employing several hidden layers.

G. *Gradient Boosting Machines (GBM)*

GBM is an ensemble approach that develops models in a sequential fashion, with each new model fixing the mistakes of the one before it. LightGBM, CatBoost, and XGBoost are popular implementations.

H. *Bayesian Networks*

Bayesian networks depict the relationships between variables using probabilistic graphical models. These models can be used to depict the dependencies and uncertainty among different project aspects in software work estimating.

I. *Clustering Algorithms*

Clustering techniques use feature similarity to group data into clusters. Following the formation of the clusters, effort estimates can be calculated using each cluster's unique properties.

J. *Genetic Algorithms*

Natural selection serves as the inspiration for optimization strategies known as genetic algorithms. By improving solutions over time, they can be used to directly optimize effort estimation models or to adjust the parameters of other machine learning models.

K. *Ensemble Learning (Stacking, Boosting, Bagging)*

Several machine learning models are combined in ensemble learning to enhance performance. Ensemble approaches seek to lessen bias and variation by utilizing the advantages of several models (such as decision trees, linear regression, and neural networks).

L. *Reinforcement Learning*

Learning from the results of actions is known as reinforcement learning. In project management, it could be used to dynamically modify effort estimation in response to ongoing project results or progress.

V. Review on Machine Learning Techniques for Software Effort Estimation

Many researchers/ authors had taken lot of efforts to find out better solution for predicting effort estimation of software using machine learning techniques Table 1 illustrates total 25 papers, which includes Research focus of author and what they had concluded.

VI. Analysis on review on Machine Learning Techniques

From literature review reported in T**able 1,** we inferred that by identifying intricate patterns and relationships in the data, machine learning models can frequently perform better than conventional estimating methods. Particularly for big projects or datasets, machine learning (ML) can minimize the manual labor involved in project work estimation. Retraining model on fresh data can gradually increase their precision and flexibility. As more data becomes available, more accurate predictions can be made because to ML algorithms' ability to handle vast amounts of data.

Table 1: Review on Machine Learning Techniques for SEE

S. N .	**Publication Details**	**Research Focus**	**Research Outcome**
1	Informatica 2024 (ISSN: 1822-8844) (DOI:https://doi. org/10.31449/inf. v48i3.4515	By adding risk exposure to a project's initial work estimate, the author suggests a way to increase the accuracy of software effort prediction. This study uses evolutionary algorithms, namely Artificial Bee Colony (ABC), Particle Swarm Optimization (PSO), and Global Learning-Based PSO (GLBPSO), to optimize the estimation process and offer a mechanism to combine risk exposure with initial estimations.	The author is presenting a novel approach to improving software effort estimation by integrating the impact of risk exposure into the estimation process. The approach assigns weights to different project cost factors based on their contribution to risk exposure. These weights are then optimized using evolutionary algorithms, specifically ABC, PSO, and GLBPSO. [1]
2	IEEE 2024 (ISSN:26441268) (DOI: 10.1109/ACCES S.2024.3404879)	The author highlights the importance of accurate software effort estimation (SEE) for successful project planning, resource allocation, and on-time delivery. Since overestimation and underestimation can create significant challenges, machine learning (ML) approaches are increasingly being used to improve estimation accuracy.	This paper makes a substantial contribution to the subject of software effort estimation by: examining and evaluating the ML methods applied in SEE. [6]
3	Scalable Computing Practice and Experience 2024 (ISSN: 1895-1767) (DOI 10.12694/scpe.v2 5i2.2213)	The author talks about how effort estimation is crucial to software development and how it aids in project managers' scheduling, planning, and management of software projects. Comparing different machine learning approaches for software effort estimation—especially the most popular and latest approaches—is the main goal of the study.	In his discussion on the value of effort estimation in software development, the author highlights how it affects scheduling, staffing, and cost. According to the report, one of the main reasons software projects fail is inaccurate early-stage project estimation.[4]

4	applied science 2023 (ISSN:2076-3417) (DOI: https://doi.org/10.3390/app13169465)	The author presents a Bayesian Network model for task effort prediction and talks about the difficulty of effort estimation in agile software development. By verifying Bayesian Networks on actual agile projects, the study seeks to increase their usefulness for software project managers.	In this research, a BN model for agile software development effort prediction is developed. The suggested model is straightforward and quite modest, and it is easy to extract all of the input data. [7]
5	IEEE 2023 (ISSN:26441268) (DOI: 10.1109/ACCESS.2023.3293432)	The author emphasizes the value of software effort estimation in digital transformation projects and industrial software systems, stressing how it enhances processes, operations, customer experiences, and overall performance. The study presents an effort estimating method based on machine learning to improve prediction accuracy.	The author emphasizes the importance of software effort estimation in industrial software systems and digital transformation initiatives, highlighting how accurate estimation is essential for planning, resource allocation, and project success. [8]
6	International Journal of Advanced Computer Science and Applications 2023 (ISSN: 2158 -107X)	No single algorithm produces improved predictions across all datasets, claims the no free lunch principle. author's work attempts to eliminate this bias by improving the software effort prediction model and minimizing the discrepancy between the anticipated and actual effort for upcoming projects.	The author talks about how software effort estimating techniques have changed over time, emphasizing the move away from traditional approaches like function points, lines of code, CPM, PERT, and expert judgment and toward more sophisticated machine learning techniques.[9]
7	IEEE 2023 (ISSN:26441268) (DOI:10.1109/ICCD59681.2023.10420603)	The study seeks to increase accuracy and dependability using conventional approaches. The most efficient approach is determined by comparing and evaluating performance with several machine learning models, such as Random Forest, Linear Regression, Support Vector Machines, Artificial Neural Networks (ANN), and others. The suggested AI-based framework could improve resource allocation and project planning, advancing the field of software project effort estimation research.	The results and technique offered in this study can be a useful reference for next research and realworld applications in the software engineering and project management areas as machine learning continues to progress.[3]
8	International Information and Engineering Technology Association 2023 (ISSN:20419031) (https://doi.org/10.18280/isi.280602)	Author has done analysis of machine learning technique for improving software effort estimation. Machine learning algorithm such as evaluated-linear regression, Gradient Boosting, Random Forest, and Decision Tree have been evaluated.	The study lays a solid basis for intelligent estimating tools and recommends more research into sophisticated regression methods and the validation of industrial datasets for improvement.[2]

9	IEEE 2023 (ISNN:26441268) (DOI: 10.1109/ACCES S.2023.3293432)	The author discuss that task-level estimation in software projects can benefit from the use of ensemble machine learning techniques, with aggregated estimates demonstrating good accuracy even in cases where individual task forecasts are less accurate.	The author comes to the conclusion that software projects can effectively estimate task-level effort and time through the use of ensemble machine learning methods.[10]
10	International Journal of Computer Applications 2022 (ISSN 0975 – 8887)	In agile software development, the author mainly investigates different machine learning methods for software effort estimation. GRNN, Probabilistic Neural Network (PNN), GMDHPNN, and Cascade-Correlation Neural Network (CCNN) are among the neural network types that are compared.	After the study, recommendations have been made to develop an ensemble-based deep learning model. Additionally, NLP can be utilized to extract keywords from use case stories. [11]
11	International Journal of Computing and Digital Systems 2022 ISSN (2210-142X) (https://dx.doi.org/10.12785/ijcds/120112)	According to the author, machine learning methods—in particular, neural networks and evolutionary algorithms—offer incredibly precise forecasts for estimating software labor. While projects based on lines of code also produced encouraging results, the findings were more accurate when story points were utilized for estimation.	In order to improve the accuracy of predictions for other development processes, the author proposes merging several machine learning algorithms with feature selection techniques. This suggests that future work will be concentrated on further refining estimation models.[12]
12	IEEE 2021 (ISSN : 2644-1268) (DOI 10.1109/ACCES S.2021.3072380)	The author evaluates the GWO and SB algorithms' performance against that of five other meta-heuristic methods that are frequently employed in software effort estimation literature.he suggested DNN model (called GWDNNSB) outperforms earlier methods that employed DNN for software effort estimation by integrating GWO and SB for weight and learning rate optimization.	The author believes that meta-heuristic algorithms like GWO and SB can considerably improve the performance of DNN models for software effort estimation by optimizing important parameters, speeding up training, and increasing accuracy.[13]
13	IEEE 2020 (ISSN : 2644-1268) (DOI:10.1109/.2020.3021664)	The author is examining how effort estimating models in Agile Software Development (ASD) have changed. They found that expert-based estimating techniques like Planning Poker are still frequently utilized in six agile approaches. In line with agile principles, the survey also reveals that team and project aspects are prioritized over technical ones.	The impact of cost considerations on estimating accuracy will be investigated in future studies, along with incremental updates and the possibility of developing an estimation model using historical data from the ISBSG repository, which provides cross-company insights for wider applicability. [14]

14	IEEE 2020 (ISSN:2644-1268) (DOI:10.1109/2024.3457771)	The author main objectives are to identify various estimating models, benchmark datasets, accuracy measurements, and frequently utilized models. This indicates the growing demand for ML-based models. Using techniques like Random Forest (RF), Support Vector Machine (SVM), AdaBoost (AB), and Gradient Boost (GB), the study investigates feature selection and relevance in ML models for SEE.	The work highlights the significance of feature selection in SEE and seeks to improve software effort estimation through the identification of important affecting factors. In the end, the research helps to enhance SEE procedures and identifies prospective topics for additional study. [15]
15	International Journal of Advanced Computer Science and Applications 2020 (ISSN: 2158-107X)	The author talks about how decision trees (DT) are receiving more and more attention in software development effort estimation (SDEE) because of their predictability, usability, and clarity.	Instead of using the potentially biased MMRE, the author recommends that future research include a comparative analysis using objective evaluation criteria like standard accuracy (SA) and effect size. This would make it easier to comprehend the advantages and disadvantages of DT methods in SDEE.[16]
16	IEEE 2019 (ISSN:2644-1268) (DOI: 10.1109/ICSESEIP.2019.00042)	According to the author, companies have had to change their development procedures as a result of the growing demand in incorporating AI capabilities into software and services brought about by recent developments in machine learning. The author details a study they carried out while watching Microsoft software teams develop AI-based products.	In this study, the author pinpoints important process modifications and best practices implemented by Microsoft teams in recent years to tackle the difficulties associated with developing and deploying large-scale ML-based applications. [17]
17	Journal of Software : Evolution and process 2019(ISSN :2047-7473) (https://doi.org/10.1002/smr.2211)	The paper's author systematically reviews recent research on machine learning (ML)-based software effort estimating methods. The two machine learning approaches that are most frequently used in the studies are artificial neural networks (ANN) and support vector machines (SVM), with ANN being the most common. The paper highlights the value of both machine learning (ML) and non-ML approaches in software effort estimation, emphasizing that the best prediction accuracy can be achieved by combining several approaches.	An further SLR on bio-inspired feature selection techniques for software effort estimation is planned by the author.[18]
18	ECAI 2019 (ISSN:12915912) (DOI :10.1109/ECAI46879.2019.9042031)	The significance of effort estimation, which entails projecting the amount of time and resources needed to finish a software project, is covered by the author. This estimate is essential for figuring out the project's cost and drawing in customers.The author proposes machine learning techniques as a potential answer to the problem of accurate estimation.	In order to estimate software effort, the author highlights the importance of meticulous data preprocessing and the best feature selection in machine learning models. The study contrasts several methods and verifies the accuracy of the findings by comparing the data to original datasets.[19]

19	IEEE 2019 (ISSN : 2644-1268) (DOI: 10.1109/3ICT.2018.8855752)	According to the author, machine learning algorithms like ANN, SVM, K-star, and linear regression are dependable methods for estimating software development effort since they can be useful instruments for forecasting software effort based on system characteristics.	The SVM model performs the best when it comes to software effort estimation, according to the author, who also notes that future research will concentrate on testing additional algorithms and enhancing accuracy through feature filtering and dataset analysis.[20]
20	Journal of Software : Evolution and process 2019(ISSN :2047-7473) (DOI: 10.1002/smr.2165)	The author details a study aimed at enhancing agile software development's ability to estimate work.The results demonstrate that the hybrid approach, which blends objective analysis, expert judgment, and the GBT model, produces estimates that are more accurate than those that only use expert judgment or model-based estimates.	According to this, the hybrid technique is successful and well-liked, offering a viable way to enhance effort estimation in agile software development.[21]
21	IAES International Journal of Artificial Intelligence (IJAI) 2019(ISSN: 2252-8938) (DOI:10.11591/ijai.v8.i4.pp399410)	The significance of precise software development effort estimation (SDEE) for project management is covered by the author. The paper emphasizes how artificial intelligence (AI) methods, including support vector regression (SVR), are increasingly being used to anticipate software effort. Using SVR in conjunction with two feature selection (FS) techniques—Random Forest and Boruta—the author suggests an enhanced estimation model.	The author comes to the conclusion that feature selection increases the accuracy of SVR models for software effort estimation, and that the Boruta technique is outperformed by backward feature deletion based on RF feature importance.[22]
22	IEEE 2018 (ISSN : 2644-1268) (DOI: 10.1109/CSIT.2018.8486222)	According to the author, conventional software estimation techniques including expert-based, analogy-based, and algorithmic approaches (like COCOMO) frequently have substantial error margins. Software projects may have delays and go over budget as a result of this. The author suggests a different strategy that uses data mining techniques on past project data to increase estimation accuracy.	Author Conclude that Random forest is best as compare to other two machine learning technique for classification accuracy and precision performance score, while naïve bayes technique perform better in ROC curve and Recall, Whereas Random forests in its AUC score. future work is recommended for investigating machine learning technique for improving drivers of COCOMO models and improve value of multiplier[5]
23	IEEE 2018 (ISNN:2644-1268) (DOI 10.1109/AiCIS.2018.00016)	According to the author, genetic algorithms produce results that are competitive with neural networks, particularly the NARX network, which provides the most accurate software effort estimation. In contrast, it was discovered that fuzzy logic was less successful.	The author comes to the conclusion that the most accurate software effort estimation is provided by neural networks, particularly the NARX network, followed by genetic algorithms (with MEP being superior to GEP)[23]

25	IEEE 2016 (ISSN:26441268) (DOI: 10.1109/ACCES S.2017)	The author covers the relevance of software cost and effort estimation in software engineering, highlighting its role in project success by helping identify obstacles and risks early in development. The article presents a systematic literature review (SLR) to examine research trends in this topic over the past 15 years and proposes a path ahead for enhancing estimating methodologies.	This study provides valuable insights for researchers and practitioners in software project planning, identifying effective estimation techniques, and highlighting gaps in emerging technologies for future exploration. [25]
24	Journal of Web Engineering (2018) (ISSN :1526-2253)	To increase the precision of software development cost and effort prediction, the author suggests utilizing the Meta-heuristic Harmony Search Algorithm in conjunction with Neural Networks (GRNN and RBFNN). Using real-world datasets from a variety of industries, the work focuses on lowering risks in the estimating process by utilizing sophisticated machine learning techniques and efficient algorithms.	The author comes to the conclusion that RBFNN outperforms GRNN in software development cost and effort estimation when optimized utilizing meta-heuristic techniques like harmonic search.[24]

Conclusion

In software engineering, estimating effort has always been a difficult issue that relies on conventional techniques like expert [4]
opinion, historical data, and parametric models. However, new chances to improve the precision and dependability of software work estimation have surfaced as a result of the quick development of machine learning (ML). With greater accuracy [5]
and flexibility than conventional approaches, machine learning techniques have demonstrated significant promise in enhancing software work estimation. Decision trees, random forests, support vector machines, and artificial neural networks are some of the [6]
approaches to machine learning that are most frequently used. This paper review the brief discussion on Machine Learning Techniques for software effort estimation. The literature review provide a range of viewpoints on machine learning methods used in software effort estimation (SEE), setting the stage for further [7]
advancements. This paper provides a comprehensive overview of the ways in which software effort estimation is being transformed by machine learning techniques. More accurate, flexible, and effective methods for overseeing software development projects [8]
are anticipated as a result of the field's continuous progress.

References:

[1] P. Singal, P. Sharma, and A. C. Kumari, “Integrated [9]
oftware Effort Estimation: a Hybrid Approach,” *Inform.*, vol. 48, no. 3, pp. 329–344, 2024, doi: 10.31449/inf.v48i3.4515.

[2] Meharunnisa, M. Saqlain, M. Abid, M. Awais, and Ž. Stević, “Analysis of Software Effort Estimation by [10]
achine Learning Techniques,” *Ing. des Syst. d’Information*, vol. 28, no. 6, pp. 1445–1457, 2023, doi: 10.18280/isi.280602.

[3] T. N. Tran, H. T. Tran, and Q. N. Nguyen, “Leveraging [11]
I for Enhanced Software Effort Estimation: A Comprehensive Study and Framework Proposal,” *2023 Int. Conf. Cogn. Comput. Complex Data, ICCD 2023*, pp. 284– 289, 2023, doi: 10.1109/ICCD59681.2023.10420603.

K. Lavingia, R. Patel, V. Patel, and A. Lavingia, “Software Effort Estimation Using Machine Learning Algorithms,” *Scalable Comput.*, vol. 25, no. 2, pp. 1276–1285, 2024, doi: 10.12694/scpe.v25i2.2213.

A. Banimustafa, “Predicting Software Effort Estimation Using Machine Learning Techniques,” *2018 8th Int. Conf. Comput. Sci. Inf. Technol. CSIT 2018*, no. July, pp. 249– 256, 2018, doi: 10.1109/CSIT.2018.8486222.

M. Rahman, H. Sarwar, A. Kader, T. Goncalves, and T. T. Tin, “Review and Empirical Analysis of Machine LearningBased Software Effort Estimation,” *IEEE Access*, vol. 12, no. March, pp. 85661–85680, 2024, doi:

10.1109/ACCESS.2024.3404879.

M. Turic, S. Celar, S. Dragicevic, and L. Vickovic, "Advanced Bayesian Network for Task Effort Estimation in Agile Software Development," *Appl. Sci.*, vol. 13, no. 16, 2023, doi: 10.3390/app13169465.

A. Jadhav, S. K. Shandilya, I. Izonin, and M. Gregus, "Effective Software Effort Estimation Leveraging Machine Learning for Digital Transformation," *IEEE Access*, vol. 11, no. January, pp. 83523–83536, 2023, doi: 10.1109/ACCESS.2023.3293432.

B. K. Kumar, S. Bilgaiyan, and B. S. P. Mishra, "Software Effort Estimation through Ensembling of Base Models in Machine Learning using a Voting Estimator," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 2, pp. 172–181, 2023, doi: 10.14569/IJACSA.2023.0140222.

A. O. Sousa *et al.*, "Applying Machine Learning to Estimate the Effort and Duration of Individual Tasks in Software Projects," *IEEE Access*, vol. 11, no. August, pp. 89933– 89946, 2023, doi: 10.1109/ACCESS.2023.3307310.

S. Kumar, M. Arora, Sakshi, and S. Chopra, "A Review of Effort Estimation in Agile Software Development using Machine Learning Techniques," *4th Int. Conf. Inven. Res. Comput. Appl. ICIRCA 2022 - Proc.*, vol. 184, no. 21, pp. 416–422, 2022, doi: 10.1109/ICIRCA54612.2022.9985542.

[12] A. Sharma and N. Chaudhary, "Analysis of Software Effort Estimation Based on Story Point and Lines of Code using Machine Learning," *Int. J. Comput. Digit. Syst.*, vol. 12, no. 1, pp. 131–140, 2022, doi: 10.12785/ijcds/120112.

[13] 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.

[14] M. Fernández-Diego, E. R. Méndez, F. González-Ladrón-De-Guevara, S. Abrahão, and E. Insfran, "An update on effort estimation in agile software development: A systematic literature review," *IEEE Access*, vol. 8, no. September, pp. 166768–166800, 2020, doi: 10.1109/ACCESS.2020.3021664.

[15] P. V. Terlapu, K. Kishore Raju, G. Kiran Kumar, G. Jagadeeswara Rao, K. Kavitha, and S. Samreen, "Improved Software Effort Estimation through Machine Learning: Challenges, Applications, and Feature Importance Analysis," *IEEE Access*, no. October, pp. 138663–138701, 2024, doi: 10.1109/ACCESS.2024.3457771.

[16] A. Najm, A. Marzak, and A. Zakrani, "Systematic review study of decision trees based software development effort estimation," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 7, pp. 542–552, 2020, doi: 10.14569/IJACSA.2020.0110767.

[17] A. Begel and T. Zimmermann, "Software Engineering for Machine Learning_ A Case Study _ Enhanced Reader.pdf".

[18] A. Ali and C. Gravino, "A systematic literature review of software effort prediction using machine learning methods," *J. Softw. Evol. Process*, vol. 31, no. 10, pp. 1– 25, 2019, doi: 10.1002/smr.2211.

[19] Z. Polkowski, J. Vora, S. Tanwar, S. Tyagi, P. K. Singh, and Y. Singh, "Machine Learning-based Software Effort Estimation: An Analysis," *Proc. 11th Int. Conf. Electron. Comput. Artif. Intell. ECAI 2019*, pp. 1–6, 2019, doi: 10.1109/ECAI46879.2019.9042031.

[20] M. Hammad and A. Alqaddoumi, "Features-level software effort estimation using machine learning algorithms," *2018 Int. Conf. Innov. Intell. Informatics, Comput. Technol. 3ICT 2018*, pp. 1–3, 2018, doi: 10.1109/3ICT.2018.8855752.

[21] B. Tanveer, A. M. Vollmer, S. Braun, and N. bin Ali, "An evaluation of effort estimation supported by change impact analysis in agile software development," *J. Softw. Evol. Process*, vol. 31, no. 5, pp. 1–17, 2019, doi: 10.1002/smr.2165.

[22] A. Zakrani, M. Hain, and A. Idri, "Improving software development effort estimation using support vector regression and feature selection," *IAES Int. J. Artif. Intell.*, vol. 8, no. 4, pp. 399–410, 2019, doi: 10.11591/ijai.v8.i4.pp399-410.

[23] F. A. Abulalqader and A. W. Ali, "Comparing different estimation methods for software effort," *Proc. - 2018 1st Annu. Int. Conf. Inf. Sci. AiCIS 2018*, pp. 13–22, 2018, doi: 10.1109/AiCIS.2018.00016.

[24] H. Azath, P. Amudhavalli, S. Rajalakshmi, and M. Marikannan, "A novel regression neural network based optimized algorithm for software development cost and effort estimation," *J. Web Eng.*, vol. 17, no. 6, pp. 3095–3125, 2018.

[25] C. H. Rashid *et al.*, "Software Cost and Effort Estimation: Current Approaches and Future Trends," *IEEE Access*, vol. 11, pp. 99268–99288, 2023, doi: 10.1109/ACCESS.2023.3312716.