

Machine learning approach for cost estimation in software project planning

Ajay Jaiswal¹, Jagdish Raikwal²

¹Department of Computer Science and Engineering, Prestige Institute of Engineering, Management and Science, Indore, India

²Department of Information Technology, Institute of Engineering and Technology, Devi Ahilya Vishwavidyalaya, Indore, India

Article Info

Article history:

Received Dec 13, 2024

Revised Mar 27, 2025

Accepted Jul 2, 2025

Keywords:

Accuracy

Deep learning

Machine learning

Project planning and budget

Software cost estimation

ABSTRACT

Successful organizing and handling of software projects depends extensively on accurate cost estimation. This study explores the effectiveness of machine learning models in estimating software project costs using datasets like Desharnais, Maxwell, and Kitchenham, aiming to prevent project delays and resource misallocation. It shows how model selection has a major impact on forecast accuracy through a thorough assessment. An R-squared value (R²) of 0.804 indicates that the support vector machine (SVM) model performs exceptionally well in the Desharnais dataset. On the Maxwell dataset, linear regression (LR) stands out with a minimum mean absolute error (MAE) of 0.483 and the greatest R² value of 0.607, while SVM has the lowest root mean squared error (RMSE) of 0.537. Similarly, on the Kitchenham dataset, LR and SVM are the top performers, with MAE of 0.201 and RMSE of 0.274, respectively, and R² values of around 0.929. These findings highlight the importance of tailored model selection for accurate cost prediction, as LR and SVM continuously demonstrate reliability across varied datasets. ML techniques like LR and SVM can enhance software project planning and management by providing accurate cost estimation, with future research exploring ensemble learning and deep learning methodologies.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Ajay Jaiswal

Department of Computer Science and Engineering

Prestige Institute of Engineering, Management and Research

Indore, India

Email: ajay.jaiswal55555@gmail.com

1. INTRODUCTION

Software measurement (SM) involves measuring software characteristics typically related to the product, method, and resources utilized in software development. These indicators can be utilized in project management systems to assist software developers in effectively managing their projects, hence reducing issues such as cost overruns and scheduling deficiencies [1]. One of the most challenging aspects of managing a project is estimating software. Accurately estimating the time, money, and effort needed to complete an endeavor has been a challenge for project managers for a long time. It is challenging to predict these factors early in a project's lifecycle when there is a lot of uncertainty about the product's features, and the boundaries of each initiative need to be defined [2]. Software project management relies heavily on estimation, particularly in the early stages of software development. The first step is to estimate how much time, money, and effort are expected to be required to finish the software project [3], [4]. Consequently, inefficient resource utilization and lengthy delivery delays may result from software cost overestimation. However, inadequate workforce numbers, going over budget, and late delivery time might occur from underestimating software expenses [5]. Planning is the most crucial phase in project management because it

estimates the time and money needed to finish a project properly [6]. The most well known aspects of software evaluation are software cost and effort estimation. The term "cost estimation" refers to the process of determining an approximate total cost for a project, program, or product using current data. Accurate cost estimation is critical for each project type to avoid unexpectedly excessive expenditures. Accurate estimates are essential for decision makers to manage risks, allocate resources, and generate precise project schedules [7]. The development of estimation approaches, which have gone from basic statistical models to complex ML algorithms, reveals the industry's dedication to managing costs and project success.

Planning and budgeting for a project requires an accurate assessment of software costs. Efficient, high-quality, and precisely estimated data are essential for successful regulation and oversight. Improve the precision and efficacy of project planning and budgeting with cutting-edge cost estimation software that relies on ML. ML algorithms generate more precise and adaptable forecasts by analyzing past data, project attributes, and other variables. ML Models, like random forests, decision trees, SVM, and logistic regression, improve project planning and budgeting [8]-[10]. Organizations can optimize resource allocation, make better decisions, and confidently and efficiently navigate software development complexity with data-driven insights.

In addition, the uncertain nature of software development, characterized by ever-changing requirements and evolving technology, further complicates the estimation process. ML provides a promising solution as it enables learning from past projects and the identification of patterns that would help predict future project costs more reliably. This research is motivated by the objective of utilizing ML to enhance the precision of software project cost estimates, thereby enabling project managers to make informed decisions. This study proposes a new approach for estimating software costs using ML techniques. It targets better predictions in terms of accuracy and dependability on expenditure during software development procedures so as to enable corporate executives to make up correct funding decisions and manage other activities related to resourcing appropriately. The goal of this research is to provide a new cost estimation framework for software project planning using machine learning (ML) techniques. The main contributions of this study are:

- Designing an innovative ML framework that supports the efficient and accurate forecasting of costs in software development.
- The use of datasets such as Desharnais, Kitchenham, and Maxwell for empirical validation of the proposed approach.
- Data driven insights for improved project planning and resource management.

These contributions aim to close the gap between traditional estimation approaches and modern dynamic requirements for software projects.

2. RELATED WORK

Accurate software effort estimation is crucial for software project management [11]. Software effort estimation refers to the technique of forecasting the effort required to build software products in terms of expenses [12]. Project planning and budget allocation are two areas where prior research in SCE has demonstrated its fundamental importance. Effective monitoring and regulation of software development projects requires precise estimates of cost, precision, and quality. Conventional models, such as the constructive cost model II (COCOMO) [13], [14], depend significantly on reliable and accurate data from the past. Olu-Ajayi [15]. These findings are unique and promising, contributing to effective business planning and risk reduction compared to previous research. Draz *et al.* [16] emphasize the essential role of planning and budgeting in software projects. A hybrid approach was proposed in this study by integrating Gray Wolf Optimization for software effort estimation. When it came to SCE, another study [17] used a hybrid model that used the tabu search (TS) method [18] with the invasive weed optimization (IWO) algorithm [19]. The TS algorithm worked better with the IWO algorithm's solutions [20]. Prior to that, in 2023, an analysis was conducted to compare the current taxonomies and methodologies employed in the estimation of software costs using neural networks [21]. A review found that the mean magnitude of relative error (MMRE), percentage relative error deviation (PRED), and root mean squared error (RMSE) are the most commonly utilized metrics for evaluating ML-SCE models [22]. Alauthman *et al.* [23] discussed software development cost estimation regression model selection. It emphasized using models that match the software development methodology and dataset utilized in estimation. Govinda *et al.* [24] used ML to calculate software cost for project managers using standard input. Akhbardeh *et al.* [25] examined the processes for computable elements that affect software cost and presented research that used ML methodologies to construct a credible estimation method. Table 1 describes the estimating approach and the contribution of the recognised papers. We have taken accuracy values under the aforementioned system from a variety of datasets and methodologies in order to investigate accurate performance analysis.

Table 1. Prediction accuracy of primary research on standalone methods

Study	Author(s)	Dataset	Estimation technique/contribution	MMRE	PRED
[26]	Malhotra and Jain	499 software projects	Bagging, ANN, DT, SVM, and linear regression (LR) were evaluated and contrasted on a software project dataset.	0.17	52
[27]	Sharma and Singh	4 software projects	Made use of random forests, multilayer perceptrons, and support vector machines.	0.30	72.09
[28]	Pospieszny <i>et al.</i>	11 variable software projects	The ensemble of support vector machines, neural networks, and general linear models was averaged.	0.13	76.91
[29]	Pandey <i>et al.</i>	SAMOA	Provided a useful method for selecting the best estimate technique for app effort estimates from of four well-liked methods: GA, MLR, MLP-NN, and nave.	0.9	94
[30]	Dan <i>et al.</i>	COCOMO-I, NASA	Particle swarm optimisation was used to improve a COCOMO integrated (PSO) artificial neural network (ANN) model.	0.40	55.10

The successful planning and execution of software development projects is dependent on accurate software project cost estimation, which also influences resource management, budget allocation, and project schedules. Expert judgment, analogous estimation, and parametric models like COCOMO are examples of traditional cost estimation techniques that frequently struggle with adaptability, precision, and ability to handle complex unpredictable relationships that arise in software development processes. Novel approaches have been developed in this field of study, and they require regular comparative assessments. Accurate software cost estimation is critical to the success of software projects because it gives information about the risks and challenges associated with development. Comparative results show that the proposed model outperforms existing techniques across all datasets and evaluation criteria. The findings were quite promising for forecasting software cost prediction. The enormous diversity of ML approaches has led to comparisons and eventually, the integration of various techniques. Determining the most effective estimating methods has become essential for improving the project development process due to their many benefits. When working on complex projects or projects with changing requirements, accuracy is sometimes an issue with evolving ML techniques. Accurate estimation of costs is essential to executing projects on time and within budget, and numerous companies make significant investments in this area in order to ensure rapid growth and satisfied customers. Apart from this fact, these challenges are made even more challenging by the dynamic changes that can occur in any software project, such as evolving requirements, improved technology, or even shifts in the team's skills. The goal of this research is to develop a ML based approach for evaluating planning costs for software projects.

3. PROPOSED METHOD

The proposed method for estimating software development costs involves nine steps framework. Figure 1 depicts the proposed method for software development project cost estimation.

Data collection and pre-processing:

- Gather the Desharnais, Kitchenham, and Maxwell datasets, which contain historical data on software projects, including attributes such as project size, effort, and duration.
- Preprocess the datasets by handling missing values and outliers and standardizing numerical features.

Word2Vec feature extraction:

- Convert textual data (if any), such as project descriptions, and requirements, into numerical vectors using Word2Vec.
- Word2Vec can capture the semantic meaning of words and phrases within the text, providing dense vector representations for each word.

Merge Word2Vec features with numerical features:

- Combine the Word2Vec features with the existing numerical features from the datasets to form a comprehensive feature set.

Feature selection using recursive feature elimination (RFE):

- Implement RFE to select the most important features from the combined feature set.
- RFE recursively removes features, fitting the model multiple times and assessing feature importance until the optimal feature subset is identified.

Data splitting:

- Split the datasets into training and testing sets, ensuring that each dataset is divided appropriately to maintain its integrity.

Model training:

- Train different ML models, including LSTM, LR, SVM, FNN, RNN, and DT, using the training dataset.

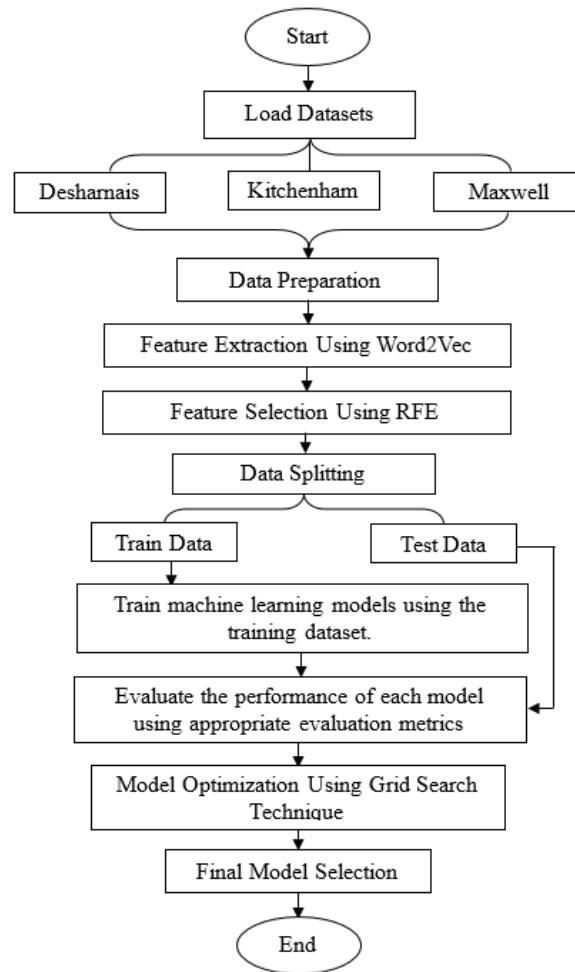


Figure 1. Proposed method

Model evaluation:

- Evaluate the performance of each model using appropriate evaluation metrics such as MAE, MSE and R^2 error.
- Compare the performance of each ML model to identify the best performing approach for each dataset.

Model optimization:

- Refine the hyperparameters of chosen models using methods such as grid search or random search to enhance performance.
- Ensure that the models are optimized to generalize well on unseen data.

Final model selection:

- Select the best performing model for each dataset based on evaluation metrics and optimization results.

4. METHOD

The two primary components that make up this section are software costing techniques and a proposed method that introduces a novel and inventive framework with the intention of enhancing the software cost estimating process.

- a) Techniques used: The subsequent section explains the techniques employed in the proposed method for cost estimation.
- b) Word2Vec: Word2vec is simpler and faster to learn than other methods. Sentence word semantics can be detected via Word2vec. The text processing in Word2vec is done by a two-layer neural network. The two primary techniques for learning Word2vec are CBOW and Skip Gram. While the CBOW model uses surrounding contexts to forecast the word, the Skip Gram uses those same contexts to forecast the word itself [31]. It improves feature selection, model training, and evaluation by bridging unstructured text and

structured numerical features. The quality of Word2Vec characteristics can considerably affect model performance in forecasting software project results.

- c) Recursive feature elimination (RFE): Irrelevant features are common in large datasets [32]. The inefficiency of the classification algorithm is impacted by recurring features. Determine which variables are crucial for making accurate forecasts using RFE. A method that iteratively searches for a target number of attributes is "recursive". Figure 2 is a visual representation of the RFE Workflow.

Next, the model is retrained with the updated feature set to improve classification accuracy and remove less important features. The loop continues as long as there are additional features to be included. RFE is an essential part of the suggested method for software development cost estimation since it improves the efficiency and accuracy of ML models' predictions. Using RFE, the most relevant features are selected from the set of features that includes both the original numerical features and the textual characteristics generated by Word2Vec.

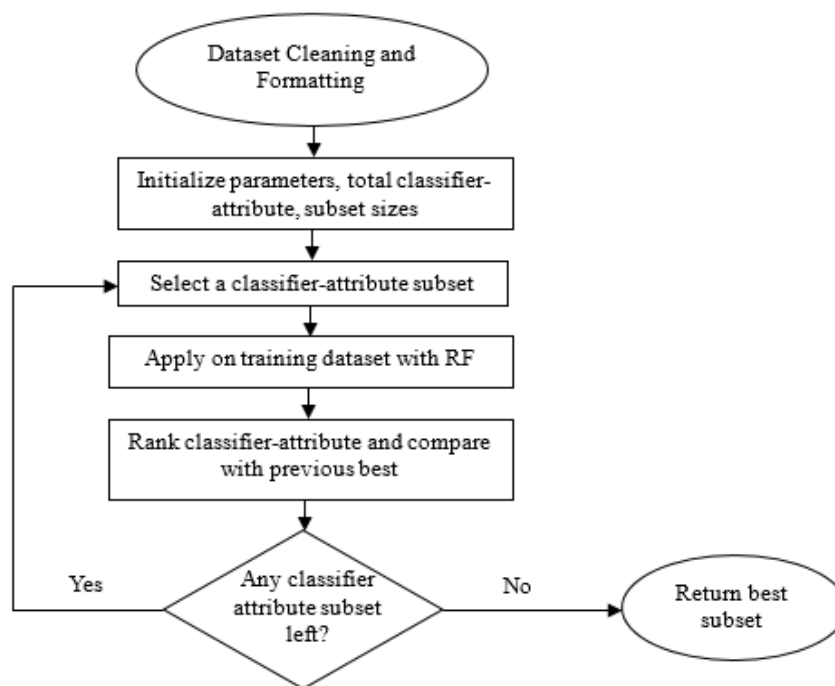


Figure 2. RFE workflow schematic [32]

- d) Linear regression model: One way to find the relationships between the two sets of variables is to use a LR model [33]. Software cost estimation involves creating predictions about the dependent variable, which is software cost, using measurements for the independent variables, which are products, projects, and procedures. The LR models are utilized in the study to conduct an analysis of the amount of money required to develop software. A single independent variable, a single LR, is used to predict the cost dependent variable. A straight line is computed to lessen the discrepancy between the actual multiple LR, which employs a large number of independent variables, and provides a formula for calculating the estimated cost using a linear combination of the metrics [34], [35].

$$y_i = a + b_1x_1 + b_2x_2 + \dots b_nx_n + \varepsilon_i \quad (1)$$

The values of x_1 through x_n are the independent variables, whereas y_i is the dependent variable, where $i=1 \dots m$. The variables b_1 through b_m stand for the regression coefficients, and the letter a represents the intercept (eq.1). Linear models are an excellent starting point but cannot guarantee flawless data linearity. Validating historical efforts helps assess their accuracy. Actionable and explicable cost estimation formulas can be constructed from current measures using LR.

- e) Recurrent neural networks (RNN): RNNs have shown promising potential for software cost estimation tasks. RNNs operate by processing sequential data, where the output at each time step is influenced by the current input and the previous hidden state. RNNs are particularly wellsuited for handling various

factors that influence software costs, such as project size, complexity, and team experience. This recurrent nature allows RNNs to maintain an internal memory, making them well-suited for problems involving sequential information, such as software project data [36].

- f) Long short-term memory (LSTM): The LSTM network is a type of RNN that was created for consecutive data processing in deep learning. The problem of exploding and vanishing gradients was successfully addressed by an LSTM, which was designed to depict sequences with long-range dependencies [37] accurately. Each gate is taught to end the input value by the system, which does this by continuously providing error signals to them [38].
- g) Decision tree (DT): DT are hierarchical tree-structured models used for cost and effort estimation in software [39]. They recursively split the data based on attribute tests represented by internal nodes, with branches depicting test outcomes and leaf nodes containing the predicted estimates. This structure allows decision trees to model the impact of factors on project cost and effort [40].
- h) Support vector machine (SVM): The SVM model is a versatile tool in software cost estimation, adept at handling both classification and regression tasks. Whether facing linear or nonlinear problems, SVM effectively partitions data by constructing a hyperplane that separates classes [41].
- i) Feed-forward neural network (FFNN): For software cost estimation is the FFNN. As its name implies, data flows in only one direction, from input to output. The most basic type of artificial neural network, known as FFNN, can only go forward and cannot reverse its direction of operation. FFNN has 3 layers (input, hidden, and output layer). FFNN allows the network to discover intricate patterns and relationships within the data by incorporating multiple hidden layers.
- j) Feature selection using recursive feature elimination: Implement RFE to select the most important features from the combined feature set. Moreover, FFNN may feature direct (linear) connections between the input and output layers, facilitating the mapping of input variables to output predictions without direct connections between individual input and output units [42].

5. RESULTS AND DISCUSSION

Results: The outcomes of the systematic research conducted for software project cost estimation are presented in this section. It looks at pertinent features that come from selection and extraction as well as the outcomes of ML model evaluation for estimation.

- a) Experimental setup: The experiment was conducted on a laptop computer with an Intel Core i7 processor, 64GB of RAM. The experiment was carried out using a variety of tools. Google Drive was utilized to upload data sets for the experiment, which were then uploaded to Google Colab. This study uses Python to present, explain, depict, and analyze the data, as well as train and test the algorithm.
- b) Evaluation criteria: In addition to standardized error measurements such as, MAE, MSE, and RMSE, two commonly used software estimating criteria mean magnitude relative error, or MMRE, and percentage relative error deviation, or PRED were employed to evaluate the produced models. Above all, they allow for the comparison of results from multiple prediction models and datasets since they are scale and unit independent. Both are based on MRE, which is explained below and measures the disparity between actuals and estimates.

Mean absolute error (MAE): MAE is a widely used statistic that finds the mean squared difference between expected and actual values by averaging the squared disparities. It evaluates the overall accuracy of the predictive model in (2).

$$MAE = \left(\frac{1}{n}\right) * \sum_{i=1}^n (\text{predicted}_i - \text{actual}_i)^2 \quad (2)$$

where n is the total number of observations, y is the actual value of sample i , and y^{\wedge} is the prediction made by the model for sample i .

RMSE: To compute the value of it, you will require the actual values and their expected values, shown in (3). Where n is the total number of data points, $^{\wedge}2$ is the square of the difference, and is the sum of square differences in the datasets.

$$RSME = \sum (\text{predicted}_i - \text{actual}_i)^2 / n \quad (3)$$

Magnitude of relative error (MRE): Determine the Magnitude of Relative Error (4) for each data point in order to assess the degree of estimating error in a single estimate.

$$MRE = |(\text{predicted}_i - \text{actual}_i)| / \text{actual} \quad (4)$$

Mean magnitude of the relative error (MMRE): The mean magnitude of the relative error (5) is the average proportion of the absolute values of the relative errors across the complete data set.

$$MMRE = (100/N) * \sum |predicted_i - actual_i| / actual_i \quad (5)$$

where, N = total number of estimate

PRED(n) Prediction Accuracy: To determine the accuracy rate PRED(n), divide the total number of data points with an MRE of 0.25 or less (represented by k) by the total number of data points in the set (represented by n).

$$PRED(x) = (100/N) * \sum_{i=1}^N 1 \text{ if } MRE_i \leq n/100, 0 \text{ otherwise} \quad (6)$$

with $n = 0.25$, the in (6) is $PRED(n) = k/n$ [43], [44].

- c) Extracting features for estimating software costs: The Word2Vec approach is used in this study to extract pertinent features from the Desharnais, Kitchenham, and Maxwell datasets. After they have been extracted, these features are added to the cost calculation procedure. Table 2, which presents the characteristics extracted from three datasets.

Table 2. List of features extracted using Word2Vec

Dataset	Extracted features
Desharnais	Project, Points NonAdjust, ManagerExp, Adjustment, Year-End, Length Transactions, PointsAjust, Effort, TeamExp
Maxwell	Effort, Har, Year, Duration, App, T14, Source, Nlan, T06, T05, T15, T09, Size, Time
Kitchenham	Adjfp, Estimate method, Client code, Estimate, Projecttype, Duration, Effort

- d) Selecting features for estimating software costs: To ensure that the predictive models are trained on the most useful variables while reducing the danger of overfitting, this study used the RFE approach to identify the most important features from the three datasets. Regression analysis makes extensive use of RFE for regularization and variable selection. It functions by repeatedly removing aspects that, according to a preset criterion, are judged unnecessary or redundant. This iterative procedure yields a more comprehensible model until just a selection of features with the highest predictive potential is left. Table 3 displays the chosen characteristics that were gathered from three distinct datasets using the RFE technique [40].

Table 3. List of features using RFE

Dataset	Selected features
Desharnais	Project, Transactions, Effort, TeamExp, PointsA-just, PointsNonAdjust
Maxwell	App, Effort, Har, Source, Nlan, T05, T09, T15, Year, Duration, Time, Size
Kitchenham	Effort, Project type, Client code, Duration, Estimate, Adjfp

Performance evaluation outcomes: This section discusses the performance metrics of various ML models on all three datasets. It describes different kinds of errors calculated for each model [45].

- a) On Desharnais dataset: Different ML models, including LR, FNN, LSTM, RNN, DT, and SVM, were evaluated on the Desharnais dataset using various performance metrics, as shown in Table 4. The SVM model demonstrated the highest R^2 of 0.804, indicating the best performance based on this metric. In terms of RMSE, FNN achieved the lowest RMSE value of 0.293, while LR had the lowest MAE of 0.263, suggesting their effectiveness in minimizing prediction errors for this dataset.

Table 4. Error metrix obtained on desharnais dataset

S. No.	Error metrix	MAE	R2	RMSE
1	LR	0.263	0.778	0.353
2	FNN	0.384	0.737	0.293
3	LSTM	0.541	0.571	0.397
4	RNN	0.429	0.672	0.331
5	DT	0.432	0.532	0.372
6	SVM	0.331	0.804	0.331

Figure 3, illustrates a comparison of different ML models based on error metrics across the Desharnais dataset. It further emphasizes the performance variations among the models, highlighting their strengths and weaknesses in predicting software project costs.

- b) On Maxwell dataset: Error metrics for various ML models applied to the Maxwell dataset are summarized in Table 5. Among the models evaluated, LR demonstrates the lowest MAE of 0.483, indicating the smallest absolute difference between predicted and actual costs. LR also exhibits the highest R^2 value of 0.607, suggesting a good fit to the data. Conversely, the LSTM model yields the highest MAE of 0.933, implying larger deviations between predicted and actual costs. Furthermore, SVM stands out with the lowest RMSE of 0.537, indicating overall accuracy in predictions. These results indicate that LR and SVM perform relatively better in terms of both MAE and RMSE on the Maxwell.

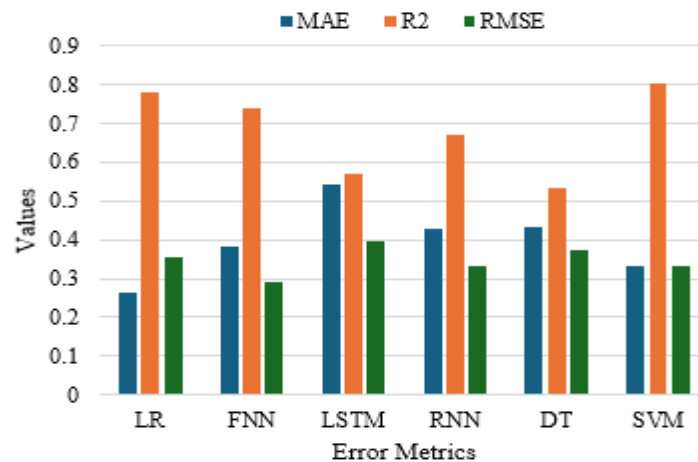


Figure 3. Performance metrics comparison on the desharnais

Figure 4 depicts the error metrics of several ML models applied to the Maxwell dataset. The graph provides a comparative analysis of the performance of each model, shedding light on their effectiveness in software cost estimation on the Maxwell Dataset.

- c) On Kitchenham dataset: Table 6 depicts the performance of various ML models evaluated on the Kitchenham dataset using different error metrics. The LR and SVM models demonstrated the best performance, with LR having the lowest MAE of 0.201 and SVM having the lowest RMSE of 0.274. Both LR and SVM also achieved a high R^2 of around 0.929, indicating a good fit to the data.

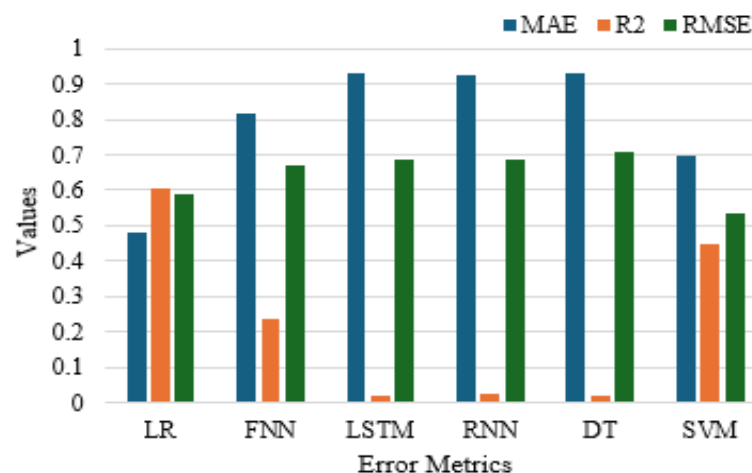


Figure 4. Performance metrics comparison on the Maxwell

Table 5. Error metrix obtained on maxwell dataset

S. No.	Error metrix	MAE	R2	RMSE
1	LR	0.483	0.607	0.588
2	FNN	0.818	0.239	0.669
3	LSTM	0.933	0.021	0.689
4	RNN	0.927	0.024	0.686
5	DT	0.929	0.017	0.706
6	SVM	0.331	0.804	0.331

The FNN and RNN exhibited relatively higher MAE of 0.249 and 0.307, respectively, and lower R^2 values. Notably, the LSTM and DT models showed relatively poorer performance and exhibited considerably low R-squared values of 0.841 and 0.761, respectively, suggesting a poor fit to the Kitchenham dataset. Figure 5 displays the error metrics of multiple ML models when applied to the Kitchenham dataset. The graph provides a comparative analysis of the performance of each model, demonstrating their effectiveness in software cost estimation on the Kitchenham dataset.

Table 6. Error metrix obtained on kitchenham dataset

S. No.	Error metrix	MAE	R2	RMSE
1	LR	0.201	0.929	0.275
2	FNN	0.249	0.910	0.309
3	LSTM	0.431	0.841	0.389
4	RNN	0.307	0.858	0.387
5	DT	0.326	0.761	0.406
6	SVM	0.202	0.929	0.274

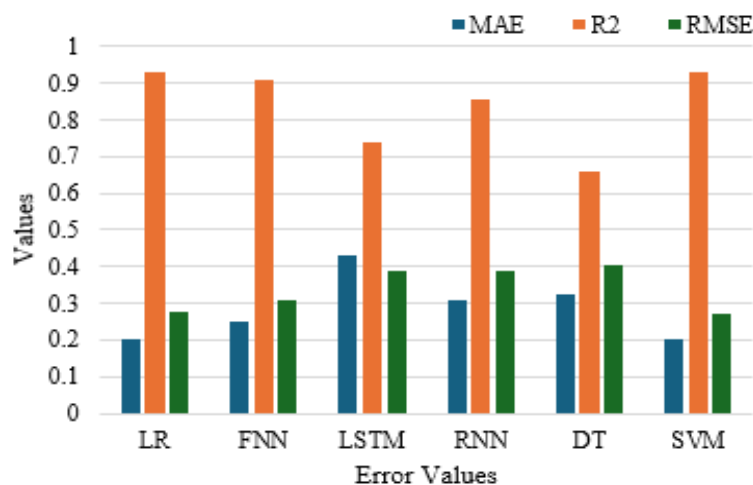


Figure 5. Performance metrics comparison on the Kitchenham Dataset

Discussion: Performance evaluation outcomes showed that different ML models performed better on the Desharnais dataset, with the SVM model showing the highest R^2 value of 0.804, indicating the best performance based on this metric. On the Maxwell dataset, LR demonstrated the lowest MAE of 0.483, indicating the smallest absolute difference between predicted and actual costs. LSTM model yielded the highest MAE of 0.933, implying larger deviations between predicted and actual costs. SVM stood out with the lowest RMSE of 0.537, indicating overall accuracy in predictions. On the Kitchenham dataset, the LR and SVM models demonstrated the best performance, with LR having the lowest MAE of 0.201 and SVM having the lowest RMSE of 0.274. FNN and RNN exhibited relatively higher MAE and lower R^2 values, while LSTM and DT models showed poorer performance and low R-squared values. However, further research could explore ensemble learning techniques and deep learning architectures to enhance the accuracy and robustness of software cost estimation models [46], [47].

6. CONCLUSION

The comparison study results demonstrate significantly higher accuracy only in three stages of evaluation in the presence of numerous learning methodologies. Based on the Performance Metrics Comparison of Kitchenham dataset results obtained from the evaluation of various ML models on the Desharnais, Maxwell, and Kitchenham datasets, it is evident that the choice of model significantly impacts the accuracy and effectiveness of software cost estimation. In the Desharnais dataset, the SVM model outperformed others with the highest R^2 value of 0.804, indicating superior predictive capability. Conversely, the Maxwell dataset showcased LR and SVM as the top performers, with LR demonstrating the lowest MAE of 0.483 and the highest R^2 value of 0.929 and SVM exhibiting the lowest RMSE of 0.537. On the other hand, the Kitchenham dataset illustrated LR and SVM as the most reliable models, displaying the lowest MAE and RMSE values and high R^2 values of 0.201, 0.275, 0.929, and 0.202, 0.274, and 0.929 respectively. These findings emphasize the importance of selecting appropriate ML models tailored to specific datasets for accurate software cost estimation. Additionally, LR and SVM consistently emerged as strong performers across all datasets, suggesting their reliability and effectiveness in this domain. Thus, leveraging ML techniques, particularly LR and SVM, holds promise for enhancing the planning and management of software projects through more precise cost estimation methodologies.

ACKNOWLEDGEMENTS

The author would like to extend deepest appreciation to “Dr. Manojkumar Deshpande” and “Dr. Piyush Chaudhary” of the PIEMR, Indore, for his contributions to this study. To everyone who made this study possible, the author would like to extend their sincere gratitude. No specific grant from a public, private, or nonprofit funding organization was obtained for this study.

FUNDING INFORMATION

The authors state that no funding was involved in the research presented in this manuscript.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Ajay Jaiswal	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓
Jagdish Raikwal	✓	✓				✓		✓	✓	✓	✓	✓	✓	

C : Conceptualization

M : Methodology

So : Software

Va : Validation

Fo : Formal analysis

I : Investigation

R : Resources

D : Data Curation

O : Writing - Original Draft

E : Writing - Review & Editing

Vi : Visualization

Su : Supervision

P : Project administration

Fu : Funding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declared no potential conflicts of interest concerning this article’s research, authorship, and publication.

DATA AVAILABILITY

The data that support the findings of this study are available upon request / publicly available at / included within the article or its supplementary materials.

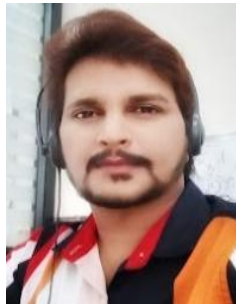
REFERENCES




- [1] M. Rahman, P. P. Roy, M. Ali, T. Gonçalves, and H. Sarwar, “Software effort estimation using machine learning technique,” *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 4, pp. 822–827, 2023, doi: 10.14569/IJACSA.2023.0140491.
- [2] A. Zai, R. Butt, and S. Nawaz, “A survey of software quality metrics for the software measurement process,” *Journal of Software Engineering and Intelligent Systems*, vol. 2, no. 1, pp. 49–56, 2017.
- [3] A. O. Sousa *et al.*, “Applying machine learning to estimate the effort and duration of individual tasks in software projects,” in *IEEE Access*, vol. 11, pp. 89933–89946, 2023, doi: 10.1109/ACCESS.2023.3307310.

- [4] E. Papatheocharous, S. Bibi, I. Stamelos, and A. S. Andreou, "An investigation of effort distribution among development phases: A four-stage progressive software cost estimation model," *Journal of Software: Evolution and Process*, vol. 29, no. 10, Jun. 2017, doi: 10.1002/smr.1881.
- [5] M. Azzeh, A. B. Nassif, and S. Banitaan, "Comparative analysis of soft computing techniques for predicting software effort based use case points," *IET Software*, vol. 12, no. 1, pp. 19–29, Feb. 2018, doi: 10.1049/iet-sen.2016.0322.
- [6] A. B. Nassif, M. Azzeh, A. Idri, and A. Abran, "Software development effort estimation using regression fuzzy models," *Computational Intelligence and Neuroscience*, vol. 2019, pp. 1–17, Feb. 2019, doi: 10.1155/2019/8367214.
- [7] A. Karimi and T. J. Gandomani, "Software development effort estimation modeling using a combination of fuzzy-neural network and differential evolution algorithm," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 1, pp. 707–715, Feb. 2021, doi: 10.11591/ijece.v11i1.pp707-715.
- [8] Bilgaiyan, "A systematic review on software cost estimation in agile software development," *Journal of Engineering Science & Technology Review*, vol. 10, no. 4, 2017.
- [9] K. P. Selvam and M. Brorsson, "Can semi-supervised learning improve prediction of deep learning model resource consumption?," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 6, pp. 74–83, 2024, doi: 10.14569/IJACSA.2024.0150610.
- [10] M. Rahman, T. Goncalves, and H. Sarwar, "Review of existing datasets used for software effort estimation," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 7, pp. 921–931, 2023, doi: 10.14569/IJACSA.2023.01407100.
- [11] D. Lock, "Project management," *Routledge*, 2020.
- [12] A. A. Fadhil, R. G. H. Alsarraj, and A. M. Altaie, "Software cost estimation based on dolphin algorithm," *IEEE Access*, vol. 8, pp. 75279–75287, 2020, doi: 10.1109/ACCESS.2020.2988867.
- [13] M. Ahmed *et al.*, "A hybrid model for improving software cost estimation in global software development," *Computers, Materials and Continua*, vol. 78, no. 1, pp. 1399–1422, 2024, doi: 10.32604/cmc.2023.046648.
- [14] P. Singal, A. C. Kumari, and P. Sharma, "Estimation of software development effort: a differential evolution approach," *Procedia Computer Science*, vol. 167, pp. 2643–2652, 2020, doi: 10.1016/j.procs.2020.03.343.
- [15] R. Olu-Ajayi, "An investigation into the Suitability of k-Nearest Neighbour (k-NN) for software effort estimation," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 6, 2017, doi: 10.14569/ijacsa.2017.080628.
- [16] M. M. Draz, O. Emam, and S. M. Azzam, "A predictive model for software cost estimation using ARIMA algorithm," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 7, pp. 656–665, 2024, doi: 10.14569/IJACSA.2024.0150764.
- [17] H. M. Mohamedyusuf, H. O. Sharif, and M. I. Ghareb, "A new approach for software cost estimation with a hybrid tabu search and invasive weed optimization algorithms," *UHD Journal of Science and Technology*, vol. 8, no. 1, pp. 42–54, Feb. 2024, doi: 10.21928/uhdjt.v8n1y2024.pp42-54.
- [18] V. K. Prajapati, M. Jain, and L. Chouhan, "Tabu search algorithm (TSA): a comprehensive survey," in *Proceedings of 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things, ICETCE 2020*, Feb. 2020, pp. 222–229, doi: 10.1109/ICETCE48199.2020.9091743.
- [19] D. Kumar, B. G. R. Gandhi, and R. K. Bhattacharjya, "Introduction to invasive weed optimization method," in *Modeling and Optimization in Science and Technologies*, vol. 16, Springer International Publishing, 2020, pp. 203–214.
- [20] S. I. Khaleel, "Designing a tool to estimate software projects based on the swarm intelligence," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 4, pp. 524–538, Aug. 2021, doi: 10.22266/ijies2021.0831.46.
- [21] R. Ramaekers, R. Silhavy, and P. Silhavy, "Software cost estimation using neural networks," in *Lecture Notes in Networks and Systems*, vol. 722 LNNS, Springer International Publishing, 2023, pp. 831–847.
- [22] M. M. I. Shamim, A. B. b. A. Hamid, T. E. Nyamasvisva, N. S. B. Rafi, "Advancement of artificial intelligence in cost estimation for project management success: a systematic review of machine learning, deep learning, regression, and hybrid models," *Modelling*, vol. 6, no. 2, p. 35, 2025, doi: 10.3390/modelling6020035.
- [23] M. Alauthman *et al.*, "Machine learning for accurate software development cost estimation in economically and technically limited environments," *International Journal of Software Science and Computational Intelligence*, vol. 15, no. 1, pp. 1–24, Oct. 2023, doi: 10.4018/ijssci.331753.
- [24] S. Govinda, "Framework for estimating software cost using improved machine learning approach," in *Lecture Notes on Data Engineering and Communications Technologies*, vol. 114, Springer Nature Singapore, 2022, pp. 713–725.
- [25] F. Akhbardeh and H. Reza, "A survey of machine learning approach to software cost estimation," in *IEEE International Conference on Electro Information Technology*, May 2021, vol. 2021-May, pp. 405–408, doi: 10.1109/EIT51626.2021.9491912.
- [26] R. Malhotra and A. Jain, "Software effort prediction using statistical and machine learning methods," *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 1, 2011, doi: 10.14569/ijacsa.2011.020122.
- [27] P. Sharma and J. Singh, "Machine learning based effort estimation using standardization," in *2018 International Conference on Computing, Power and Communication Technologies, GUCON 2018*, Sep. 2019, pp. 716–720, doi: 10.1109/GUCON.2018.8674908.
- [28] 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, Mar. 2018, doi: 10.1016/j.jss.2017.11.066.
- [29] M. Pandey, R. Litoriya, and P. Pandey, "Validation of existing software effort estimation techniques in context with mobile software applications," *Wireless Personal Communications*, vol. 110, no. 4, pp. 1659–1677, Sep. 2020, doi: 10.1007/s11277-019-06805-0.
- [30] Z. Dan, "Improving the accuracy in software effort estimation: Using artificial neural network model based on particle swarm optimization," in *Proceedings of 2013 IEEE International Conference on Service Operations and Logistics, and Informatics, SOLI 2013*, Jul. 2013, pp. 180–185, doi: 10.1109/SOLI.2013.6611406.
- [31] W. Liu, Z. Cao, W. Jun, and W. Xiaoyi, "Short text classification based on Wikipedia and Word2vec," in *2016 2nd IEEE International Conference on Computer and Communications, ICC3 2016 - Proceedings*, Oct. 2017, pp. 1195–1200, doi: 10.1109/CompComm.2016.7924894.
- [32] J. Al Faysal *et al.*, "XGB-RF: A hybrid machine learning approach for IoT intrusion detection," *Telecom*, vol. 3, no. 1, pp. 52–69, Jan. 2022, doi: 10.3390/telecom3010003.
- [33] Linear regression basics for absolute beginners, towardsai, May 29, 2020, Available: <http://towardsai.net/p/data-science/linear-regression-basics-for-absolute-beginners>.
- [34] R. Silhavy, P. Silhavy, and Z. Prokopova, "Analysis and selection of a regression model for the Use Case Points method using a stepwise approach," *Journal of Systems and Software*, vol. 125, pp. 1–14, Mar. 2017, doi: 10.1016/j.jss.2016.11.029.




- [35] Deep Learning - introduction to recurrent networks, AILABPAGE, January 8, 2019, Available: <http://ailabpage.com/2019/01/08/deep-learning-introduction-to-recurrent-neural-networks/>.
- [36] A. Kaushik, N. Choudhary, and Priyanka, "Software cost estimation using LSTM-RNN," in *Advances in Intelligent Systems and Computing*, vol. 1164, Springer Singapore, 2021, pp. 15–24.
- [37] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, C. Ragkhitwetsagul, and A. Ghose, "Automatically recommending components for issue reports using deep learning," *Empirical Software Engineering*, vol. 26, no. 2, Feb. 2021, doi: 10.1007/s10664-020-09898-5.
- [38] T. Kangwantrakool, K. Viriyayudhakorn, and T. Theeramunkong, "Software development effort estimation from unstructured software project description by sequence models," *IEICE Transactions on Information and Systems*, vol. E103D, no. 4, pp. 739–747, Apr. 2020, doi: 10.1587/transinf.2019IIP0014.
- [39] E. Alpaydin, "Introduction to machine learning," *MIT Press*, 2020.
- [40] S. J. Russell and P. Norvig, "Artificial intelligence: a modern approach," *Pearson*, 2016.
- [41] Non linear SVM, machine learning tutorial, 04 May 2024, Available: <https://www.scaler.com/topics/machine-learning/non-linear-svm/>.
- [42] N. A. Zakaria, A. R. Ismail, A. Y. Ali, N. H. M. Khalid, and N. Z. Abidin, "Software project estimation with machine learning," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, pp. 726–734, 2021, doi: 10.14569/IJACSA.2021.0120685.
- [43] A. Negi, C. V. Verma, and Y. Tayyebi, "Artificial intelligence empowered language models: a review," in *Lecture Notes in Networks and Systems*, vol. 891, Springer Nature Singapore, 2024, pp. 535–548.
- [44] M. Jørgensen *et al.*, "When should we (not) use the mean magnitude of relative error (MMRE) as an error measure in software development effort estimation?," *Information and Software Technology*, vol. 143, p. 106784, 2022.
- [45] B. S. Liang *et al.*, "A project model for software development," *Journal of Information Science and Engineering*, vol. 16, pp. 423–446, 2000.
- [46] A. R. Gili *et al.*, "Making programmer effective for software development teams: an extended study," *Journal of Information Science and Engineering*, vol. 33, pp. 1447–1463, 2017.
- [47] D. S. Al-Azzawi *et al.*, "Evaluation of genetic algorithm optimization in machine learning," *Journal of Information Science and Engineering*, vol. 36, pp. 231–241, 2020.

BIOGRAPHIES OF AUTHORS



Ajay Jaiswal    received M.E. (Master of Engineering, 2012) in Computer Engineering from the Institute of Engineering and Technology, DAVV, Indore-MP (India). He is currently pursuing a Ph.D. from the Department of Computer Engineering, IET DAVV, Indore (India). His research interests include software engineering and machine learning. He can be contacted at email: ajay.jaiswal55555@gmail.com.



Dr. Jagdish Raikwal    is working as an Associate Professor at the Institute of Engineering and Technology, DAVV, Indore-MP (India). His research interests include data mining, data warehousing, and software engineering and its applications. He can be contacted at email: jraikwal@ietdavv.edu.in.