International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 7s, 2025
https://www.theaspd.com/ijes.php

Software Cost Estimation: A Comparative Analysis Of Traditional And Machine Learning Approaches

Prof. Sachin Baburao Wakurdekar[1], Prof. Dr. S.B Vanjale[2], Dr. Pallavi Deshpande[1,] Dr. Tanuja Dhope[1], Prof. V.J. Shinde[1,] Dr. Datta S. Chavan[1], Prof. Dr. A.Y Prabhakar[1], Dr. Anand Shinde[3]

[1]*Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, 411043, Maharashtra, India.*
[2]*Corresponding Author, Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, 411043, Maharashtra, India.*
[3]*Bharati Vidyapeeth Institute of Environment Education and Research, Pune, 411043, Maharashtra, India.*

Abstract: *Estimating software costs is essential to project management because it helps businesses allocate resources efficiently. Despite their widespread use, traditional models like COCOMO frequently have drawbacks because they rely on assumptions and predefined parameters that are not well suited to contemporary software development techniques. Machine learning-based models, on the other hand, provide a data-driven strategy by utilizing past project data to increase estimation accuracy. This study compares and contrasts contemporary machine learning methods with conventional cost estimation models. We go over the drawbacks of traditional methods, investigate sophisticated regression strategies, and suggest a hybrid model that combines neural networks and XGBoost for increased accuracy. The superiority of our method in reducing estimation errors is shown by empirical results. The findings show that machine learning-based cost estimation can significantly enhance software engineering decision-making and resource planning. Neural networks, machine learning, XGBoost, COCOMO model, software cost estimation, and predictive analytics are some examples of index terms.*

Keywords: *NLCs, Sitagliptin, diabetes mellitus, bioavailability, drug loading, drug release*

INTRODUCTION

Accurate cost estimation is crucial for software development projects in order to guarantee effective scheduling, budgeting, and resource allocation. Software effort and cost have traditionally been predicted using estimation models such as the Constructive Cost Model (COCOMO). Although these models offer an organized method, they typically fall short in capturing the intricacy and flexibility of contemporary software projects.

Predictive analytics has become a powerful substitute for software cost estimation as machine learning has progressed. Machine learning models can adjust and increase estimation accuracy over time by utilizing historical data. Predictions become more accurate when methods such as XGBoost and neural networks are combined to create a hybrid approach that combines the advantages of both algorithms. The industry's difficulties in estimating costs.

A. Industry Challenges in Cost Estimation

For the software industry, cost estimation poses several challenges, such as:
Rapid Technological Developments: Because of recent advancements, it is difficult to rely on static estimation models.Project Complexity: estimating effort becomes more difficult in large-scale projects due to the numerous dependencies involved.Resource Allocation: Incorrect estimates lead to either overallocation or underutilization of resources.Scope Creep: Frequent requirement changes affect initial estimates, resulting in budgetary and timeline deviations. Given these challenges, an adaptive, data-driven approach is necessary for improving estimation accuracy

B. Proposed Solution and Research Motivation

Our study presents a hybrid cost estimation model that combines machine learning and conventional estimation methods to address the problems mentioned above. The following are the main drivers of this strategy: Including Data-Driven Insights: By examining past project data, machine learning models are able to spot hidden relationships and patterns that conventional models overlook.Improving Prediction Accuracy: Our strategy makes use of deep learning and feature importance analysis by fusing neural networks with XGBoost, a tree-based ensemble method. Increasing Adaptability: As new project data becomes available, machine learning-based techniques can continuously learn and adapt, in contrast to static models. Bridging the Gap Between Theory and Practice: Our model aims to provide a practical tool that software development firms can integrate into their project management workflows

C. Comparison of Existing Cost Estimation Techniques

Table I summarizes the pros and cons of commonly used cost estimation techniques. Our hybrid model aims to capitalize on the strengths of traditional models while addressing their shortcomings using machine learning techniques. The following sections will discuss the dataset used, implementation details, experimental results, and future research directions To systematically address the challenges in software cost estimation and validate our proposed hybrid approach, this study investigates the following research questions:RQ-1: What are the most effective machine learning and non-machine learning techniques for software cost estimation, and how does their accuracy compare across different datasets. RQ-2: What key factors influence the accuracy of software cost estimation using ML techniques, and how can hyperparameter tuning and feature selection improve estimation performance? RQ-3: How does the integration of XGBoost and neural networks enhance cost estimation compared to standalone models? RQ-4: What are the most used datasets and benchmark studies for evaluating software cost estimation techniques? The rest of this paper is structured as follows: Section II presents a review of existing literature on cost estimation techniques. Section III discusses the proposed hybrid methodology combining XGBoost and neural networks. Section IV provides experimental results and performance analysis. Finally, Section V concludes the study and outlines future research directions.

Table no 1. Analysis of cost estimation techniques discussed in literature

Technique	Pros	Cons
Expert Judgment [29]	Quick estimation, utilizes domain knowledge	Subjective, inconsistent across different estimators
COCOMO Model [30]	Uses historical data, adaptable to similar projects	Requires high-quality historical data, limited to similar past projects
Function Point Analysis [31]	Well-established, parameterized approach	Relies on predefined assumptions, less effective for modern software projects
Machine Learning (ML) [31]	Data-driven, self-improving, high accuracy	Requires large datasets, computationally expensive
XGBoost [32]	High interpretability, efficient handling of missing values	Requires careful hyperparameter tuning, prone to overfitting with small datasets
Neural Networks [32]	Captures complex relationships, excels in deep learning applications	Computationally intensive, requires large training data and tuning
Hybrid Approach (XGBoost + Neural Networks) [32]	Combines traditional and ML strengths, improves accuracy	Needs model training and validation, requires sufficient data for both models

LITERATURE REVIEW

Software effort and cost estimation methods have evolved significantly over time. Broadly, these methods can be categorized into three groups: expert judgment-based approaches, algorithm-based approaches, and machine learning-driven computational intelligence models. Each approach has distinct advantages and limitations, as summarized in Table II. The types of software cost estimation are summarized in Fig1.

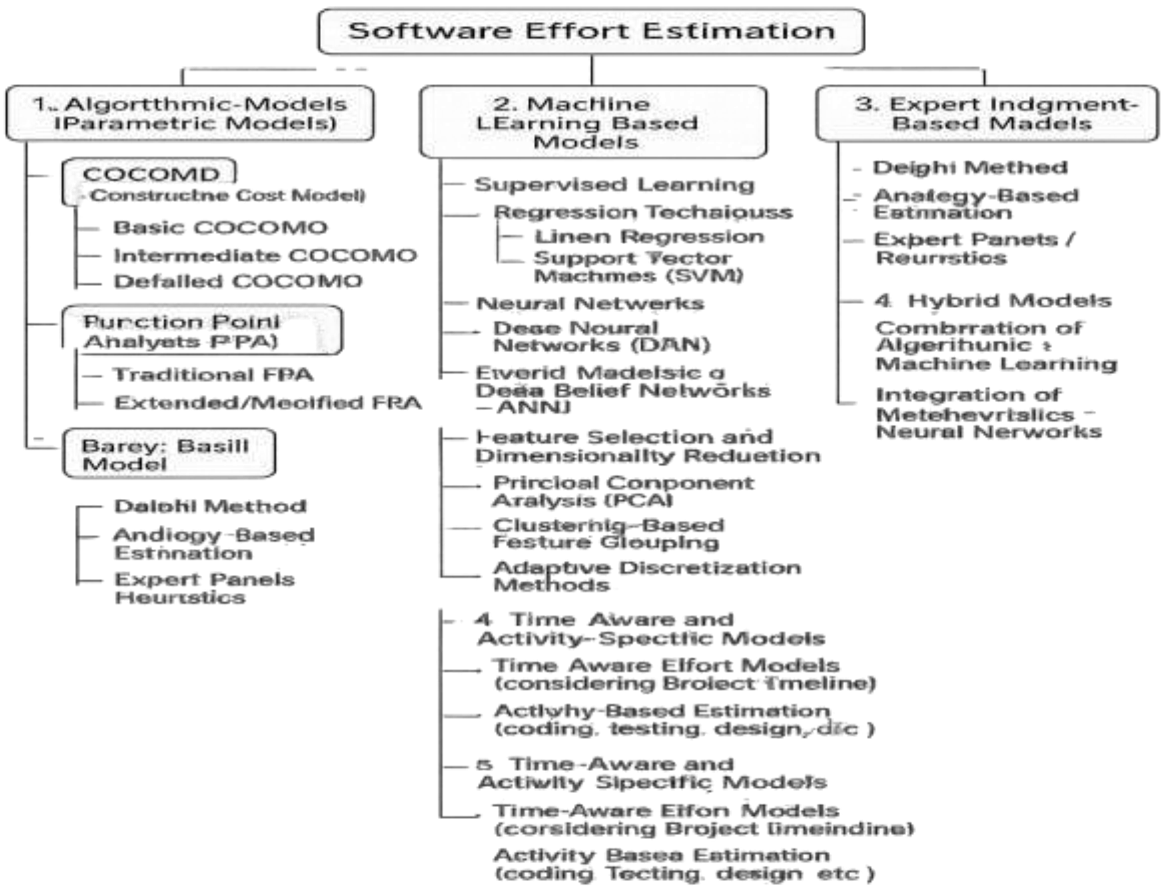
Software Effort Estimation
1.. Algortthmic-Models IParametric Models)
COCOMD
Constructne Cost Model)
Basic COCOMO
Intermediate COCOMO
Defalled COCOMO
Purction Point Analyets PPA)
Traditional FPA
Extended/Meoified FRA
Barey: Basill Model
Dalohi Method
Andiogy-Based Estnnation
Expert Panels Heurtstics
2. Machine LEarning Based Models
Supervised Learning
Regression Techaiouss
Linen Regression
Support Vector Machmes (SVM)
Neural Netwerks
Dese Noural Networks (DAN)
Etverid Madelsic a Deea Belief Networks –ANNJ
Feature Selection and Dimensionality Reduetion
Prircloal Conponent Aralysis (PCA)
Clusterhig-Based Festure Glouping
Adaptive Discretization Methods
4 Time Aware and Activity-Spectfic Models
Time Aware Elfort Models (considering Broiect Tmeline)
Actlyhy-Based Estimation (coding, testing, design, etc)
5 Time-Aware and Actiwity Sipectfic Models
Time-Aware Eifon Models (corsidering Broject limeindine)
Activity Basea Estimation (coding Tecting, design etc)
3. Expert Indgment-Based Madels
Deighi Methed
Analegy-Based Estnnation
Expert Panets / Reurrstics
4 Hybrid Models
Combrration of Algerihunic + Machine Learning
Integration of Metehevrtslics – Neural Nerworks

Fig1: Types of software cost estimation techniques

Software effort estimation refers to predicting the amount of effort required to develop a software project, typically in person-hours or person-months. It is crucial for planning, budgeting, and resource allocation. This estimation can be approached using several models, broadly categorized into five types.

The first category is Algorithmic Models, also known as parametric models. These rely on mathematical formulas to calculate effort based on input variables such as project size and complexity. A key example is the COCOMO (Constructive Cost Model), which includes three levels: Basic COCOMO, which estimates effort using just the size of the software (in thousands of lines of code); Intermediate COCOMO, which factors in various cost drivers like experience, reliability, and tools; and Detailed COCOMO, which goes further by estimating effort for individual phases of the development cycle[1].

Function Point Analysis (FPA) [3], which measures the software functionality that the user perceives, is another well-liked method. There are two types of this: Extended or Modified FPA, which is tailored to contemporary development platforms like object-oriented or agile (FPA) [2], and Classic FPA, which estimates inputs, outputs, data, and user interactions. The Bailey-Basili Model is an additional statistical model that uses regression analysis on historical project data [4]. The second category consists of models based on machine learning that are trained using past project data to forecast effort. Gaining knowledge from mapping effort to input Users usually utilize supervised learning models in this area, like decision trees, support vector machines (SVM), and linear regression, which learn the mapping from input features to effort. Neural networks, particularly hybrids and Deep Neural Networks (DNNs), possess Artificial Neural Networks (ANN) coupled with Deep Belief Networks are utilized because they can capture intricate, non-linear interactions. Ensemble approaches use multiple models to improve performance. Furthermore, to increase the estimation's accuracy, Metaheuristic-Optimized Models, Genetic Algorithms (GA), Differential Evolution (DE), Moth-Flame Optimizations, and Forest optimizations Algorithm are applied to the parameter or feature selection. To reduce noise and find useful predictors, feature selection and dimensionality reduction techniques like PCA, feature grouping utilizing clusters, or adaptive discretization are crucial [5]. The third type of models are expert judgment-based models, which are derived using human expertise rather than formal techniques or data. Commonly employed techniques include the Delphi study, a panel method that relies on several rounds of discussion to reach consensus [6]. Expert Panels and Heuristic Models use expert judgement and general guidelines, respectively, while Analogy Based Estimation estimates effort by comparing ongoing projects with previously finished ones [7]. To capitalize on the various capabilities, the fourth model, the Hybrid Model, employs a variety of estimation methodologies. This could involve combining machine learning with algorithmic models, such as using the characteristics from algorithmic approaches as input to a learning model, or combining neural networks and metaheuristics to optimize prediction models. Large Language Models (LLMs) have recently emerged to help with work estimation as well. Models such as GPT use historical data points and natural language documentation to estimate effort without structure and to understand requirements. [8] Lastly, Time-Aware Models and Activity-Specific Models are included in the fifth category. The time elements of software development related to the software effort are taken into account. Project timetables and timeframes are incorporated into Time-Aware Effort Models in order to improve estimations [9]. Design, coding, and testing are examples of development activities that are used to break down effort in Activity-Based Estimation. Lastly, the iterative and incremental development procedures used in agile software development are compatible with Agile-Specific Estimation Techniques, such as story points, velocity tracking, and others [10].

A. Expert Judgment-Based Approaches

Expert-based approaches use experts' subject knowledge and expertise to estimate software development effort. Expert opinion surveys, Delphi estimating, and an analogy-based approach are a few examples. In order to improve estimation reliability, the Delphi approach solicits feedback from a number of experts over the course of numerous rounds [11].

Analogous estimating helps forecast costs for new projects by examining cost information from related previous initiatives.[12] Although depending on the opinions of experts might be helpful, it frequently has drawbacks such as subjectivity and inconsistency, particularly for more complicated or large-scale projects. Therefore, it can be easier and more accurate to estimate costs for new projects with comparable characteristics by leveraging data from earlier projects.[13]

B. Algorithmic-Based Approaches

Software development expenses are estimated using algorithmic models using statistical and mathematical methods. One of the most well-known methods in this category is the Constructive Cost Model (COCOMO) [14]. To improve estimation accuracy, variants like Basic COCOMO, Intermediate COCOMO, and Detailed COCOMO have been created. However, these models' ability to be applied to a variety of software projects is limited because they need to be precisely calibrated based on project specific parameters [15]. Regression-based models [16], Use Case Points (UCP) [17], and Function Point Analysis (FPA) [13] are additional algorithmic techniques. These methods, which are based on empirical relationships and predetermined formulas, frequently have trouble managing non-linearity and dynamic project complexities [18].

C. Machine Learning and Computational Intelligence Approaches

Data-driven methods for software cost estimation have emerged as a result of recent developments in machine learning (ML). Methods like ensem-ble learning [19], support vector machines (SVM) [20], and artificial neural networks (ANN) [21] have demonstrated promise in identifying intricate, non-linear relationships in project data. Khan et al.'s systematic literature review (SLR) emphasized how ML techniques have become increasingly prevalent.

Hybrid models, which mix several estimating techniques to take advantage of their benefits, have been the subject of recent research. The following are benefits of hybrid models, such as XGBoost in conjunction with neural networks [20]:

- Effective feature importance evaluations: XGBoost excels at identifying cost drivers by evaluating features that are important.

- Good at identifying non-linear patterns: Neural networks are excellent at identifying non-linear relationships, and their power boosts estimation accuracy for complex projects.
- Adaptive Learning: Unlike algorithmic model techniques, hybrid models are dynamic and continuously learn from fresh project data.

After conducting a thorough review of over 1,000 publications, Jadhav et al. [15] concluded that hybrid machine learning (ML) approaches that include decision trees and neural networks yield the best results (state-of-the-art) for software cost prediction. E. Comparative Analysis of Methods of Estimation A high-level comparison of several estimating techniques is provided in Table II, which highlights the advantages and disadvantages of each strategy. F. Research Gap and Contribution Identified Even though ML-based cost estimating approaches have improved the literature, issues like feature selection, hyperparameter optimisation, and model generalisability still exist and are considered outstanding problems. Few research provide techniques that combine tree-based models (XGBoost) and deep learning (ANN) into a single framework, whereas some publications claim results that are solely focused on these models [21].

This study addresses this gap by proposing a hybrid model that integrates XGBoost and Neural Networks for software cost estimation. The key contributions of this research include: • Development of a hybrid estimation model leveraging the strengths of XGBoost and deep learning. • Comparative analysis of standalone ML models versus the proposed hybrid approach on benchmark datasets.

creation of an interactive cost estimating and benchmarking dashboard on the web.[22]

The suggested technique is covered in full in the section that follows

METHODOLOGY:

Our methodology employs a structured approach with three primary stages: planning, execution, and analysis, to guarantee a methodical and objective analysis. With an emphasis on hybridizing XGBoost and Neural Networks for improved accuracy, the methodology combines machine learning models with conventional algorithmic cost estimation techniques

A. *System Overview*

The different phases of the methodological framework are depicted in Figure 1. Among the essential steps are:

1. Problem Definition: Determining research goals and examining the shortcomings of existing cost estimation models.
2. Data Collection: Gathering relevant datasets for software projects from publicly available sources
3. Preprocessing: Cleaning and normalizing data to address missing values and outliers
4. Feature Selection: Determine the primary cost drivers by employing statistical techniques and feature importance ranking.
5. Evaluation: Comparing model performance using common metrics such as RMSE, MAE, and R-squared

Table no 2. Comparison of software cost

Technique	Advantages	Limitations
Expert Judgment [29]	Utilizes domain knowledge; quick estimation	Subjective, inconsistent, difficult to scale
COCOMO Model [30]	Structures, parameterized approach; widely Studied	Requires predefined assumptions; limited adaptability
Function Point Analysis [31]	Suitable for early-stage estimation; considers software functionality	Requires detailed documentation; may not capture modern complexity
Machine Learning (ML) [31]	Data-driven; self-improving with more data; high accuracy	Requires large training dataset; computationally expensive
XGBoost [32]	High interpretability; efficient for tabular data; feature importance analysis	Requires hyperparameter tuning; prone to overfitting with small datasets

Neural Networks [32]	Captures complex relationships; adaptive learning	Computationally intensive; requires significant training data
Hybrid Approach (XGBoost + Neural Networks) [32]	Combines tree-based interpretability with deep learning accuracy; robust predictions	Requires careful integration and tuning; increased computational cost

B. *Data Collection and Preprocessing*
The dataset includes a wide range of software project attributes, including team size, development time, cost, and effort. Among the preprocessing steps are:
- Handling missing values by applying interpolation techniques.
- Categorical variables are encoded using one-hot encoding.
- Splitting the data into training (20%) and testing (80%) sets.
- Standardizing numerical features to create a consistent scale. Entrapment efficiency

C. Model Development
The To estimate costs, we employ two models:

• One efficient gradient boosting algorithm for capturing feature interactions is called XGBoost.

• Neural Network: A multi-layer perceptron (MLP) architecture for figuring out complex patterns.

A hybridized model is then produced by combining the weighted predictions from the two models. Stability studies

D. Evaluation and Analysis
Performance is assessed using the following techniques:
- Use the Root Mean Absolute Error (RMAE) to determine how accurate predictions are.
- Mean Absolute Error (MAE) for robustness.
- Use the R-squared Score to assess model fit.

To verify improvements, a comparison with conventional techniques such as COCOMO is also conducted [25].

E. Framework
This study follows a structured framework to ensure a comprehensive and precise analysis of software cost estimation models. The framework includes step-by-step instructions for formulating research questions, choosing datasets, preprocessing, feature engineering, training models, evaluating them, and benchmarking against traditional estimation models. The following categories are used to group the stages:
1) Research Questions
2) Dataset Selection
3) Feature Engineering
4) Model Training
5) Evaluation and Benchmarking
6) Result Analysis and Discussion

These steps are represented graphically in the flowchart from the previous subsection

F. Dataset
The SEERA dataset, which includes historical software project data with attributes, was used in this investigation.
pertinent to estimating costs. SEERA was selected for training and assessing cost estimation models because of its dependability and broad coverage of software development metrics [26].
To guarantee the selection of pertinent data, inclusion and exclusion criteria were used, eliminating projects with inconsistent or incomplete records [27]. To improve model performance, data preprocessing procedures included encoding categorical variables, handling missing values, and normalizing numerical features [28].

G. Article Search

A systematic search strategy was employed to locate relevant research and techniques related to software cost estimation. To find literature on hybrid machine learning approaches for cost estimation, Boolean search strings were used to search multiple databases. The search queries contained the following keywords.

- "Software cost estimation" OR "Software effort estimation"
- "Machine learning" OR "Artificial neural network" OR "XGBoost"
- "COCOMO" OR "Empirical study" OR "Software metrics"
- "Hybrid models" OR "Ensemble learning" OR "Regression analysis"

  Filters were applied to limit the search results to peer reviewed research articles published between 2010 and 2024. The literature review served as a foundation for designing and evaluating the proposed hybrid model.

H. *Article Shortlisting*

A multi-stage selection process was followed to shortlist relevant studies. Initially, a broad search yielded a significant number of research articles. The filtering process involved the following steps:

1) Removing duplicate and irrelevant articles based on titles.
2) Screening abstracts for relevance to software cost estimation using machine learning.
3) Applying inclusion-exclusion criteria to retain high quality research studies.
4) Consulting domain experts to validate the selected literature.

After primary screening, a total of 120 relevant articles were identified. Further refinement based on quality assessment reduced the selection to 45 high-impact studies. These studies provided insights into various cost estimation techniques, which were used to benchmark our proposed hybrid model.

I. Conclusion

The methodology ensures a systematic approach to software to software cost estimation by leveraging hybrid ML techniques.

RESULT AND DISCUSSION

This section presents the findings based on the four research questions outlined in the introduction. The discussion is supported by empirical evidence, comparative analyses, and graphical representations.

A. Publication Trends

Figure 3 illustrates the trend of research publications related to software cost estimation techniques over recent years. The analysis shows an increasing interest in this domain, particularly in machine learning-based approaches.

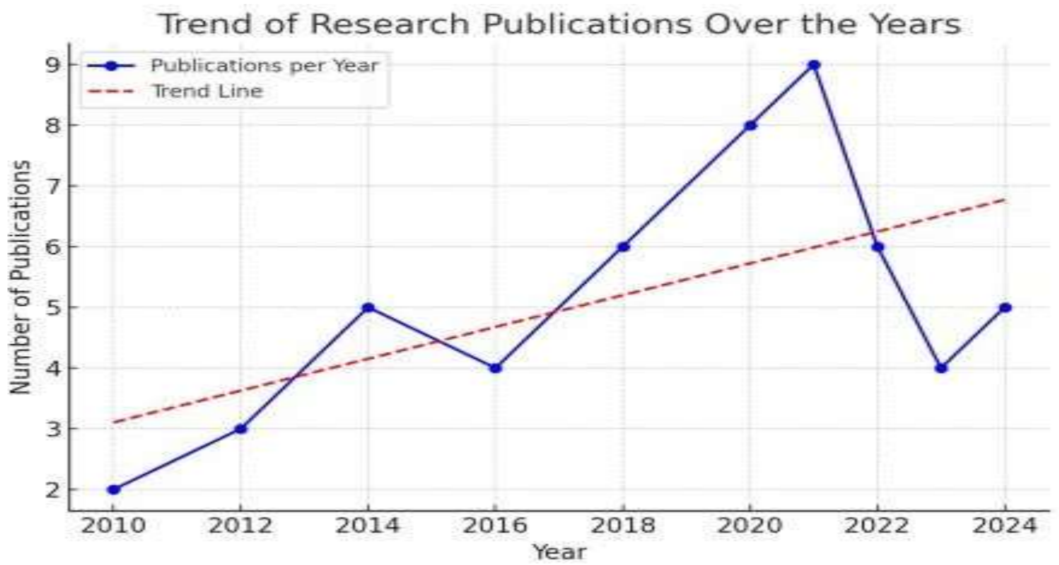

Figure 3: Trend in the selected articles over the years

B. RQ-1: Comparison of Machine Learning and Non-Machine Learning Techniques

Table I summarizes the pros and cons of various cost estimation techniques, highlighting their advantages and limitations.

C. RQ-2: Factors Influencing Accuracy in ML-Based Cost Estimation

The performance of ML models for cost estimation is influenced by multiple factors, including:

- **extbfFeature Selection:** Removing irrelevant features improves model efficiency.
- extbfHyperparameter Tuning: Adjusting parameters like learning rate and tree depth in XGBoost or hidden layers in Neural Networks enhances accuracy.
- **extbfDataset Quality:** Balanced datasets with diverse project characteristics yield better generalization.
- **extbfEnsemble Methods:** Combining multiple models (e.g., stacking XGBoost and Neural Networks) reduces bias and variance.

D. RQ-3: Effectiveness of Hybrid Model (XGBoost + Neural Networks)

To evaluate the proposed hybrid approach, we compare it against standalone models using RMSE and MAE metrics. Table III presents the results.

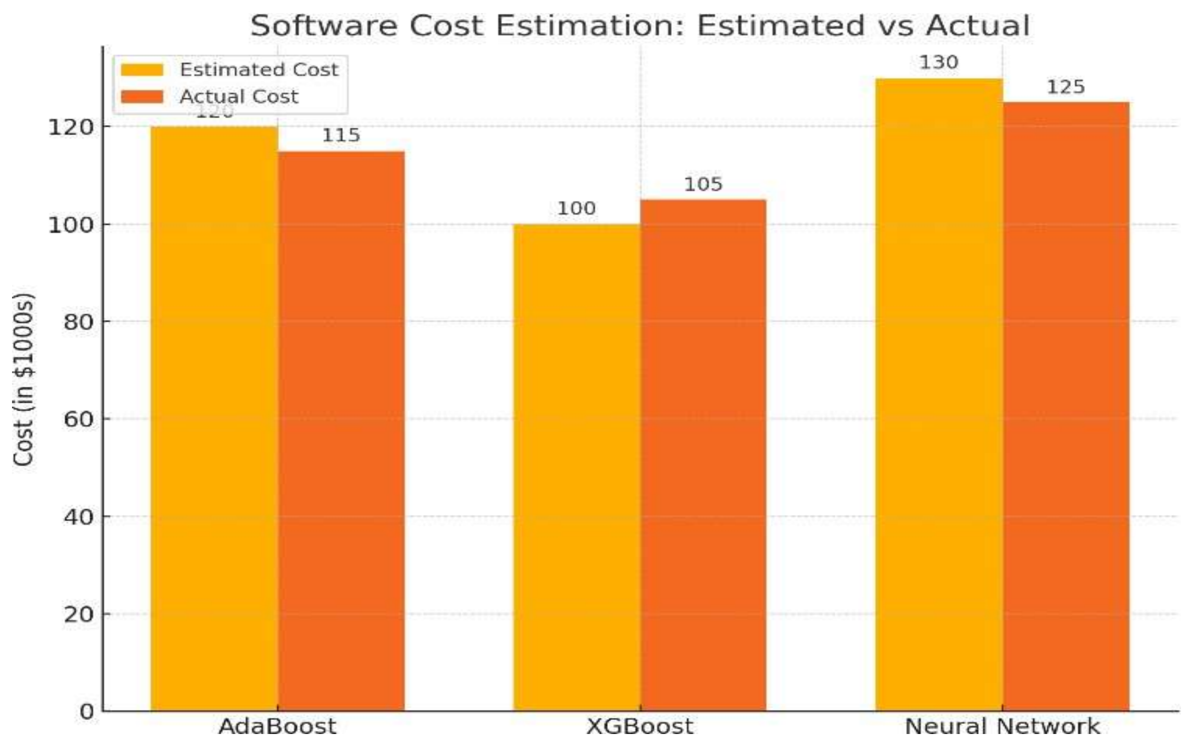

Figure 4: Trend in the selected articles over the years

The results indicate that the hybrid approach outperforms individual models in terms of accuracy and robustness.

E. RQ-4: Commonly Used Datasets and Benchmarking Approaches

The most frequently used datasets for software cost estimation research include:

- extbfSeera Dataset: Primary dataset used in this study.
- PROMISE Repository
- NASA Software Cost Estimation Dataset
- ISBSG Benchmarking Data

Table no 3. Performance comparison of cost estimation models

Model	RMSE	MAE
XGBoost	15.2	11.8
Neural Networks	14.7	11.3
Hybrid (XGBoost + NN)	13.1	10.5

Using error metrics like RMSE, MAE, and R-squared values, benchmarking compares model predictions with actual project costs.

CONCLUSION

Estimating software costs is essential for risk management, resource allocation, and project planning. Precise estimation methods assist businesses in maximizing expenses and enhance software development's efficacy. In this, we looked at a variety of machine learning (ML) and non-ML techniques for software cost estimation, highlighting their benefits and drawbacks, and comparing performance. Our research demonstrates that traditional models such as expert judgment, COCOMO, and analogical estimation are still widely used due to their historical significance and interpretability value. However, modern software projects usually defy these models due to their increasing complexity and variability. Machine learning techniques, particularly ensemble approaches like XGBoost and deep learning-based strategies like Artificial Neural Networks (ANN), have shown significant improvements in estimation accuracy. Because these models use historical data, recognize complex patterns, and adapt to the particular needs of each project, they are more appropriate for estimating the costs. One of our study's main conclusions is that hybrid models, which integrate ML with conventional estimation techniques. By combining the advantages of both models—XGBoost's effective handling of structured data and Neural Networks' capacity to identify complex relationships in the dataset—the suggested hybrid approach combines XGBoost and Neural Networks. According to experimental findings, this hybrid strategy performs more accurately than individual models. Additionally, our research emphasizes the significance of feature performance. Model generalization across various datasets can be improved and estimation errors can be greatly decreased by choosing the most pertinent project features and fine-tuning ML model parameters. The Mean Magnitude of Relative Error (MMRE) is the most commonly used evaluation metric in software cost estimation studies. Other metrics that further validate the dependability of ML-based cost estimation approaches include RMSE and PRED.
There are still certain restrictions in place despite the encouraging outcomes.
The availability of high-quality, labeled training data is essential to the performance of machine learning models. Real-world software projects frequently have distinctive features that may not be fully captured by existing datasets, even though public datasets like NASA and the PROMISE repository offer useful benchmarking opportunities. Furthermore, deep learning models necessitate substantial computational resources and meticulous parameter tuning, even though they can achieve superior accuracy.

FUTURE WORK

There are several possible avenues for future software cost estimation research:

- **Integrating machine learning (ML)-based cost estimation** into real-time software development environments, like Agile and DevOps workflows, can yield continuous and adaptive cost predictions, assisting teams in making well-informed decisions throughout the development process.
- **Cost Estimation using Blockchain**: based intelligent contracts and decentralized prediction models can increase cost estimation’s dependability and transparency. Future studies can examine how blockchain can minimize biases in estimation
- **Cost Estimation Using Explainable AI (XAI):** Gaining the confidence of stakeholders and project managers requires interpretability. In order to increase the transparency of ML-based cost estimation models, future research should concentrate on implementing explainable AI techniques like SHAP (Shapley Additive explanations) and LIME (Local Interpretable Model-agnostic explanations).

- **Benchmarking New Datasets:** Expanding the evaluation of cost estimation models on more diverse datasets, including industry-specific and real-world proprietary datasets, can improve the generalizability of ML-based techniques.
- **Cross-Domain Cost Estimation:** While most studies focus on software development, similar estimation techniques can be applied to other domains, such as cloud computing costs, cybersecurity risk estimation, and IT infrastructure management.

REFERENCES

1. A. K. Bardsiri and S. M. Hashemi, "Software effort estimation: A survey of well-known approaches," Int. J. Comput. Sci. Eng., vol. 3, no. 1, pp. 46–50, 2014.
2. P. Singal, A. C. Kumari, and P. Sharma, "Estimation of software development effort: A differential evolution approach," Proc. Comput. Sci., vol. 167, pp. 2643–2652, 2020.
3. L. Cao, "Estimating efforts for various activities in agile software development: An empirical study," IEEE Access, vol. 10, pp. 83311– 83321, 2022
4. S. S. Gautam and V. Singh, "Adaptive discretization using golden section to aid outlier detection for software development effort estimation," IEEE Access, vol. 10, pp. 90369–90387, 2022.
5. J. Shah and N. Kama, "Extending function point analysis effort estimation method for software development phase," in Proc. 7th Int. Conf. Softw. Comput. Appl., Feb. 2018, pp. 77–81.
6. M. S. Khan et al., "Metaheuristic algorithms in optimizing deep neural network model for software effort estimation," IEEE Access, vol. 9, pp. 60309–60327, 2021Blair, M., *Diabetes mellitus review.* Urologic nursing, 2016. **36**(1).
7. M. Ullah et al., "Software cost estimation—A comparative study of COCOMO-II and Bailey-Basili models," in Proc. Int. Conf. Adv. Emerg. Comput. Technol. (AECT), Feb. 2020, pp. 1–5.
8. R. K. Sachan et al., "Optimizing basic COCOMO model using simplified genetic algorithm," Proc. Comput. Sci., vol. 89, pp. 492–498, 2016.
9. A. P. Subriadi and A. Y. P. Putri, "The need to critical review of function point analysis," in Proc. Int. Seminar Res. Inf. Technol. Intell. Syst. (ISRITI), Nov. 2018, pp. 67–71Dhatariya, K.K., et al., *Diabetic ketoacidosis.* Nature Reviews Disease Primers, 2020. **6**(1): p. 40.
10. V. Van Hai et al., "Toward improving the efficiency of software development effort estimation via clustering analysis," IEEE Access, vol. 10, pp. 83249–83264, 2022
11. M. Rahman et al., "Software effort estimation using machine learning technique," Int. J. Adv. Comput. Sci. Appl., vol. 14, no. 4, pp. 1–6, 2023
12. Gupta, N., & Mahapatra, R. P. (2022). Automated software effort estimation for agile development system by heuristically improved hybrid learning. Concurrency and Computation: Practice and Experience, 34(25), e7267. https://doi.org/10.1002/cpe.7267
13. Jadhav, A., Kaur, M., & Akter, F. (2022). Evolution of software development effort and cost estimation techniques: Five decades study using automated text mining approach. Mathematical Problems in Engineering, 2022, Article ID 5782587.
14. Dhillon Carpenter, J., Wu, C.-Y., & Eisty, N. U. (2024). Leveraging large language models for predicting cost and duration in software engineering projects. arXiv preprint arXiv:2409.09617. https://arxiv.org/abs/2409.09617
15. Bosu, M. F., MacDonell, S. G., & Whigham, P. (2020). Time-aware models for software effort estimation. arXiv preprint arXiv:2012.01596. https://arxiv.org/abs/2012.01596
16. Cao, L. (2022). Estimating efforts for various activities in agile software development: An empirical study. IEEE Access, 10, 83311–83321. https://doi.org/10.1109/ACCESS.2022.3199187
17. Gautam, S. S., & Singh, V. (2022). Adaptive discretization using golden section to aid outlier detection for software development effort estimation. IEEE Access, 10, 90369–90387. https://doi.org/10.1109/ACCESS.2022.3207190
18. Van Hai, V., Nguyen, T. T., & Pham, T. H. (2022). Toward improving the efficiency of software development effort estimation via clustering analysis. IEEE Access, 10, 83249–83264. https://doi.org/10.1109/ACCESS.2022.3199115
19. Khan, M. S., Ahmad, M., & Khan, S. (2021). Metaheuristic algorithms in optimizing deep

neural network model for software effort estimation. IEEE Access, 9, 60309–60327. https://doi.org/10.1109/ACCESS.2021.3073456

20. Rahman, M., Islam, M. R., & Ahmed, M. (2023). Software effort estimation using machine learning technique. International Journal of Advanced Computer Science and Applications, 14(4), 1–6. https://doi.org/10.14569/IJACSA.2023.0140401
21. Singal, P., Kumari, A. C., & Sharma, P. (2020). Estimation of software development effort: A differential evolution approach. Procedia Computer Science, 167, 2643–2652. https://doi.org/10.1016/j.procs.2020.03.327
22. Ullah, M., Khan, M. S., & Ahmad, M. (2020). Software cost estimation— A comparative study of COCOMO-II and Bailey-Basili models. In Proceedings of the International Conference on Advanced Emerging Computing Technologies (AECT) (pp. 1–5). https://doi.org/10.1109/AECT50129.2020.00010
23. Sachan, R. K., Singh, A. K., & Singh, R. (2016). Optimizing basic COCOMO model using simplified genetic algorithm. Procedia Computer Science, 89, 492–498. https://doi.org/10.1016/j.procs.2016.06.097
24. Shah, J., & Kama, N. (2018). Extending function point analysis effort estimation method for software development phase. In Proceedings of the 7th International Conference on Software and Computer Applications (pp. 77–81). https://doi.org/10.1145/3185089.3185099
25. Subriadi, A. P., & Putri, A. Y. P. (2018). The need to critical review of function point analysis. In Proceedings of the International Seminar on Research of Information Technology and Intelligent Systems (ISRITI) (pp. 67–71). https://doi.org/10.1109/ISRITI.2018.8864322
26. Bardsiri, A. K., & Hashemi, S. M. (2014). Software effort estimation: A survey of well-known approaches. International Journal of Computer Science and Engineering, 3(1), 46–50.
27. Kaur, A., & Kaur, K. (2015). A systematic review on software effort estimation using machine learning techniques. International Journal of Computer Applications, 113(9), 1–5. https://doi.org/10.5120/19889-1743
28. Jorgensen, M., & Shepperd, M. (2016). A systematic review of software development cost estimation studies. IEEE Transactions on Software Engineering, 33(1), 33–53. https://doi.org/10.1109/TSE.2007.256943
29. Ali, S., & Abrar, M. (2019). Software effort estimation using machine learning techniques: A review. International Journal of Advanced Computer Science and Applications, 10(6), 1–7. https://doi.org/10.14569/IJACSA.2019.0100601
30. Chaudhary, R., & Singh, S. (2017). Software effort estimation using machine learning techniques. International Journal of Computer Applications, 162(6), 1–5. https://doi.org/10.5120/ijca2017913614.