

Original Article

SOFTWARE DEVELOPMENT COST PREDICTION USING ENSEMBLE LEARNING TECHNIQUES

Mazin Mohammed Ismael ^{1,*}, Ibrahim Ahmed Saleh ², Firas Abdulrahman Yousif ¹, and Shamil Khadr Ramadhan ³

¹Department of Information Technology, College of Computing and Informatics, AL-Hamdaniya University, Nineveh, Iraq.

²Department of Software, College of Computer Science and Mathematics, Al-Mosul University, Nineveh, Iraq.

³Department of Medical Instrumentation Techniques, Northern Technical University, Nineveh, Iraq.

*Corresponding author; E-mail: mazinmohammed@uohamdaniya.edu.iq (Tel: +964-7510344636)

ABSTRACT

Received
July 28, 2026

Accepted
September 1, 2026

Published
October 20, 2026

Cost estimation of software is an important task and should be done to complete the software project successfully. Traditional approaches of software cost estimation suffer from subjectivity and additional cost. To overcome these limitations with the traditional methods, the field of Machine learning has been incorporated as a promising method for the problem of software cost estimation. This paper explores using ensemble learning to improve cost predictions on two public benchmark datasets, Albrecht and Maxwell. Sequential Forward Feature Selection (SFS) is applied to pick the most relevant features and simplify models. Four ensemble methods ("Bagging", "AdaBoost", "XGBoost", and "CatBoost") have been implemented and evaluated on the benchmark Albrecht and Maxwell datasets. To avoid overfitting and to assess the model properly, the 10-fold cross-validation strategy was used. The results indicate that model performance is dependent on the dataset. CatBoost performed best on the Albrecht dataset, achieving the highest R^2 score of 0.943, while AdaBoost achieved the best performance on the Maxwell dataset, with an R^2 score of 0.735. These findings demonstrate the effectiveness of ensemble learning methods in improving the predictive accuracy of software cost estimation models.

Keywords: AdaBoost, Bagging, CatBoost, Ensemble Learning, Software Cost Estimation.

1. INTRODUCTION

Accurate software cost estimation is an important part of project management with significant influence on the success of the project. These estimations have great effects on the project execution if they are estimating the accuracy, and project managers can make the prediction of the resources required to complete the activities under the certain time and specification (Dawson, 2026). In doing so, estimates are established at the beginning of the project, reducing the risks of failure, providing project managers a realistic view of resources needed, enabling the project team to plan their task sensibly and enabling customers to budget with confidence rather than blind optimism (Mahmood *et al.*, 2022; Singh & Kumar, 2023).

The consequences of inaccurate estimation are well documented and can often lead to severe outcomes. Overestimates waste money and time and leave teams idle; underestimates are far worse and cause schedule delays, unmet technical requirements and sometimes project failure altogether. Industry evidence highlights the scale of the problem according to the CHAOS Report (2015), 56% of software projects went over budget, 60% missed delivery deadlines, and 44% failed to accomplish their goals (Sakhrawi *et al.*, 2024). These numbers highlight a hard truth that poor estimates are a common occurrence rather than an exception. One practical way to counter this trend is to start with high-quality, reliable data, as data integrity forms the backbone of any credible estimation effort. For organizations, precise cost estimates are essential for budgeting and for effective resource allocation

throughout the software development lifecycle (Martínez-Aguilar *et al.*, 2025; Aljohani & Qureshi, 2017).

Over the past two decades, ML-based techniques have gained widespread adoption in software cost estimation because of their capacity to learn from past data and adjust to changing project attributes. Unlike traditional parametric and statistical models, machine learning approaches are more suitable to tackle dynamic project environments and the interactions of features. However, it has not been found that one learning model is superior to another across different data sets and circumstances. By combining several estimators, ensemble learning can overcome this shortcoming and create more robust and accurate predictions. In software cost estimation, ensemble methods have been shown to be useful for improving estimation performance by leveraging the advantages of multiple models (Suresh Kumar *et al.*, 2022).

The main goal of this study is to introduce an ensemble learning approach for software cost estimation that can enhance the prediction accuracy and stability from a dataset point of view. To select the most informative features and to keep the models simple, the Sequential Forward Feature Selection (SFS) is used. Four different ensemble methods (Bagging, AdaBoost, XGBoost, CatBoost) are implemented and evaluated on both Albrecht and Maxwell benchmark datasets. Models are calibrated using values determined empirically using literature to ensure an apples-to-apples comparison. The remaining part of this paper is organized as follows. In Section 2, the literature review of methodologies used in forecasting software cost estimation is presented. In Section 3, the proposed methodology with

Access this article online



DOI: <https://doi.org/10.25271/sjuoz.2026.14.4.1988>

Printed ISSN 2663-628X;
Electronic ISSN 2663-6298

Science Journal of University of Zakho
Vol. 14, No. 04, pp. 679-686, October-2026

This is an open access article under a CC BY-NC-SA 4.0 license
(<https://creativecommons.org/licenses/by-nc-sa/4.0/>)

description of the dataset, feature selection and ensemble modeling techniques are described. Section 4 presents the analysis of results. Finally, Section 5 summarizes the main points of conclusion arrived at throughout the study, giving some suggestions for future works.

2. LITERATURE REVIEW

Cost estimation is an essential task of software projects. Initial estimates can help with effective planning, minimize uncertainty and decrease the risk of delays or failure, while poor estimates can have an impact on the entire project. There are several studies on software cost estimation using the traditional models and more recent machine-learning models, summarized below.

Fadhil *et al.* (2020) presented a hybrid optimization approach named DolBat which utilized the dolphin swarm and bat algorithms to optimize the α and β parameters in the ("COCOMO II") model. For optimization problems with a small number of individuals but many evaluations of the objective function, the dolphin swarm algorithm is particularly suitable. This hybrid framework found optimal solutions in an efficient manner over the available solutions using the principles of echolocation and different search heuristics. Consequently, the DolBat model yielded minimized estimation errors, achieving Mean Magnitude of Relative Error (MMRE) values of 50.2757 and 14.576 for the "NASA-93" and "NASA-60" datasets, respectively, thereby outperforming previously deployed algorithms.

Rai *et al.* (2021) suggested a hybrid model based on the size of the team, which combines Support Vector Regression (SVR) with the famous Constructive Cost Model (COCOMO). They wanted to enhance predictive accuracy without discarding the traditional estimation model by combining data-driven learning with the traditional estimation model.

Van Hai *et al.* (2022) created a machine learning-based model called EEAC (Effort Estimation Applied to Clusters). They specifically worked on the analysis of the impact of data clustering on software development effort estimation and the possibility of estimating the prediction reliability by grouping similar projects.

Hammad (2023) used a stacking ensemble approach to software cost estimation with machine learning algorithms (ANN, K-star, SVM, RF). The most relevant attributes were identified by a feature selection technique and experiments were done with and without it. The findings indicated that the accuracy was better when feature selection was used, but the selection of suitable features took more time and was vulnerable to changes in data. The model was applied to 76 student software projects with Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) of 2.1801 and 4.0779, respectively.

Sakhrawi *et al.* (2024) examined the improvement of Cost Estimation precision by leveraging the ISBSG dataset to contrast Neu-

ral Networks (NN), Support Vector Regression (SVR), and Decision Trees (DT). One methodological improvement of this paper endeavor is the utilization of a Stacked Ensemble Model (SEM), referred to as StackSDCE, which integrates these base estimators to harness their aggregated predictive capabilities. The authors reported that while International Function Point Users Group (IFPUG) standards proved suitable for measuring development scope, the selection of the final estimator was paramount. Specifically, the Decision Tree model persistently surpassed others with a reduced (MAE = 0.07). Furthermore, the investigation underscores the significance of Grid Search (GS) parameter tuning, which elevated the model's R^2 score from 0.7 to 0.8, confirming that hyperparameter optimization is as important as model selection in achieving precise cost predictions.

Qassem and Saleh (2025) proposed a stacking method using the ISBSG dataset, where multiple base learners were integrated into a Random Forest meta-model. The results showed that the individual models like AdaBoost and Linear Regression and K-Nearest Neighbors (KNN) achieved below-average results (54%, 43% and 65% accuracy) respectively. However, the stacked ensemble demonstrated an enhanced forecasting accuracy of 98.% and an error rate of 0.09278 for (MMRE), demonstrating that a meta-model can effectively mitigate the errors of individual learners to provide highly reliable forecasts for software development costs.

3. PROPOSED METHODOLOGY

The suggested framework combines a number of consecutive steps to improve the accuracy of software cost estimation. In the first step, that data were collected by consulting two benchmark datasets, i.e. Albrecht and Maxwell. Second, the data were pre-processed as a way of guaranteeing data quality through filtering out noisy cases as well as dealing with missing values. Specifically, the mean imputation technique was used for replacing missing or NaN (Not a Number) values with the calculated mean of their respective features. However, there was no missing data point so they did not require imputation. Third, the Sequential Forward Selection (SFS) was used to detect the most relevant features and reduce the dimensions. Fourth, 10-fold cross-validation strategy was implemented to achieve a sound evaluation. Hence, the dataset was partitioned into ten different folds and each fold worked as a test set one time and other nine folds as the training set. Fifth, bagging, adaboost, XGBoost and CatBoost ensemble learning models were implemented and trained on the processed data. Finally, these models were evaluated for software cost estimation in order to determine which approach provides the highest predictive accuracy. The parameter settings of the models are set to a consistent and fair comparative analysis through empirically determined parameter settings. Fig. 1 shows the overall architecture of the suggested approach.

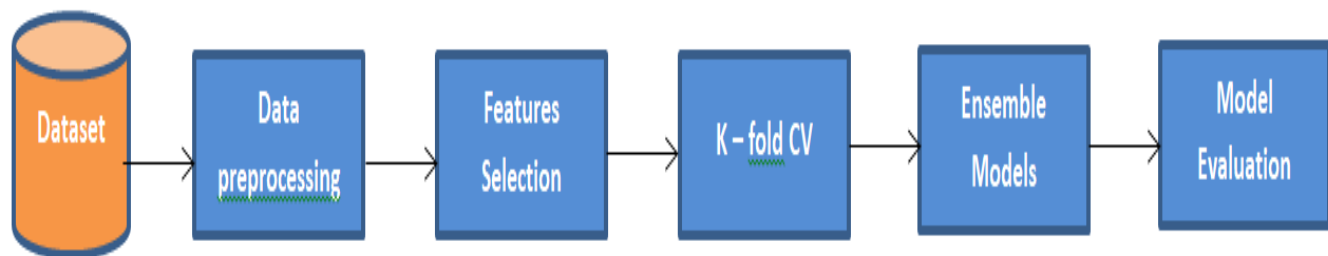


Fig (1): Proposed Methodology to Software Cost Estimation

Stage 4 (K-Fold Cross-Validation): Fig. 2 shows an overview of K-fold CV. K-fold CV is an approach for assessing Machine Learning Models (MLM). In K-fold CV, the dataset is divided into k folds. Each iteration is then split into one fold as testing data and the rest of the folds as training data (Zhang & Liu, 2023; Gorriz *et al.*, 2026). Therefore, it is repeated until the entire dataset is analyzed. K-Fold

CV results are reiterated to give the average score of the values of the MLM. To identify the most effective hyperparameter of the model, a variety of k values [2, 3, 5, 7 and 10] was tested in this study. The experimental results indicated k=10 had the highest accuracy in regression, which was better than the rest of the values tried.

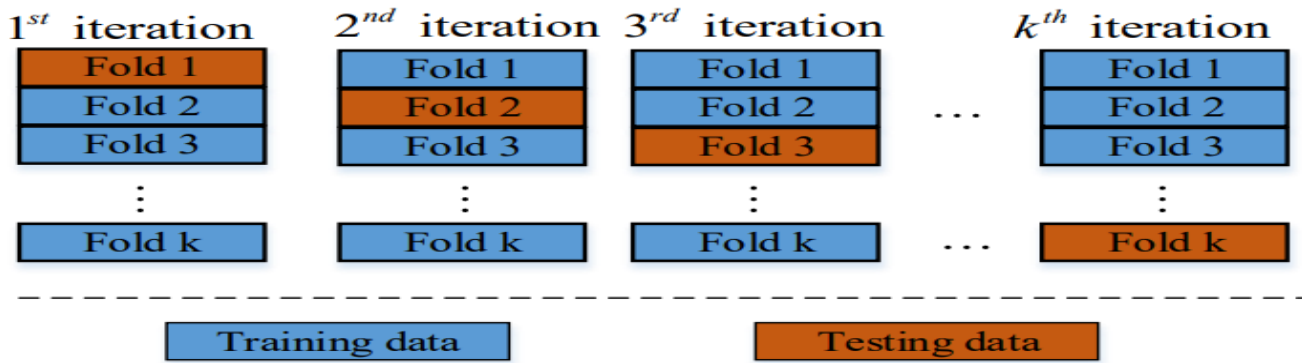


Fig (2): k – fold CV.

Datasets

The datasets that are employed in this paper give crucial information on software cost estimation, which covers different aspects of software projects. They contain various features of project characteristics, which facilitate efficient predictive modeling. The difference in the size of the datasets allows the machine learning models to be trained and tested effectively, which increases their generalizability to various project sizes. In general, these datasets are essential in enhancing the accuracy of prediction and data-driven decision-making.

Albrecht

In this paper, the Albrecht dataset is employed to predict the cost of software. The dataset includes 24 projects that were developed in the 1980s and 2000s. It has eight attributes, i.e., "Input, Output, Inquiry, File, FPAdj, RawFPcounts, AdjFP, and Effort". The initial seven attributes are software size and complexity metrics based on Function Point analysis and Effort is used as the dependent variable in predicting software cost (Tayyab *et al.*, 2016), as illustrated in Fig. 3.

Input	Output	Inquiry	File	FPAdj	RawFPcou	AdjFP	Effort
25	150	75	80	1	1750	1750	102.4
193	98	70	38	1	1902	1902	105.2
70	27	0	12	0.8	535	428	11.1
40	60	20	12	1.15	660	759	21.1
10	69	1	9	0.9	478.89	431	28.8
13	19	0	23	0.75	377.33	283	10
34	14	0	5	0.8	256.25	205	8
17	17	15	5	1.1	262.73	289	4.9
45	64	14	16	0.95	715.79	680	12.9
40	60	20	15	1.15	690.43	794	19
41	27	29	5	1.1	465.45	512	10.8
33	17	8	5	0.75	298.67	224	2.9
28	41	16	11	0.85	490.59	417	7.5
43	40	20	35	0.85	802.35	682	12
7	12	13	8	0.95	220	209	4.1
28	38	24	9	1.05	487.62	512	15.8
42	57	12	5	1.1	550.91	608	18.3
27	20	24	6	1.1	363.64	400	8.9
48	66	13	50	1.15	1073.91	1235	38.1
69	112	21	39	1.2	1310	1572	61.2
25	28	4	22	1.05	476.19	500	3.6
61	68	0	11	1	694	694	11.8
15	15	6	3	1.05	189.52	199	0.5
12	15	0	15	0.95	273.68	260	6.1

Fig (3): Albrecht Dataset

Maxwell

The Maxwell dataset comprises data collected from 52 projects created during the 1990 and the early 2000, with data on project attributes including "application domain, programming language, development environment, source lines of code", among other contributing variables (Tayyab *et al.*, 2016; Meharunnisa *et al.*, 2023). This dataset encompasses twenty-seven attributes, as detailed in Table (1).

Table 1: Summary of data sets used in software cost estimation

Dataset name	Total features	Language generations	Total projects	Missing Values	Project Years
Albrecht	8	1G, 2G, 3G	24	None	1980–2000
Maxwell	27	1G, 2G, 3G	52	None	1990–2000

Feature Selection

Sequential Forward Selection (SFS) is a "bottom-up" search method that starts with no features and progressively builds an optimal subset. In every iteration, one feature is picked out of the remaining candidate features using a pre-defined evaluation criterion aimed at minimizing Mean Square Error (MSE). The feature selected at each step is the one that contributes the most substantial improvement in model performance when added to the existing feature subset. This is repeated by considering all the non-selected features and adding a new feature to the feature set only when the addition of a new feature leads to a lower prediction or classification error than the other candidate. Pseudo code for the SFS procedure is detailed in Fig. 4 (Marcano-Cedeño *et al.*, 2010; Saha *et al.*, 2020).

Algorithm 1 Pseudo Code for Sequential Forward Feature Selection (SFS) algorithm

- 1: Procedure Sequential Forward Selection (SFS)
- 2: Start with an empty set $y_0 = \{0\}$
- 3: Choose the next best features
- 4: $X^+ = \arg_{x \in y_k} \max_j (y_k + x)$
- 5: Update set
- 6: $X^+ = \arg_{x \in y_k} \max_j (y_k + x)s$
- 7: Return to Step 3
- 4: End Procedure

Fig (4): Pseudocode for (SFS) Algorithm

Algorithms

This study employs multiple ensemble learning techniques, varying in how they process data, to build a more dependable estimation framework. The following sections provide a detailed technical overview of these algorithms, specifically Bagging, AdaBoost, XGBoost, and CatBoost, highlighting their individual mechanisms and their respective role in enhancing predictive accuracy.

Bootstrap Aggregating (Bagging)

Bootstrap Aggregation (Bagging) is an ensemble learning method proposed by Breiman (1996a) that can be effectively applied to both classification and regression tasks. It aims to reduce the variance of predictions and improve overall predictive performance. The core concept is quite straightforward: many samples of the data set are drawn from the original set, and a predictive model is built independently for each sample. Finally, the outputs of all models are combined together (average mean of the outputs in regression; highest voting in classification). This aggregation process can lower the variance of the models and make the prediction stable (Ngo *et al.*, 2022; Galar *et al.*, 2011). The bagging algorithm consists of the steps shown in Fig. 5:

Algorithm 1 Bagging

Input: training data $S = (x_1, y_1), \dots, (x_n, y_2), \dots, (x_n, y_n)$
 Base ML algorithm L
 The number of base learners T .
Procedure:
for $t = 1, \dots, T$:
 1) Generate a bootstrap sample S_j from the input data S
 2) Fit a base learner h_j using S_j , i.e. $h_j = L(S_j)$
end for
Output: Combine the outputs of the base learners, $H(x) = \text{mode}(h_1(x), \dots, h_T(x))$

Fig (5): Algorithm for Bagging

AdaBoost

Adaptive Boosting ("AdaBoost") is an ensemble learning framework grounded in the construction of multiple weak predictors whose outputs are combined through a weighted voting mechanism. In contrast to bagging, which independently generates base learners using bootstrap replicas of the training dataset, AdaBoost follows a sequential learning paradigm. At the outset, all instances in the learning set are assigned equal sampling probabilities (Wang, 2012). During each iteration, a new training subset is formed by sampling from the original dataset with replacement according to the current probability distribution. Once a classifier is trained on this resampled dataset, the sampling weights are updated based on the classification performance achieved up to that stage. Specifically, instances that have been misclassified in earlier iterations are assigned higher weights, thereby increasing their likelihood of being selected in subsequent training subsets (Mienye & Sun, 2022). The procedural steps of the bagging algorithm are outlined as follows:

Algorithm 2 AdaBoost

Input: Training set $S = \{x_i, y_i\}$, $i = 1, \dots, N$; and $y_i \in \{-1, +1\}$; T : Number of iterations; I : Weak learner
Output: Boosted classifier: $H(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x)\right)$ where h_t, α_t are the induced classifiers (with $h_t(x) \in \{-1, 1\}$) and their assigned weights, respectively
 1: $D_1(i) \leftarrow 1/N$ for $i = 1, \dots, N$
 2: **for** $t = 1$ to T **do**
 3: $h_t \leftarrow I(S, D_t)$
 4: $\varepsilon_t \leftarrow \sum_{i: y_i \neq h_t(x_i)} D_t(i)$
 5: **if** $\varepsilon_t > 0.5$ **then**
 6: $T \leftarrow t - 1$
 7: **return**
 8: **end if**
 9: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
 10: $D_{t+1}(i) = D_t(i) \cdot e^{(-\alpha_t h_t(x_i) y_i)}$ for $i = 1, \dots, N$
 11: Normalize D_{t+1} to be a proper distribution
 12: **end for**

Fig (6) : Algorithm for AdaBoost

XGBoost

Extreme Gradient Boosting ("XGBoost") is a scalable and optimized implementation of gradient boosting algorithms. It was developed by Tianqi Chen from the University of Washington and is based on the Gradient Boosting Decision Tree (GBDT) framework. The basic principle of GBDT is to build the decision trees one by one, each one aiming to correct the errors of the previous ensemble by fitting the negative gradients of a given loss function. This way, the model is progressively improved by minimizing the overall prediction error. Usually, the number of trees is predetermined, and the early stopping is done using the validation performance. In prediction, the final prediction is derived from the sum of all the trees in the ensemble (Li & Zhang, 2020; Airlangga, 2024).

Algorithm 3 Gradient Boosting

Input: training data $S = (x_1, y_1), \dots, (x_2, y_2), \dots, (x_m, y_m)$
 A differential loss function $L(y, F(x))$.
 The number of iterations T .
Procedure:
 1) Initialize the model with a constant value using
 $F_0 = \text{argmin}_{\alpha} \sum_{i=1}^N L(y_i, \alpha)$
 2) **for** $m = 1, \dots, M$:
 (i) Calculate the false residuals
 $r_{mi} = [-\frac{\delta L(y_i, F(x))}{\delta F(x)}]_{F(x)=F_{m-1}(x)}$ for $i = 1, \dots, n$ (ii)
 Train a base learner using the training set
 $D = x_i, r_{mi}$
 (iii) Obtain ρ_m by performing the line search optimization:
 $(\rho_m h_m(x)) = \text{argmin}_{\rho, h} \sum_{i=1}^N L(y_i, F_{m-1}(x_i) + \rho h(x_i))$
 (iv) Update the model:
 $F_m(x) = F_{m-1}(x) + \rho_m h_m(x)$
end for
Output: Return the final model $F_m(x)$.

Fig (7) : Algorithm for Gradient Boosting

CatBoost

CatBoost is an advanced gradient-boosted decision tree algorithm optimized for handling tabular datasets without requiring extensive manual preprocessing. Operating sequentially, it iteratively constructs symmetric, oblivious decision trees to minimize a loss function by fitting each new learner to the residuals of the previous ensemble. Unlike alternative frameworks like XGBoost, CatBoost introduces Ordered Boosting and Ordered Target Statistics, which utilize random data permutations to process categorical features while preventing target leakage and overfitting. Its symmetric tree structure provides key advantages, including high out-of-the-box accuracy, rapid inference speed, and robust generalization on noisy data. Consequently, CatBoost is widely applied in machine learning tasks such as fraud detection, recommendation engines, and complex predictive analytics (Hancock & Khoshgoftaar, 2020; Ibrahim *et al.*, 2020).

The detailed formula of the CatBoost model is as follows:

$$y_{meta} = \text{CatBoost}(y_{prophet}, y_{XGB}) \quad (1)$$

Where y_{meta} is the final prediction from the meta-model.

The Cat Boost Regressor tries to learn a function $\tilde{y}_{meta}$ that minimizes the loss function 1:

$$\tilde{y}_{meta} = \text{CatBoost}(y_{prophet}, y_{XGB}) = \arg \min \sum_{i=1}^N l(y_i, \tilde{y}_i) \quad (2)$$

Where $l(y_i, \tilde{y}_i)$ is the loss function (RMSE).

Evaluation Metrics

The performance of the proposed model was assessed using the following evaluation metrics (Gupta *et al.*, 2026; Rahman *et al.*, 2023):

- R-Squared

This metric is commonly used to assess model performance and is computed using Equation (3). Its score lies within the range of (0) to (1), where a score closer to 0 indicates limited explanatory capability, whereas a score closer to 1 reflects a stronger ability to capture the variance in the data.

$$R^2 = 1 - \frac{\sum_{i=1}^n (|y_i - \tilde{y}_i|)^2}{\sum_{i=1}^n (|y_i - \bar{y}_i|)^2} \quad (3)$$

Where (y_i) denotes the real effort, ($\tilde{y}_i$) refers to the expected effort, and ($\bar{y}_i$) denotes the average real effort.

- Mean Absolute Error (MAE)

This metric measures the average magnitude of prediction errors without considering their direction. It is computed as the mean of the absolute differences between the actual and predicted values, as shown in Equation (4).

$$MAE = \frac{1}{N} \sum_{i=1}^N y_i - \tilde{y}_i \quad (4)$$

Where (N) refers to the total number of data, (y_i) refers to the actual effort, and ($\tilde{y}_i$) denotes predicted effort.

- Root Mean Square Error (RMSE)

This metric is calculated by finding the residual (the difference between the actual and predicted values) for each data point, squaring it, taking the mean of these squared values, and then taking the square root. RMSE is represented in equation (5).

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \tilde{y}_i)^2} \quad (5)$$

Where (n) denotes the number of observations, (y_i) represents real values, and ($\tilde{y}_i$) denotes predicted values.

4. EXPERIMENTAL SETUP AND RESULTS

This section presents the experimental setup, evaluation metrics, and results obtained from applying the ensemble learning methods to the Albrecht and Maxwell datasets for software cost estimation.

- The process of feature selection was used before the model training to identify the most significant attributes to use in cost estimation. In the case of the Albrecht dataset, 6 features were chosen among 8, and in the case of the Maxwell dataset, 20 features were chosen among 27.

- Instead of using a single train-test partition, 10-fold cross-validation was applied to ensure a comprehensive evaluation of the models. By repeatedly utilizing different subsets of the dataset for training and validation, this approach reduced the likelihood of overfitting and produced a more robust estimation of model generalization performance.
- Four ensemble learning models were applied to software cost estimation: Bagging, AdaBoost, XGBoost and CatBoost.
- The hyperparameters of all four ensemble learning models were manually adjusted based on experimental performance. Table 2 presents the optimized hyperparameter settings used in this study.

Table 2: Hyperparameter Settings of the Ensemble Learning Models

Model	Hyperparameter	Value
Bagging.	n_estimators. max_samples. max_features. Bootstrap.	20 0.5 1.0 True
AdaBoost.	n_estimators. learning_rate. Loss.	20 0.1 Linear
XGBoost.	n_estimators. learning_rate. max_depth. Subsample. colsample_bytree.	20 0.1 3 0.8 0.8
CatBoost.	Iterations. Learning Rate. Verbose.	20 0.1 False

The experimental evaluation was conducted on both the Albrecht and Maxwell datasets using three performance metrics: R^2 , MAE, and RMSE. Table 3 presents the results obtained.

Table 3: Results of Ensemble Models Based on R^2 , MAE, and RMSE Values on the Albrecht and Maxwell Datasets

Dataset	Model	R^2	MAE	RMSE
Albrecht	Bagging	0.873	7.350	12.678
	AdaBoost	0.939	5.530	8.768
	XGBoost	0.612	16.031	22.202
	CatBoost	0.943	5.831	8.494
Maxwell	Bagging	0.709	4726.972	8820.016
	AdaBoost	0.735	4619.145	8426.871
	XGBoost	0.691	5074.711	9092.849
	CatBoost	0.704	4971.168	8903.312

In Albrecht dataset, CatBoost achieved the highest R^2 value (0.943) while achieving lowest error values (MAE = 5.831, RMSE = 8.494) which means that it has a good predictive power. AdaBoost also presented good results with $R^2 = 0.939$. Comparatively, Bagging showed average results, whereas XGBoost model yielded lowest results, including $R^2 = 0.612$. However, the findings on the Maxwell dataset had a different trend. The AdaBoost Ensemble performed the most with $R^2 = 0.735$, and MAE = 4619.145, RMSE = 8426.871, which is a good predictive performance. Bagging ($R^2 = 0.709$) was second followed by CatBoost ($R^2 = 0.704$) and XGBoost ($R^2 = 0.691$). The result shows that CatBoost was the best-performing model on the Albrecht dataset, whereas AdaBoost achieved the best performance on the Maxwell data.

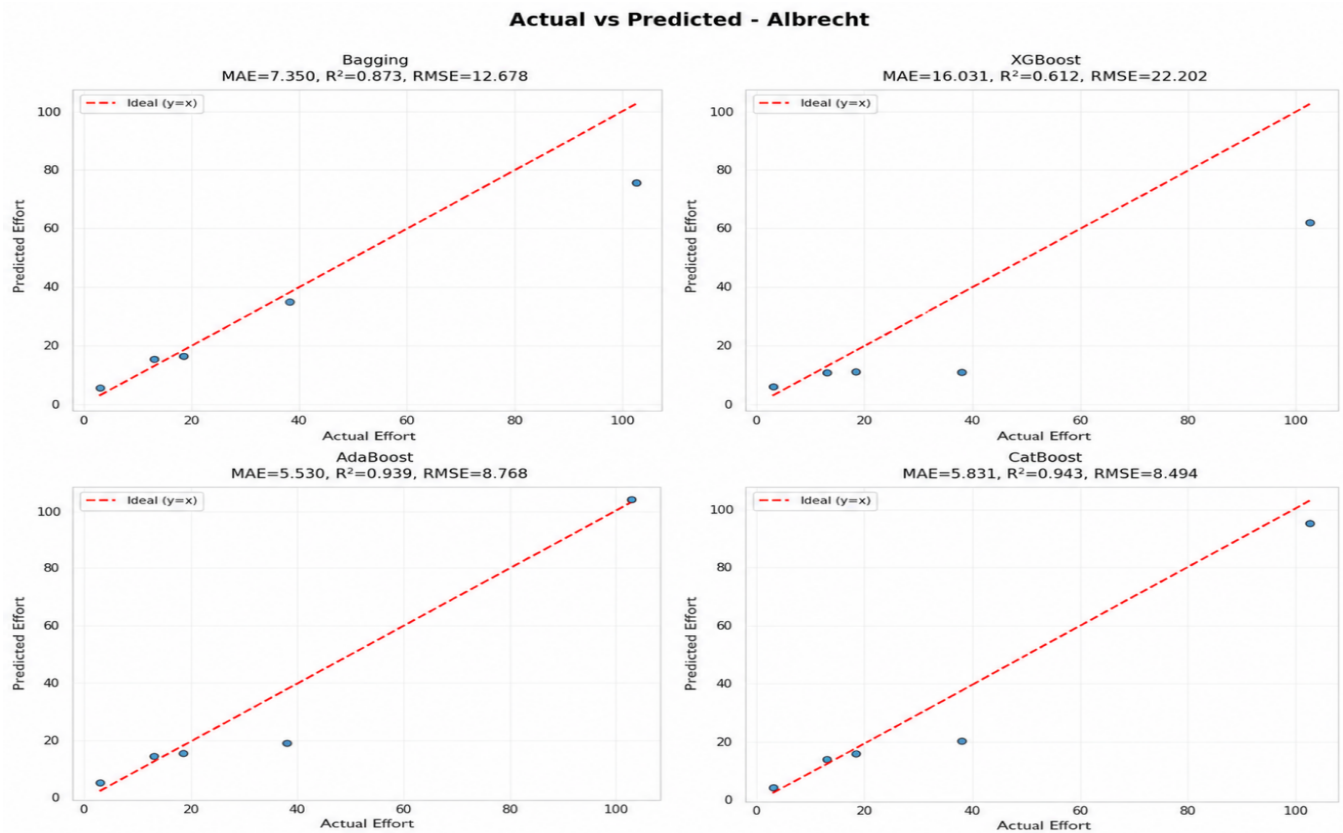


Fig (8) : Actual vs. Predicted Effort: A Comparative Analysis Using the Albrecht Dataset

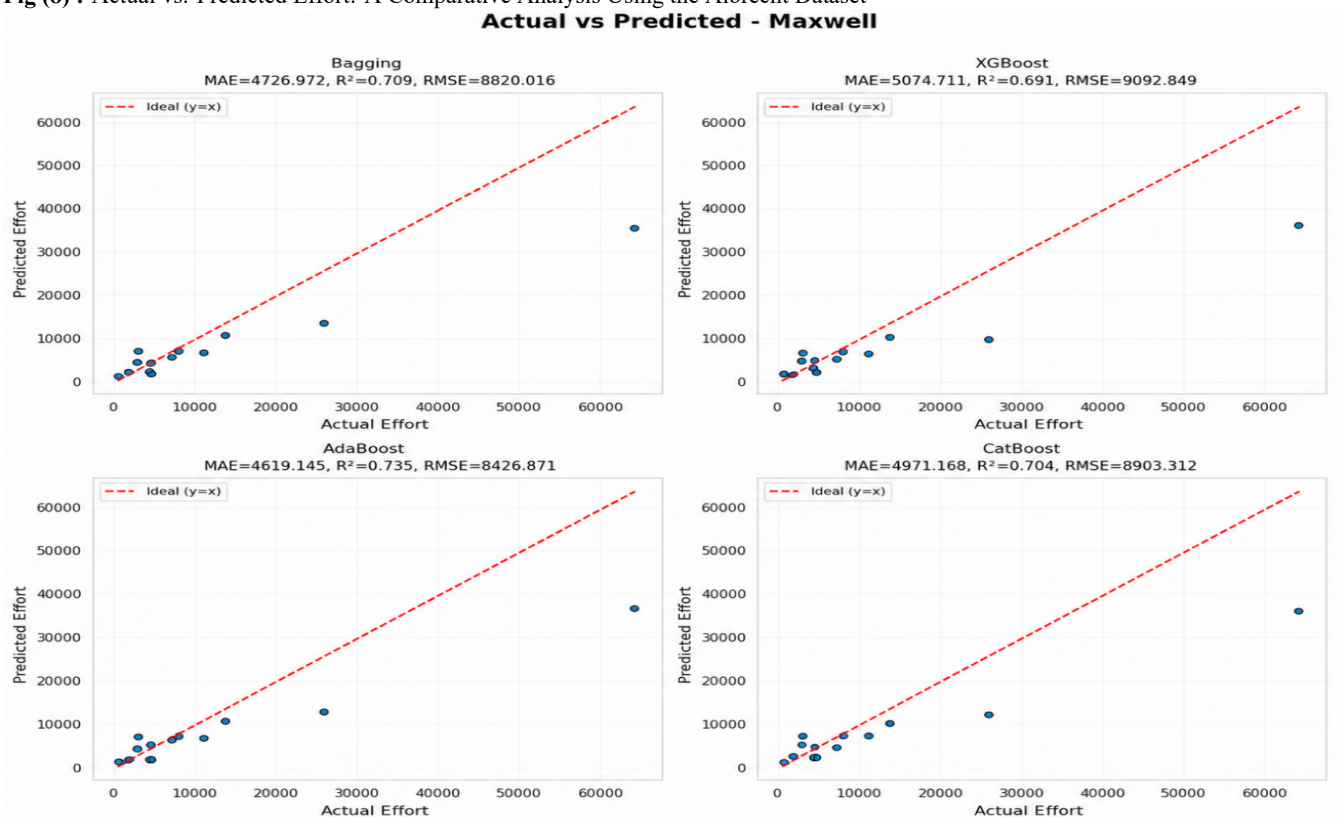


Fig (9): Actual vs. Predicted Effort: A Comparative Analysis Using the Maxwell Dataset

As depicted in Figures (8, 9), there is a comparative performance analysis of four ensemble learning methods, namely, Bagging, AdaBoost, XGBoost, and CatBoost, on the Albrecht and Maxwell datasets. The findings demonstrate that the predictive performance of the model is strongly influenced by the inherent characteristics and structure of the dataset. Although CatBoost performed best on the Albrecht dataset, AdaBoost performed better on the Maxwell dataset. The results demonstrate that no single ensemble technique consistently outperforms others across all datasets, as model effectiveness is strongly influenced by the underlying structure and complexity of the data.

CONCLUSION

This study investigated the effectiveness of four ensemble learning methods, namely, Bagging, AdaBoost, XGBoost and CatBoost, in improving the accuracy of software cost estimation. The adoption of a structured methodology, which included careful feature selection and 10-fold cross-validation, helped to guarantee a solid evaluation methodology, and reduce risks of overfitting and optimizing the use of the Albrecht and Maxwell datasets. The features selection achieved in our pre-processing phase proved effective, eliminating features to just 6 and 20 for each of our two datasets, making this predictive process more efficient.

The experimental results show that ensemble methods are very powerful tools to enhance the predictability, but they are limited to the data's particular characteristics. CatBoost outperformed all other models in the Albrecht dataset with a very high R^2 of 0.943 and the best error metrics. For the Maxwell data set, the AdaBoost ensemble showed the highest R^2 of 0.735, whereas the other approaches showed lower R^2 values. The results here stress one of the most important lessons to be learned from software engineering research, which is that there is no single algorithm for cost estimation. The ensemble model, on the contrary, should be selected in a way that the dynamics within the dataset are taken into account. This work offers a model-based, data-driven framework for researchers and practitioners to select the best planning strategies for software projects.

Acknowledgements:

The author would like to express sincere gratitude to the College of Computing and Informatics, University of AL-Hamdaniya, for providing academic support throughout this research. Appreciation is also extended to all researchers whose publicly available datasets and previous studies contributed to this work.

Author Contributions:

The authors conducted all parts of this article.

Ethical Approval:

This research did not require ethical approval because it did not involve humans or animals.

Funding:

None

5. REFERENCES

- Airlangga, G. (2024). Integrating LightGBM and XGBoost for software defect classification problem. *Jurnal Media Informatika Budidarma*, 8(1), 430–440.
- Aljohani, M., & Qureshi, R. (2017). Comparative Study of Software Estimation Techniques. *International Journal of Software Engineering & Applications*, 8(6), 39–53. DOI:<https://doi.org/10.5121/ijsea.2017.8603>
- Dawson, C. W. (2026). A neural network approach to software project effort estimation. *WIT Transactions on Information and Communication Technologies*, 16.
- Fadhil, A. A., Alsarraj, R. G. H., & Altaie, A. M. (2020). Software Cost Estimation Based on Dolphin Algorithm. *IEEE Access*, 8, 75279–75287. DOI:<https://doi.org/10.1109/ACCESS.2020.2988867>
- Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4), 463–484.
- Gorritz, J. M., Martin-Clemente, R., Segovia, F., Ramirez, J., Ortiz, A., & Suckling, J. (2026). Is K-fold cross validation the best model selection method for machine learning? *Information Fusion*, 135, 104404. DOI:<https://doi.org/10.1016/j.inffus.2026.104404>
- Gupta, D., Pandey, M., Kumar, A., & Suganthan, P. N. (Eds.). (2026). *Next-Generation Networks and Deployable Artificial Intelligence* (Vol. 1794). Springer Nature Switzerland. DOI:<https://doi.org/10.1007/978-3-032-15398-2>
- Hammad, M. (2023). Software cost estimation using stacked ensemble classifier and feature selection. *International Journal of Advanced Computer Science and Applications*, 14(6).
- Hancock, J. T., & Khoshgoftaar, T. M. (2020). CatBoost for big data: an interdisciplinary review: JT Hancock, TM Khoshgoftaar. *Journal of Big Data*, 7(1), 94.
- Ibrahim, A. A., Ridwan, R. L., Muhammed, M. M., Abdulaziz, R. O., & Saheed, G. A. (2020). Comparison of the CatBoost classifier with other machine learning methods. *International Journal of Advanced Computer Science and Applications*, 11(11), 738–748.
- Li, S., & Zhang, X. (2020). Research on orthopedic auxiliary classification and prediction model based on XGBoost algorithm. *Neural Computing and Applications*, 32(7), 1971–1979.
- Mahmood, Y., Kama, N., Azmi, A., Khan, A. S., & Ali, M. (2022). Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation. *Software: Practice and Experience*, 52(1), 39–65. DOI:<https://doi.org/10.1002/spe.3009>
- Marcano-Cedeño, A., Quintanilla-Domínguez, J., Cortina-Januchs, M. G., & Andina, D. (2010). Feature selection using sequential forward selection and classification applying artificial metaplasticity neural network. *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*, 2845–2850.
- Martínez-Aguilar, S., Sánchez-García, Á. J., López-Martín, C., & Octavio Ocharán-Hernández, J. (2025). Systematic Literature Review on Effort Estimation by Software Development Life Cycle Phases. *IEEE Access*, 13, 153340–153358. DOI:<https://doi.org/10.1109/ACCESS.2025.3605102>
- Meharunnisa, M. S., Abid, M., Awais, M., & Stevic, Z. (2023). Analysis of software effort estimation by machine learning techniques. *Ing. Syst. Inf.*, 28(6), 1445–1457.
- Mienye, I. D., & Sun, Y. (2022). A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *IEEE Access*, 10, 99129–99149.
- Ngo, G., Beard, R., & Chandra, R. (2022). Evolutionary bagging for ensemble learning. *Neurocomputing*, 510, 1–14.
- Qassem, N. T., & Saleh, I. A. (2025). Prediction of Software Cost Estimation Using Stacking Ensemble Learning method. *Procedia*

Computer Science, 258, 727–736.

- Rahman, M., Roy, P. P., Ali, M., Gonçalves, T., & Sarwar, H. (2023). Software Effort Estimation using Machine Learning Technique. *International Journal of Advanced Computer Science and Applications*, 14(4). DOI:<https://doi.org/10.14569/IJACSA.2023.0140491>
- Rai, P., Verma, D. K., & Kumar, S. (2021). A hybrid model for prediction of software effort based on team size. *IET Software*, 15(6), 365–375.
- Saha, P., Patikar, S., & Neogy, S. (2020). A correlation-sequential forward selection based feature selection method for healthcare data analysis. *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*, 69–72.
- Sakhrawi, Z., Labidi, T., Sellami, A., & Bouassida, N. (2024). A stacking ensemble learning model for software development cost estimation. *International Journal of Computer Information Systems and Industrial Management Applications*, 16(3), 13.
- Singh, S., & Kumar, K. (2023). Software Cost Estimation: A Literature Review and Current Trends. *2023 Third International Conference on Secure Cyber Computing and Communication (ICSCCC)*, 469–474. DOI:<https://doi.org/10.1109/ICSCCC58608.2023.10176495>
- Suresh Kumar, P., Behera, H. S., Nayak, J., & Naik, B. (2022). A pragmatic ensemble learning approach for effective software effort estimation. *Innovations in Systems and Software Engineering*, 18(2), 283–299. DOI:<https://doi.org/10.1007/s11334-020-00379-y>
- Tayyab, M. R., Usman, M., & Ahmad, W. (2016). A machine learning based model for software cost estimation. *Proceedings of SAI Intelligent Systems Conference*, 402–414.
- Van Hai, V., Nhung, H. L. T. K., Prokopova, Z., Silhavy, R., & Silhavy, P. (2022). Toward Improving the Efficiency of Software Development Effort Estimation via Clustering Analysis. *IEEE Access*, 10, 83249–83264. DOI:<https://doi.org/10.1109/ACCESS.2022.3185393>
- Wang, R. (2012). AdaBoost for feature selection, classification and its relation with SVM, a review. *Physics Procedia*, 25, 800–807.
- Zhang, X., & Liu, C.-A. (2023). Model averaging prediction by K-fold cross-validation. *Journal of Econometrics*, 235(1), 280–301.