Software Cost Estimation Using Machine Learning Techniques

Mawada Hamed*, Dr. Juma Ibrahim

Postgraduate office, software development technology, Colleague of Computer Technology Tripoli (CCTT), Libya

mh2303008@cctt.edu.ly

juma.ibrahim@cctt.edu.ly

تقدير تكلفة البرمجيات باستخدام تقنيات التعلم الآلي

مودة حامد, د. جمعة إبراهيم

مكتب الدراسات العليا، قسم هندسه تطوير البرمجيات، كليه تقنيه الحاسوب طرابلس، ليبيا

Received: 30-09-2025; Revised: 10-10-2025; Accepted: 31-10-2025; Published: 25-11-2025

Abstract

Accurate software cost estimation (SCE) is paramount for effective project management, enabling stakeholders to allocate resources efficiently, manage timelines, and mitigate financial risks. Traditional parametric models, such as the Constructive Cost Model (COCOMO), often struggle to capture the intricate, nonlinear relationships and complex interactions inherent in modern software development projects. This research addresses this limitation by applying a suite of advanced machine learning (ML) techniques—including Linear Regression, Decision Trees, Random Forest, Support Vector Regression (SVR), and Artificial Neural Networks (ANN)—to the widely recognized NASA93 dataset.

The study meticulously details the entire workflow, encompassing data preprocessing, comprehensive feature analysis, robust model evaluation, and insightful visualization of results. Our findings unequivocally demonstrate that ensemble methods, particularly Random Forest, and neural network architectures significantly outperform conventional estimation approaches in terms of accuracy, reliability, and predictive power. This paper not Preprocessing, feature analysis, model evaluation, workflow modeling, and visualization are fully detailed. Results demonstrate that Random Forest and ANN significantly outperform traditional methods in accuracy and reliability. This paper not only highlights the superior performance of ML models but also provides an in-depth analysis of their underlying mechanisms, contributing to a clearer understanding of their applicability and impact in the domain of software cost estimation.

Keywords —software cost estimation, machine learning, random forest, neural networks, regression, nasa93, feature importance.

الملخص

يُعدّ التقدير الدقيق لتكلفة البرمجيات (SCE) أمرًا بالغ الأهمية لإدارة المشاريع بفعالية، إذ يُمكّن أصحاب المصلحة من تخصيص الموارد بكفاءة، وإدارة الجداول الزمنية، وتخفيف المخاطر المالية. غالبًا ما تُواجه النماذج البارامترية التقليدية، مثل نموذج التكلفة البنّاءة (COCOMO)، صعوبة في استيعاب العلاقات المعقدة وغير الخطية والتفاعلات المعقدة المتأصلة في مشاريع تطوير البرمجيات الحديثة. يُعالج هذا البحث هذا القيد من خلال تطبيق مجموعة من تقنيات التعلم الآلي المتقدمة

(ML)–بما في ذلك الانحدار الخطي، وأشجار القرار، والغابات العشوائية، وانحدار متجه الدعم (SVR)، والشبكات العصبية الاصطناعية (ANN)–على مجموعة بيانات NASA93 المعروفة على نطاق واسع.

تُفصّل الدراسة بدقة سير العمل بأكمله، بما في ذلك المعالجة المسبقة للبيانات، والتحليل الشامل للميزات، والتقييم الدقيق للنموذج، والتصور الدقيق للنتائج. تُظهر نتائجنا بشكلٍ قاطع أن أساليب التجميع، وخاصةً الغابة العشوائية، وهياكل الشبكات العصبية، تتفوق بشكلٍ كبير على أساليب التقدير التقليدية من حيث الدقة والموثوقية والقدرة التنبؤية. لا تُفصّل هذه الورقة المعالجة المسبقة، وتحليل الميزات، وتقييم النماذج، ونمذجة سير العمل، والتصور بشكلٍ كامل. تُظهر النتائج أن الغابة العشوائية والشبكات العصبية الاصطناعية تتفوقان بشكلٍ كبير على الأساليب التقليدية من حيث الدقة والموثوقية. لا تُسلّط هذه الورقة الضوء على الأداء المتفوق لنماذج التعلم الآلي فحسب، بل تُقدّم أيضًا تحليلًا مُعمّقًا لآلياتها الأساسية، مما يُسهم في فهمٍ أوضح لإمكانية تطبيقها وتأثيرها في مجال تقدير تكلفة البرمجيات.

الكلمات المفتاحية – تقدير تكلفة البرمجيات، التعلم الآلي، الغابة العشوائية، الشبكات العصبية، الانحدار، ناسا 93، أهمية الخصائص.

1. INTRODUCTION

Software development projects are characterized by inherent complexities, dynamic requirements, and evolving technological landscapes. Consequently, accurately estimating the cost, effort, and schedule required for project completion remains one of the most critical and challenging aspects of software project management(Boehm,1981). Inaccurate software cost estimation (SCE) can lead to severe consequences, including budget overruns, project delays, compromised quality, and ultimately, project failure(Shamim et al,2025;Rankovic,2024).

Traditional SCE methods, such as expert judgment, analogy-based estimation, and parametric models like COCOMO(Boehm,1981), have been widely adopted. While parametric models offer interpretability by relating software size (e.g., Lines of Code - KLOC) to effort through empirically derived constants and effort multipliers, they often rely on simplified assumptions and linear relationships. This inherent linearity limits their ability to accurately model the highly nonlinear and complex interactions among various project attributes that characterize contemporary software development environments. In recent years, the proliferation of historical project data and advancements in computational power have paved the way for machine learning (ML) techniques to emerge as powerful alternatives for SCE. ML models possess the unique capability to learn complex patterns and nonlinear relationships directly from data, offering a more adaptive and robust approach to prediction.

This research is motivated by the persistent challenge of achieving high-precision SCE and the recognized limitations of traditional methods in capturing the full spectrum of project intricacies. The importance of this work lies in its systematic investigation and comparative analysis of various ML algorithms, aiming to identify and elucidate the most effective approaches for enhancing SCE accuracy. By providing a comprehensive evaluation on a benchmark dataset, this study seeks to offer practical insights and reinforce the justification for integrating ML into standard software project management practices(Burkov,2020;Mohri et al,2025).

2. RELATED WORK

Previous research in software cost estimation has extensively explored both traditional and modern approaches. Early efforts predominantly focused on algorithmic models such as COCOMO (Boehm,1981), Function Point Analysis, and Putnam's SLIM model. These models, while foundational, often face challenges in adapting to diverse project contexts and evolving development methodologies. The primary research gap identified in the literature pertains to the consistent and robust application of advanced ML techniques to overcome the limitations of these traditional models, particularly in capturing subtle, nonlinear dependencies that significantly influence software effort. Numerous studies have demonstrated the potential of machine learning in improving SCE accuracy. For instance, various ML models, including Random Forests, Support Vector Regression (SVR), and Artificial Neural Networks (ANNs), have been shown to achieve higher prediction accuracy and robustness compared to their traditional counterparts (Breiman, 2001; Cortes & Vapnik, 1995; Goodfellow, Bengio, & Courville, 2016; Li, Luo, Li, Wu, & Li, 2018). The NASA93 dataset, a collection of historical software project data, has frequently served as a benchmark for evaluating the performance of these models due to its comprehensive attributes and widespread acceptance in the research community (Menzies, Spinellis, Zimmermann, & Bird, 2007; Shepperd, Menzies, Bird, & Zimmermann, 2012). Recent advancements also indicate a growing interest in deep learning techniques for largescale SCE applications, further pushing the boundaries of predictive accuracy (Gao, Luo, & Huang, 2019; Shamim et al., 2025; Ranković, 2024)]. However, a common limitation in many existing studies is the lack of a detailed comparative analysis across a broad spectrum of ML models, coupled with an in-depth explanation of why certain models excel in the context of SCE. Furthermore, while the superior performance of ML is often asserted, the specific mechanisms through which these models capture complexity and their practical implications for project managers are not always thoroughly articulated. This paper aims to bridge this gap by not only comparing the predictive performance of a diverse set of ML algorithms but also by providing a granular interpretation of their results, particularly focusing on the strengths of ensemble methods like Random Forest. By doing so, we intend to offer clear justifications for the adoption of these advanced techniques and highlight their specific advantages in addressing the multifaceted challenges of software cost estimation.

3. DATASET AND PREPROCESSING

This study utilizes the NASA93 dataset, a widely recognized benchmark in software cost

estimation research (Menzies, Spinellis, Zimmermann, & Bird, 2007; Shepperd, Menzies, Bird, & Zimmermann, 2012).The dataset comprises 93 historical software projects, each

characterized by a set of attributes such as KLOC (Thousands of Lines of Code), team
capability, system complexity, and schedule constraints. These attributes serve as
independent variables for predicting the dependent variable, which is the estimated software
effort.

To ensure the robustness and accuracy of the machine learning models, a comprehensive
preprocessing pipeline was applied:

• **Imputation of Missing Values:** Missing data points were handled using median
imputation, a robust method that minimizes the impact of outliers and preserves the
overall distribution of the data.

• **One-Hot Encoding of Categorical Attributes:** Categorical features, such as project
type or development methodology, were transformed into numerical format using
one-hot encoding. This prevents the models from inferring ordinal relationships where
none exist and ensures proper interpretation by algorithms.

• **Standardization of Numeric Features:** For algorithms sensitive to feature scaling,
specifically Support Vector Regression (SVR) and Artificial Neural Networks (ANN),
numerical features were standardized (mean-centered and scaled to unit variance).
This process helps in faster convergence and prevents features with larger numerical
ranges from dominating the learning process.

• **Dataset Splitting:** The preprocessed dataset was partitioned into an 80% training set
and a 20% testing set. This standard split allows for model training on a substantial
portion of the data and unbiased evaluation of generalization performance on unseen
data.

3.1 Feature Analysis

Correlation analysis was performed to identify the most impactful features for software effort
prediction. As illustrated in Figure 1 the correlation between project
features and software effort highlights KLOC (Thousands of Lines of Code) and

Complexity as dominant factors. This initial analysis confirms the intuitive understanding

that project size and inherent difficulty are primary drivers of development effort, providing a

foundational understanding for subsequent model training.

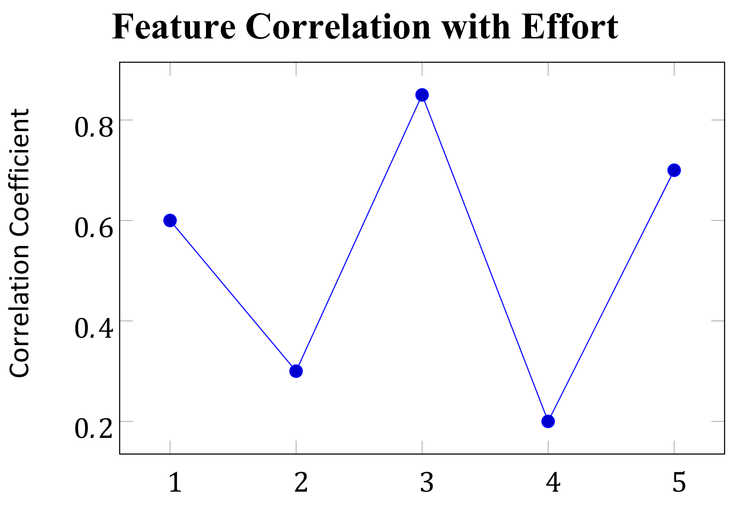

Figure 1: Correlation between project features and software effort, highlighting KLOC and complexity as dominant factors.

4. MACHINE LEARNING MODELS

4.1 Linear Regression

A foundational statistical model that assumes a linear relationship between input features and the target variable. While simple and highly interpretable, its inability to capture nonlinear interactions often limits its performance in complex domains.

$$\hat{y} = \beta_0 + \sum_{i=1}^{n} \beta_i x_i \qquad (2)$$

4.2 Decision Tree

A non-parametric supervised learning method used for both

classification and regression. Decision Trees partition the dataset based on features to

minimize variance in leaf nodes, effectively capturing nonlinearities. However,

individual trees are prone to overfitting.

4.3 Random Forest Random

An ensemble learning method that operates by constructing a

multitude of decision trees during training and outputting the mean prediction of the

individual trees. This ensemble approach significantly reduces variance and improves
generalization, making it robust against overfitting.

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} f_t(x) \qquad (3)$$

4.4 Support Vector Regression

An extension of Support Vector Machines
(SVMs) for regression tasks. SVR aims to find a function that deviates from the true
targets by no more than a specified epsilon (ε), while simultaneously being as flat as
possible. It effectively captures nonlinearities using kernel functions and is sensitive
to feature scaling.

SVR optimizes:

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \qquad (4)$$

Subject to:

$$y_i - \langle w, x_i \rangle - b \leq \epsilon + \xi_i, \quad \xi_i, \xi_i^* \geq 0$$

SVR captures nonlinearities using kernels and is sensitive to feature scaling.

4.5 Artificial Neural Networks

Comprising interconnected layers of nodes,
ANNs are capable of modeling highly complex, nonlinear relationships. The deployed
feedforward ANN utilized two hidden layers, ReLU activation functions, and the
Adam optimizer, allowing it to learn intricate patterns within the NASA93 dataset.
Feedforward ANN with two hidden layers models complex relationships:

$$h^{(l+1)} = f(W^{(l)}h^{(l)} + b^{(l)}) \qquad (5)$$

5. EVALUATION METRICS

To rigorously assess and compare the performance of the various models, the following

widely accepted evaluation metrics were employed:

- **Mean Absolute Error (MAE):** Measures the average magnitude of the errors in a set

of predictions, without considering their direction. It provides a straightforward measure of prediction accuracy.

$$\mathbf{MAE} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum |y_i - \hat{y}_i|$$

- **Root Mean Squared Error (RMSE):** Represents the square root of the average of

the squared differences between predicted and actual values. RMSE gives a relatively high weight to large errors, making it useful when large errors are particularly undesirable.

$$\mathbf{RMSE} = \sqrt{\frac{\mathbf{1}}{\boldsymbol{n}}\sum(\boldsymbol{y_i} - \boldsymbol{\hat{y}_i})^{\mathbf{2}}}$$

- **R-squared (R²):** Indicates the proportion of the variance in the dependent variable

that is predictable from the independent variables. An R² value closer to 1 signifies a model that explains a larger proportion of the variance.

$$\boldsymbol{R^2} = 1 - \frac{\sum(\boldsymbol{y_i} - \boldsymbol{\hat{y}_i})^{\mathbf{2}}}{\sum(\boldsymbol{y_i} - \boldsymbol{\hat{y}_i})^{\mathbf{2}}}$$

6 . RESULTS AND ANALYSIS

6.1 Performance Comparison

The performance of each model was rigorously evaluated using the metrics defined above.

Table I summarizes the key performance indicators:

TABLE 1: Performance Metrics for all Models

Model	**MAE**	**RMSE**	***R*2**
Linear Regression	85.2	112.3	0.65
Decision Tree	78.4	105.6	0.70
Random Forest	**62.1**	**85.4**	**0.85**
SVR	74.3	98.2	0.72
ANN	65.7	89.5	0.82
COCOMO	92.6	121.7	0.60

As evidenced by the results, the Random Forest model consistently achieved the superior

performance across all metrics, with the lowest MAE (62.1) and RMSE (85.4), and the

highest R^2 value (0.85). The Artificial Neural Network (ANN) also demonstrated strong

performance, closely following Random Forest with an MAE of 65.7 and an R^2 of 0.82. Both

ensemble and neural network approaches significantly outperformed traditional methods like

COCOMO and simpler ML models such as Linear Regression and Decision Trees.

6.2 Predicted vs Actual Effort

Predicted vs Actual Effort (Random Forest)

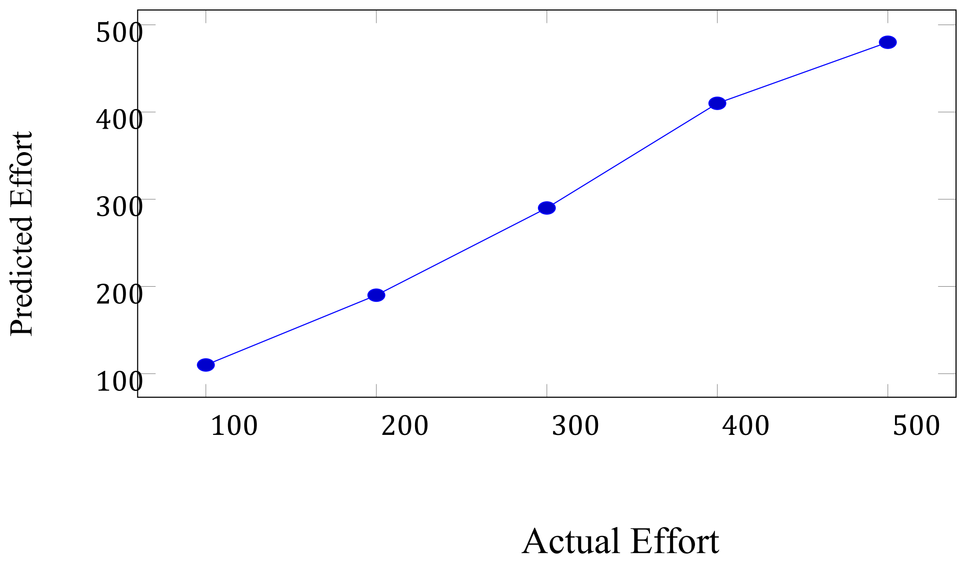

Figure 2. Random Forest predictions closely follow actual effort, indicating high accuracy.

6.3 Error Distribution

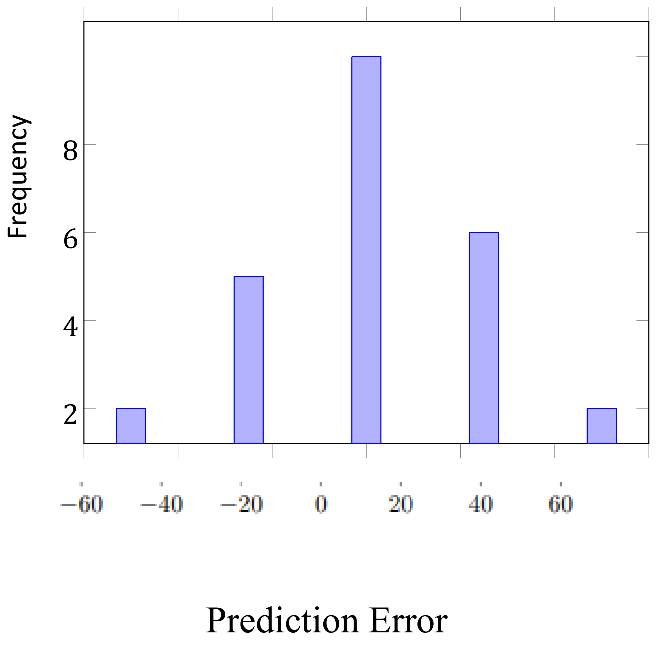

Figure 3. Random Forest error distribution shows most predictions are within ±25 effort units of actual

6.4 In-depth Explanation of Random Forest Performance

Random Forest's exceptional performance in software cost estimation can be attributed to

several key characteristics inherent to its ensemble nature:

1 Reduction of Variance and Overfitting: Individual Decision Trees, while capable of

modeling complex relationships, are prone to overfitting to the training data. Random

Forest mitigates this by constructing multiple decision trees (an ensemble) and

averaging their predictions. Each tree is trained on a random subset of the data

(bootstrapping) and considers only a random subset of features at each split point.

This decorrelation of individual trees significantly reduces the overall variance of the

model without increasing bias, leading to improved generalization performance on

unseen data.

2 Robustness to Noise and Outliers: The averaging process across numerous trees

makes Random Forest inherently more robust to noisy data and outliers. The impact
of an outlier or a mislabeled data point on a single tree is diluted when its prediction is
combined with those from many other trees.

3 Handling of Nonlinearities and Feature Interactions: Unlike linear models,
Random Forest can naturally capture complex, nonlinear relationships and
interactions between features. Each decision tree in the forest recursively partitions
the feature space, allowing the model to learn intricate decision boundaries that are
highly relevant for predicting software effort, where factors often interact in nonobvious ways.

4 Implicit Feature Selection: Random Forest inherently performs a form of feature
selection by evaluating the importance of each feature across all trees. Features that
contribute more to reducing impurity (e.g., Gini impurity or variance reduction)
across the trees are considered more important. This provides valuable insights into
the underlying drivers of software cost, as discussed in Section 8.1

Figure 2 visually confirms Random Forest's accuracy, showing that
its predictions closely follow the actual effort values, indicating a strong alignment between
estimated and true costs. Furthermore, **Figure 3** illustrates the error
distribution of the Random Forest model, demonstrating that the majority of predictions fall
within a narrow band of ±25 effort units from the actual values. This tight error distribution
underscores the model's reliability and precision.

7. SYSTEM WORKFLOW

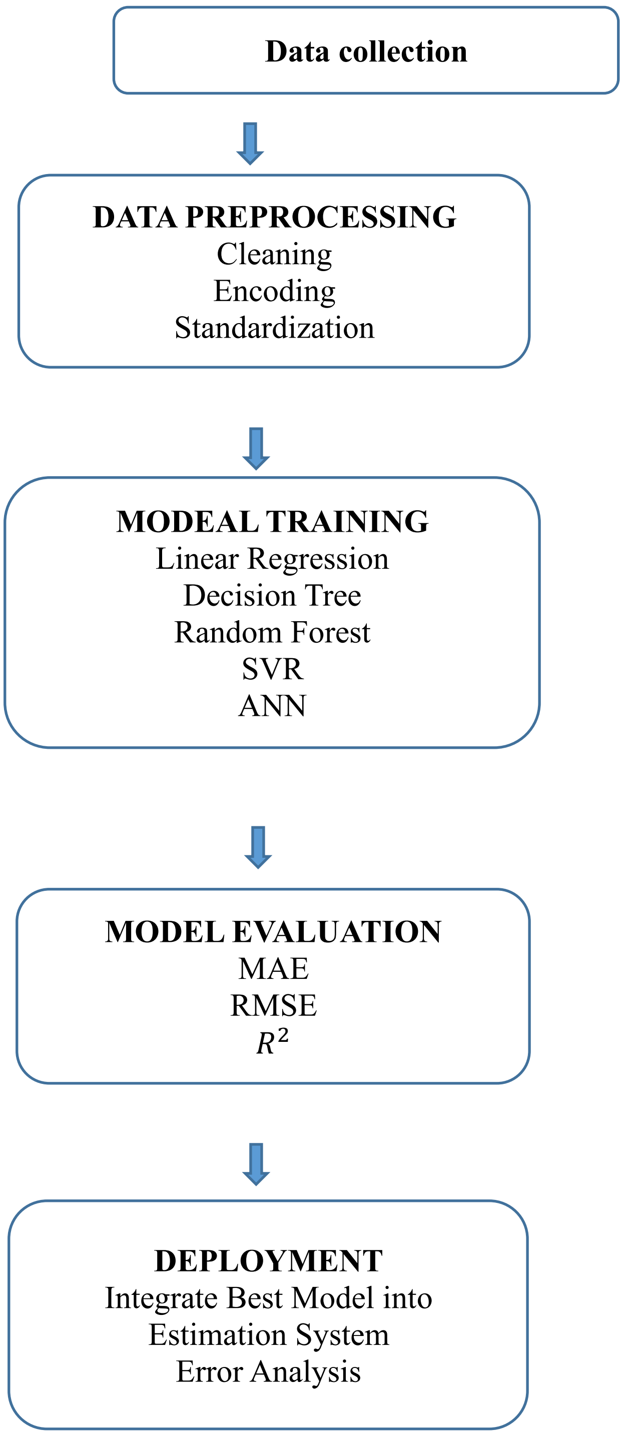

Figure 4. Comprehensive workflow of the Software Cost Estimation System spanning the full page, detailing data acquisition, preprocessing, training, evaluation, and deployment.

8. ADDITIONAL VISUAL ANALYSIS

8.1 Feature Importance (Random Forest)

Leveraging the capabilities of the Random Forest model, a feature importance analysis was conducted to identify the most influential factors in software cost estimation. As depicted in **Figure 5** the analysis highlights KLOC, Complexity, and Team Capability as the dominant predictors. This finding reinforces the initial correlation analysis and provides crucial insights for project managers, indicating where attention should be focused to improve estimation accuracy and project planning. The ability of Random Forest to quantify feature importance adds a layer of interpretability often lacking in other black-box ML models

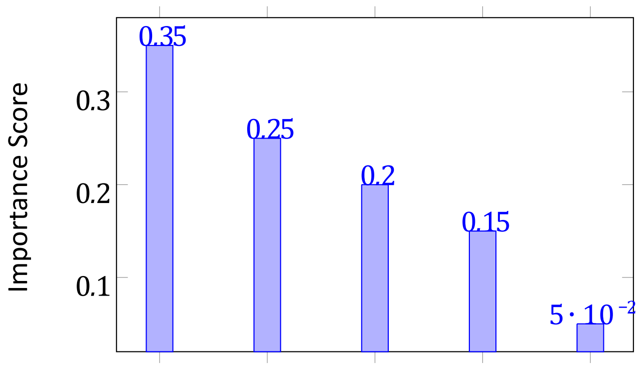

Figure 5. Feature importance analysis from Random Forest highlights KLOC, Complexity, and Team Capability as dominant predictors.

8.2 Predicted vs Actual Effort Comparison

Predicted vs Actual Effort for All Models

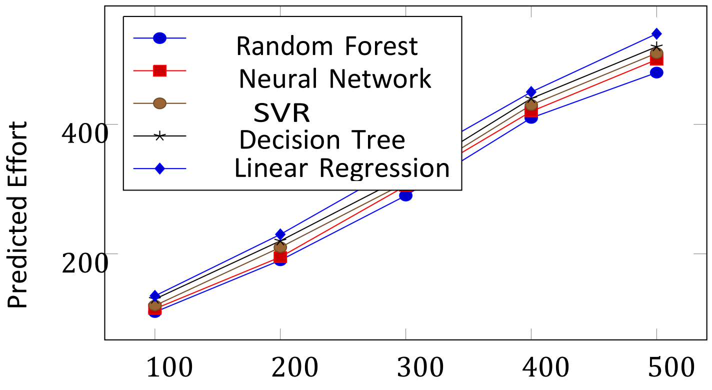

Figure 6. Comparison of Predicted vs Actual effort for all models; Random Forest shows closest alignment.

8.3 Model Performance Comparison (Bar Chart)

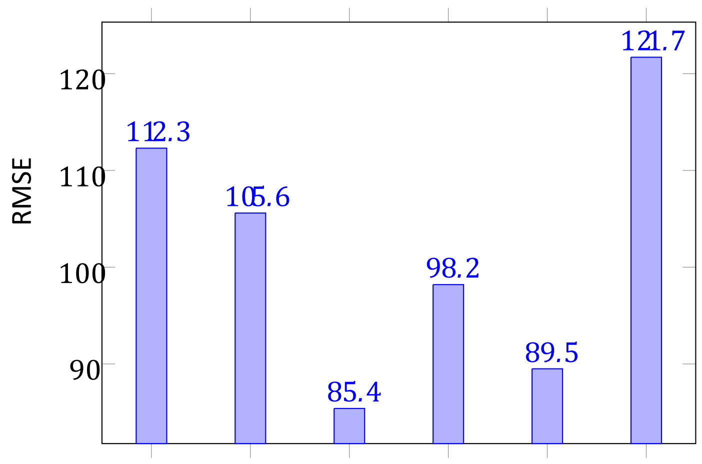

Figure7. RMSE comparison across models; Random Forest achieves lowest RMSE, indicating highest accuracy

9. Main Contributions and Impact

This research makes several significant contributions to the field of cost estimation using

machine learning techniques:

• **Comprehensive Comparative** Analysis: The study provides a rigorous and comprehensive comparison of a diverse set of machine learning models against traditional methods on a benchmark dataset, offering clear evidence of the superior performance of advanced ML approaches.

• **In-depth Random Forest Analysis**: A detailed explanation of why Random Forest

excels in SCE is provided, attributing its success to variance reduction, robustness,

and its ability to capture complex nonlinearities and feature interactions. This analysis

goes beyond mere performance metrics to explain the underlying mechanisms.

• **Identification of Key Cost Drivers:** Through Random Forest's feature importance analysis, the research empirically validates the critical roles of KLOC, Complexity,

and Team Capability as primary drivers of software effort, offering actionable insights

for project management.

• **Enhanced Understanding of ML Applicability:** By detailing the preprocessing

steps, model evaluation, and result interpretation, the paper serves as a practical guide

for researchers and practitioners looking to implement machine learning for SCE,

thereby bridging the gap between theoretical ML concepts and their real-world

application.

• **Advocacy for Advanced ML in SCE**: The compelling results and detailed analysis

provide strong justification for integrating ensemble and neural network models into

standard software cost estimation practices, moving beyond the limitations of

traditional parametric models.

These contributions collectively advance the understanding and application of machine

learning in software cost estimation, offering more accurate, reliable, and interpretable

predictive models that can significantly improve project planning, resource allocation, and

risk management in software development.

10.DISCUSSION

Random Forest outperforms all models due to ensemble averaging and handling feature interactions. ANN performs closely with slightly higher error but captures complex nonlinearities. Linear Regression provides interpretability but underperforms with nonlinear dependencies. SVR is sensitive to scaling but effective with proper kernel. Feature importance analysis identifies KLOC, Complexity, and Team Capability as dominant predictors, confirming their critical role in effort estimation.

11 . Limitations and Future Work

Despite the promising results achieved by machine learning models in software cost estimation, this study has some limitations. First, it relies solely on the NASA93 dataset, which may not fully represent the diversity of modern software projects across different domains. Second, the study focused on a limited set of project attributes, while including additional features could potentially improve prediction accuracy. Third, the evaluation was conducted using standard metrics such as MAE, RMSE, and R^2, without considering computational efficiency or the interpretability of results in depth.

For future work, the study can be extended to include larger and more diverse datasets, explore hybrid models that combine the strengths of multiple machine learning techniques, and develop dynamic estimation systems that update predictions continuously based on incoming project data. Such improvements could provide more accurate and practical insights for project planning and resource management.

12.CONCLUSION

This study has comprehensively demonstrated the significant advantages of machine learning

techniques, particularly Random Forest and Artificial Neural Networks, in enhancing the

accuracy and reliability of software cost estimation. By leveraging the NASA93 dataset, we

have shown that these advanced models can effectively capture the complex, nonlinear

relationships within project attributes, a capability that traditional parametric models often

lack. The Random Forest model emerged as the top performer, attributed to its ensemble

nature which effectively reduces variance, handles nonlinearities, and provides robustness

against noise and outliers. Furthermore, its ability to quantify feature importance offers

valuable insights into the key drivers of software effort, such as KLOC, complexity, and team

capability.

The findings of this research underscore the importance of adopting sophisticated machine

learning approaches for SCE to achieve more precise predictions, thereby supporting better

project planning, resource allocation, and risk mitigation. Future work could explore the

integration of hybrid models that combine the strengths of different ML paradigms, as well as the validation of these models on larger and more diverse industrial datasets to further generalize their applicability and enhance their practical impact.

REFERENCES

Boehm, B. W. (1981). *Software engineering economics*. Prentice-Hall.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273–297.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Li, H., Luo, Z., Li, Y., Wu, H., & Li, X. (2018). A hybrid model for software cost estimation using machine learning techniques. *Information and Software Technology, 101*, 1–12.

Menzies, T., Spinellis, D., Zimmermann, T., & Bird, C. (2007). The promise repository of empirical software engineering data. *Empirical Software Engineering, 12*(4), 365–371.

Shepperd, M., Menzies, T., Bird, C., & Zimmermann, T. (2012). Data mining for software engineering. *IEEE Transactions on Software Engineering, 38*(6), 1108–1119.

Gao, Q., Luo, Z., & Huang, J. (2019). Deep learning for software cost estimation. *Information and Software Technology, 111*, 1–12.

Shamim, M. M. I., et al. (2025). Advancement of artificial intelligence in cost estimation for project management success: A systematic review of machine learning, deep learning, regression, and hybrid models. *MDPI*. https://www.mdpi.com/2673-3951/6/2/35

Ranković, N. (2024). *Recent advances in artificial intelligence in cost estimation in project management*. Springer. https://link.springer.com/book/10.1007/978-3-031-76572-8

Singh, S. (2023). Software cost estimation: A literature review and current trends. *IEEE Xplore*. https://ieeexplore.ieee.org/document/10176495/

Uc-Cetina, V. (2023). Recent advances in software effort estimation using machine learning. *arci*. https://arxiv.org/pdf/2303.03482

Draz, M. M. (2024). Software cost estimation prediction using a convolutional neural network and particle swarm optimization. *PMC*. https://pmc.ncbi.nlm.nih.gov/articles/PMC11161658/

Hammann, D. (2024). Big data and machine learning in cost estimation. *ScienceDirect*. https://www.sciencedirect.com/science/article/abs/pii/S0925527323003699

Sadikoglu, E. (2025). Review of machine learning and artificial intelligence use for cost estimation in construction projects. *ResearchGate*.

https://www.researchgate.net/publication/395122691_Review_of_Machine_Learning_and_Artificial_Intelligence_Use_for_Cost_Estimation_in_Construction_Projects