Al-Qadisiyah JournalFor Engineering Sciences 19 (2026) 188 – 194

Contents lists available at: http://qu.edu.iq

Al-Qadisiyah Journal for Engineering Sciences

Journal homepage: https://qjes.qu.edu.iq

Research Paper

Software cost estimation technique based on bagging ensemble learning algorithm

Nedaa T. Qassem and Ibrahim A. Saleh

Software Department, College of Computer & Math,University of Mosul, Iraq.

ARTICLE INFO

Article history:
Received 22 April 2024
Received in revised form 14 March 2025
Accepted 22 April 2026

keyword:
Ada boost
Decision tree
Gradient boosting
ISBSG dataset
Machine learning
Random Forest
Software cost estimation

ABSTRACT

Recently, software cost estimation has become a more important issue in the software project development cycle, software quality, and decision-making management. In view of the common problem of inaccurate and difficult cost estimation in the software industry, in this article, the proposed bagging method is one of the ensemble learning methods to estimate the cost of software development. Five algorithms were used as the basic models: Random Forest, Decision Tree, AdaBoost, K-Nearest Neighbor, and Gradient Boosting, compiled using the bagging method. The proposed method was applied to a data set (ISBSG). The contribution of the paper suggests a more accurate method compared to previous studies, and applying it to high-quality data, which was prepared to obtain more accurate results when applying the proposed model, which showed its superiority over individual models in estimating the cost of software development. The results showed high accuracy in R^2 prediction in ratio (97%) and gave a lower error rate (MMRE: Mean Magnitude of Relative Error) compared to previous studies in ratio (0.1). This indicates its accuracy in prediction is closer to the real cost, where the RF model was the basic estimator model in this method, because it surpasses the main models that were used in the proposed method. The KNN model gave the lowest accuracy ratio (73%) among the main models when trained on the ISBSG dataset.

1. Introduction

Information technology practitioners are often asked how much it costs to develop a website, how much it costs to develop a mobile application, and how much it costs to develop a system. To answer these questions, it is necessary to evaluate the cost of software development, the functional requirements, and the non-functional requirements of software, which are the main factors that determine the cost of software. Even different development teams have different budgets for software development with specific, clear needs [1]. The act of estimating software development costs using a collection of procedures or models is known as software cost estimation. It is used during the whole software development cycle, which includes system design, testing, requirements analysis, coding, implementation, and maintenance. Software cost estimates will affect the management of the entire project, leading to delayed project delivery, low-quality software projects, a waste of manpower and material resources, etc. Cost estimates are reasonable, beneficial to management and cost control of software projects, and can also standardize the software service outsourcing market and reduce malicious bidding [2]. Current software cost estimation methods are roughly divided into two categories: computational and non-computational model methods. Software cost estimation methods based on computational models are mainly based on IFPUG "International Function Point Users Group" function points and COCOMO model calculation methods. The primary function of this kind of estimating technique is to offer one or more algorithm models that develop software cost estimation into a procedure where a number of distinctive characteristics that influence software cost are computed using the algorithm model. Since most of the calculation models were proposed earlier, the software design and development process has changed significantly over the past few years, so estimation methods based on computational models make it difficult to accurately estimate current domestic software R&D costs. Software cost estimation methods other than computational models no longer use calculation formulas to estimate costs [3]. However, expert estimating methods, neural networks, analog estimation methods, or other machine learning estimation methods require a significant quantity of historical data to estimate the cost of the software project to be assessed. The typical parameters that influence software cost are entered into the machine learning estimate algorithm [1, 2]. Usually, in order to achieve rapid convergence and avoid local optimal solutions, domestic and foreign scholars use artificial bee colony, particle swarm, ant colony algorithm, whale optimization algorithm, Gray wolf optimization, and other algorithms to optimize neural networks [3–6]. When the amount of historical data is small, the neural network estimation method is prone to overfitting, which greatly affects the estimation results. The analogy estimation method is to predict the cost of a software project by comparing the software project to be estimated with the completed project. From a broad perspective, the expert estimation method is also an analogy method [7]. Compared with the neural network estimation method, the analogy method can still guarantee a certain accuracy of estimation even when the amount of historical data is insufficient. The existing analogy method mainly uses Euclidean distance as a standard to measure differences between projects. While ignoring the correlation between characteristic parameters that affect the cost. This paper proposes a bagging method to estimate the effort and cost of software development. It is one of the methods of ensemble learning that works to train a set of models and then collect their predictions to obtain a final prediction as an output for this method.

*Corresponding Author.
E-mail address: i.hadedi@uomosul.edu.iq ; Tel: (+964) 770-201 1124 (Ibrahim Saleh)

https://doi.org/10.30772/qjes.2024.147128.1138
2411-7773

Nomenclature

D	Distance between two points	MRE	Mean Relative error
E	actual effort	MSE	Mean square error
Ea	Represents the real effort	N	Represents the number of projects
Ep	represents the expected effort	$RMSE$	Root Mean Square Error
$\hat{e}$	each expected effort	RSS	Remaining Sum of Squares
TSS	Total Sum of Squares	X, Y	One of the points that are used to calculate the distance
MAE	Mean absolute error	Y	Actual values
$MMRE$	Mean Magnitude of Relative Error		

This method has been proposed to obtain better accuracy of prediction. The following models are adopted: Random Forest algorithms, Decision Tree, Ada boosting, K -nearest neighbor, gradient enhancement, as basic learners of the packing method, and the ISBSG dataset containing 10600 software projects was used to train the basic learners. It is a large data set that contributes to improved performance and increased accuracy, unlike historical data of small size, which affects prediction accuracy. The rest of the paper is divided into related work that focuses on reviewing software cost estimation, illustrated in Section 2. While Section 3 present methodology of the paper include dataset, preprocessing, and ML methods. Section 4 presents experimental results that are being compared and analyzed in detail.

1.1 Literature review and related work

The software cost evaluation method based on the analogy method is used to measure the difference between two software projects. These two types of distance are defined as the cumulative sum of the differences of multiple characteristic parameters that affect cost in two software projects. For each characteristic parameter that affects cost, characteristic parameters are usually weighted and accumulated to achieve more accurate estimation results. Each cost of the analogy method affects the weight of the characteristic parameters. Currently, there are mainly expert artificial intelligence and machine learning-based methods. The expert artificial assignment method is the accuracy of software cost estimation, which depends on the expert's personal ability level and their understanding of the software industry. The accuracy of results cannot be guaranteed in software cost estimation in different fields. The cost characteristic parameter weight estimation method based on machine learning uses heuristic algorithms and other methods that assign weights, usually have certain advantages in accuracy over expert manual assignment methods [8]. There are many approaches in the literature about software cost estimation using machine learning. Bhaskar M. (2019) suggested using the linear regression algorithm and the k-nearest neighbors applied algorithm to COCOMO81, COCOMONASA, COCOMONASA$_2$ data sets. Used accuracy metrics (RMSE, RAE, RRSE, MAE, Correlation Coefficient (25) to measure the performance of the model to estimate the cost and effort of software development. The results showed the superiority of the linear regression algorithm by means of the correlation metrics, which had a higher score, and the relative error scale had a low value compared to the k nearest neighbor algorithm [9]. ANFAL et al. (2020) proposed a model for a hybrid algorithm from the (Dolphin & Bat) algorithm that was used to estimate the cost of a COCOMO II model. The hybrid algorithm was applied to two sets of data (NASA-93 and NASA-60) using the measure of accuracy MMRE. The results when applying the (Dol-Bat) algorithm were better compared to other algorithms and thus contribute to improving the guessing process [10]. In the same year, researchers Nawaz, M. A., et al. Presented a research to estimate the cost using machine learning algorithms and their data set (COCOMO81), where they chose (ABC)&(MCS) algorithms and the ABC-MCS Hybrid algorithm. They modified the neural network using these algorithms, after which the cost of effort was estimated and the cost was assessed using the two evaluation scales (MARE, MMRE), the results were highly accurate in estimation [11]. Also in the same year Muhammad Arif, et al. with other searchers proposed a model BABE is a model consisting of two algorithms (Artificial Bee Colony) and (Analogy-Based Estimation) guess directed to measurement for a more accurate guess This resulted in an algorithm BABE measurement-based model of an artificial bee colony for a more accurate prediction (ABC) algorithm gives different weights , appropriate weight is taken and used in (ABE) The similarity function during the training phase The proposal was applied to six sets of data where results gave accuracy in the training using data set (ISBSG) [12]. Bilal Khan et al, (2020) made a proposal to use flower pollination algorithm (FPA) to estimate the cost of the software; the MMRE metric was used to measure proposed model. The proposed model was compared with COCOMO model. Experimental results showed that FPA algorithm performed better than COCOMO model. Three data sets from NASA software projects were used NASA93, NASA63, NASA60, where improvement results were better when using the NASA63 data, improvement rate was (77.38%) [13]. Carvalho and Halcyon et al. in (2021) compared Extreme Machine Learning (ELM) and other machine learning techniques in software work estimates using the Desharnais dataset. learning strategies, including (Multilayer Perceptron, KNN, Linear Regression, and SVR). To determine which features have the most influence on the project effort, they have employed the Pearson correlation coefficient. Five attributes were selected from the original dataset's twelve attributes. Before using machine learning techniques, all of the characteristics were standardized. It was determined that the ELM model outperforms the other approaches under investigation because of its faster learning rate and greater capacity for generalization [14]. A new improved optimization algorithm for (eDTO) was proposed in 2021 by (Shweta.KR) and others to reduce uncertainty in estimating cost of software projects, the performance measures were used (ACC,VAR ,BRE, MRE, MMRE), where proposed algorithm was compared with current algorithms such as COCOMO, neural network, and strawberry algorithm, where results showed that improved EDTO algorithm has high variance and a lower error rate, And high efficiency in performance to calculate effort and development time of NASA projects [15]. Khan, et al (2022) proposed hybrid optimization method (HACO-BA) which merging Ant colony optimization and Bat optimization algorithms for cost estimate .resulted in a hybrid optimization algorithm that supports solving the problems of many variables model to adjust its coefficients and improve training stage, as results showed that the hybrid (HACO-BA) gave better results in adjusting the parameters of and gave better performance in terms of improving terminology, accuracy and implementation time in improving (DNN) The accuracy in approach Deep Neural Network that they proposed reached (98%), either Neural Network when using same dataset gave a lower resolution (85%) [16]. In 2022, Rashid et al introduced a new model to estimate the software development effort. They used the COCOMO model and neural networks (ANN), where they used neural networks to improve the COCOMO model used (backpropagation feedforward) in the training of the neural network. The model was applied to three data models: NASA 93, COCOMO 81, and COCOMO SDR. The results showed that the proposed model gave better results compared to COCOMO models, BPNN, RBNN, and DANN. The model was evaluated using the metrics MMRE and MRE, where the ratio of metrics MMRE (0.002), while the value of MRE is less compared to the rest of the models and their results are better [17]. Marco et al. (2022) presented. a method for enhancing software development effort prediction. Applying machine learning techniques, they enhanced the hyperparameters, leading to enhanced prediction. Using Bayesian technology, random forest, and Adaboost, the hyperparameters for the following machine learning models (SVM, KNN, and MLP) were selected. Research had been performed on a variety of data sets, such as ISBSG data, and the outcomes improved the accuracy of prediction models that were more accurate than they were before the improvement. R2 metric was employed for comparing the outcomes, and the models' respective findings were (0.5687), (0.5687), and (0.522) [18].

2. Methodology

In this part of the paper described proposed method (Bagging Ensemble) to estimate cost which illustrate in Fig. 1. It is proposed to use an ensemble learning approach by means of a Bagging model, which consists of several learners; each learner has its own results, and then expectations are collected. The highest majority of the expectations are voted on. The final forecast is given, a ISBSG dataset was used, which contains 10,600 software projects from different countries, this data was adopted to diversify of its projects and containing a number of necessary characteristics in prediction process, learners were trained on this data, where data was divided into 20% test 80% training, algorithms Random Forest, Diction Tree, Ada boosting, K nearest neighbor Gradient boosting, were applied as separate learners in proposed model.

2.1 Data Set

The dataset used in this paper, ISBSG (International Software Benchmarking Standards Group) data set for 2021, second edition, was used from the ISBSG

repository, which is a repository that contains a large number of software project data collected from more than 26 countries. It contains 10,600 software projects and 252 properties for various medical, engineering, financial, and other fields, providing data for development, maintenance, and software support projects. Data obtained from the repository ISBSG through the link http://www.isbsg.org/. Table 1 shows the properties used in the search.

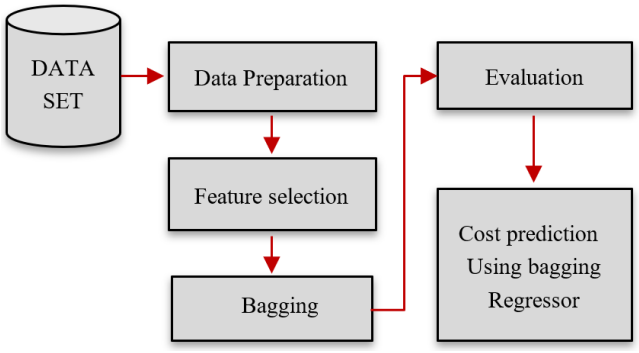


Figure 1. Proposed method (Bagging Ensemble).

Table 1. ISBSG Attributes Description.

No.	Attributes	Description
1	Data Quality Rating	Stands for Project Quality Rating to A ,B,C,D
2	UFP rating	Unadjusted function point Rating
3	Application Type	Specifies the type of application
4	Development Type	Type of development is described
5	Count Approach	Description of the technique used for Project size
6	Functional Size	Shows the number of unadjusted function points
7	Relative Size	Classification of functional size by relative size
8	Adjusted Function Points	Project Adjusted Functional Size Using Final Count
9	Normalized Level1 PDR (ufp)	Normalized Level1 Productivity Rate (unadjusted function points)
10	Normalized PDR (ufp)	Normalized Productivity Rate (unadjusted function points)
11	Pre 2002 PDR	Pre 2002 Productivity Delivery Rate
12	Speed of Delivery	Speed of Delivery Rate

2.2 Data pre-processing

After the data was collected accurately by advanced teams and according to warehouse standards, ISBSG. The second section manipulated the data set for different projects of maintenance, guessing, and development. The data is classified according to its quality into four categories (A, B, C, and D). Projects with class (A) are characterized by the fact that the data of their projects are sound, and there are no factors affecting their integrity. Class B refers to the integrity of the data, but there are factors that affect it, while in classes C and D, the data is not important and does not give credibility due to the invalidity of the data and the factors that affect it. The initial processing of data involves a number of steps:

- Step 1: In the first step, the projects with classification (A, B) were taken whose data are sound and not affected by factors affecting their credibility, and projects with classification (C, D) were deleted.
- Step 2: Identified projects with Unadjusted functional points, such as Projects with classification (A, B) with Unadjusted functional points were taken to have sound projects. In the Count Approach column, the projects that were scaled were taken using one of two methods: (IFPUG 4+) (International Function Points Users Group version 4 and greater) or NESMA (Netherlands Software Metrics Users Association) and neglect of projects that are measured in other ways.
- Step 3: Process the deleted data, rows, and columns containing more than 20% of missing data were deleted.
- Step 4: Deleted columns that had nothing to do with the cost estimation process and removed them by instructing drop. In this approach, is removed columns ('ISBSG Project ID', 'Year of Project', 'Resource Level', 'Implementation Date', 'Normalized Work Effort, 'Summary Work Effort', 'Effort Unrecorded') are removed.
- Step 5: Then, processed missing values using the (Most_Frequent) strategy [19] and using the simple Imputer command, so it fills the missing values with the most frequent values in the column.
- Step 6: Converting categorical values to numbers to deal with them and forming a new data frame with column names by instructing (Select_dtypes) determined new columns using(z-score) [20]. The method to determine the outliers far from the average in the data set If there are outliers is set (True) when the condition is $(z - score > 3)$, After the filtration process, 17 columns out of 130 columns and 2943 records out of 6760 records will remain. The goal is defined as (Normalized Work Effort Level 1), which is the value of (Y), and the remaining columns are values of (X). The following Fig. 2 shows the data before pre-processing, and Fig. 3 shows the data after pre-processing.

	A	B	C	D	E
1	ISBSG Project ID	Data Quality Rating	UFP rating	Year of Project	Industry Sector
2	10001	D	A	1998	Service Industry
3	10011	B	A	1996	Construction
4	10012	B	A	2002	Wholesale & Retail
5	10014	B	A	2004	
6	10015	B	A	2000	Wholesale & Retail
7	10026	B	A	2000	Insurance

Figure 2. Part of (ISBSG) dataset before pre-processing.

	A	B	C	D	E	F	G
1	Data Quality Rating	UFP rating	Application Type	Development Type	Functional Size	Relative Size	Count Approach
2	1	1	90	0	223	1	0
3	1	0	91	1	410	2	0
4	1	1	244	0	8	7	0
5	1	0	244	0	36	3	0
6	1	0	244	0	118	1	0

Figure 3. Part of (ISBSG) dataset after pre-processing.

2.3 Models

Bagging Ensemble Learning (Bootstrap Aggregating): It is one of the types of ensemble learning. This method works training a group of decision trees by taking part of the original training data randomly and training these trees, each tree is an independent model from the other tree after the end of training It works to collect expectations from the trees and take a final prediction using the method of voting on the majority with a similar expectation or find the average for these decisions [21]. The Bagging method was proposed to estimate the effort and cost of software development. Ensemble learning models proved superior to single models in the accuracy of guessing and making appropriate decisions to solve guessing problems. Figure 4 shows the Bagging method. The independent models used in the proposed method are presented below.

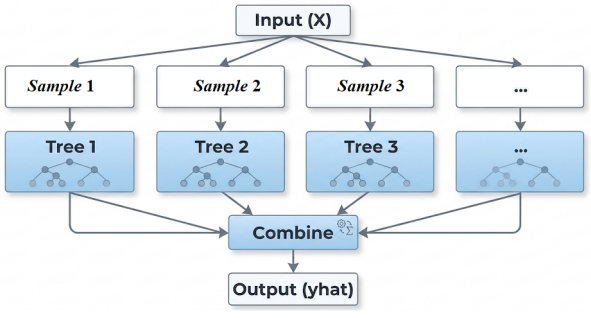


Figure 4. Bagging method [21].

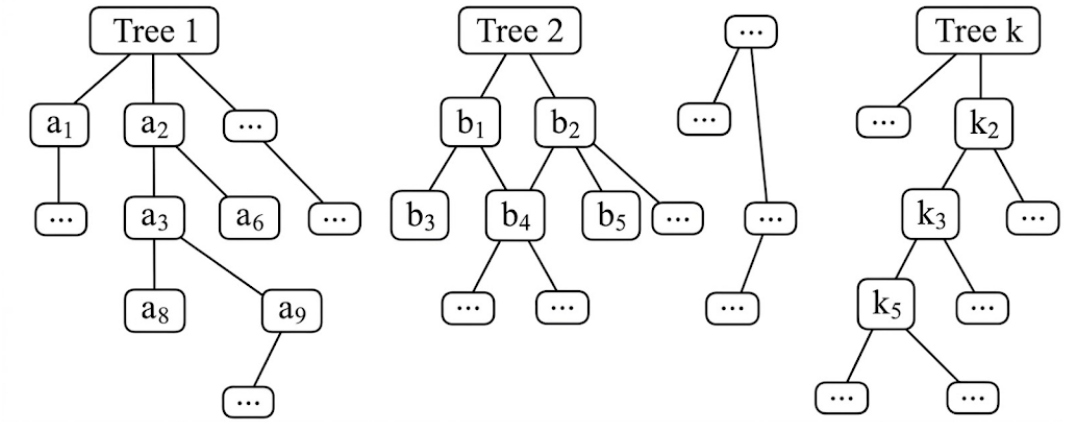


Figure 5. Random Forest Algorithm [22].

2.3.1 Diction tree

It is one of the supervised machine learning algorithms in the form of a tree whose structure consists of the first node, which is called the root, down to the end nodes. The condition is placed on the node in the event that the condition is fulfilled; it goes to the right of the node if it is not achieved; it goes to the right side; and so on to the end of the conditions and down to the terminal nodes. It can be described as hierarchical in shape and used to make decisions [23].

2.3.2 Random forest algorithm

It is one of the supervised ensemble learning algorithms widely used for classification or regression operations [24]. It consists of a group of decision trees in the form of multiple individual groups, where the algorithm calculates the average results from all trees and presents them as a final output [25]. It gives better performance than individual models because it is a grouping technique and possesses randomness, which works to reduce the variance of the model. It has become most widespread in the current decade. Due to its ease and accuracy of results [26], Fig. 5 shows the random forest algorithm [22].

2.3.3 Adaboosting

It is one of the methods of supervised machine learning, which is one of the methods of ensemble learning used to solve classification and regression problems. It works to train weak models by adjusting weights to improve their performance and make them strong models. Models with poor performance are given higher weights than models with good performance. The weights of training samples are adjusted, and basic models are trained on that data, thus improving the performance of the models to give good results. The process of training models is done sequentially: the first model is trained, then its expectations are taken and used in the next model. The process continues until reaching the last model and obtaining a final prediction. Figure 6 illustrates how Adaboosting works [27, 28].

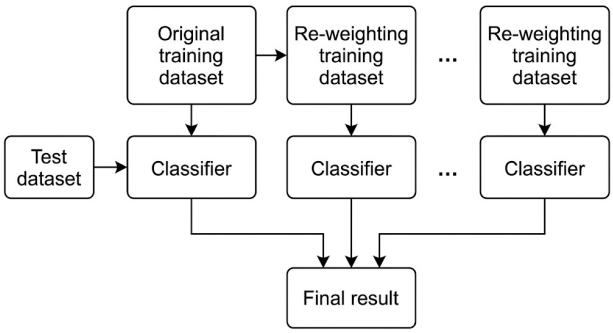


Figure 6. Adaboosting framework [28].

2.3.4 Gradient boosting

It is one of the supervised machine learning and ensemble learning techniques used for prediction or regression, and is the most widespread. It consists of a group of individual learners. All learners are given the same weight in the first step, and learners are trained on samples of training data, and then weights are measured. The learner who has a higher weight is a weak learner. Errors are identified, the learner is retrained, then the output enters the next learner, and the process continues sequentially to the last learner to get a strong learner with high performance and accurate results in classification or regression. Figure 7 illustrates how Gradient boosting works [28, 29].

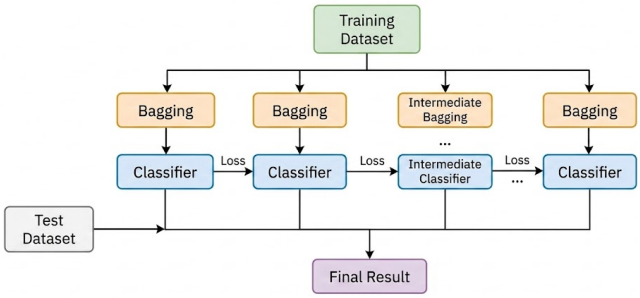


Figure 7. Gradient boosting framework [28].

2.3.5 K-Nearest Neighbor (KNN)

The K-nearest neighbour algorithm is one of the machine learning algorithms characterized by its simplicity of performance. Samples are classified based on the nearest neighbour; samples with similar characteristics are collected, and the test sample belongs to the part with the most votes. It works by taking elements adjacent to the input element from the data set, and the distance between two points is measured in the Euclidean distance, determining the parameters and data, and then training the algorithm is trained to obtain an accurate prediction. To calculate the Euclidean distance between two points of the following Eq. 1, [30, 31].

$$d(A+B) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1}$$

2.4 Evaluation Criteria

They are measures used to measure the performance of predictive models. The following will mention the measures used to measure the performance of the proposed method:

- R^2: It is one of the evaluation scales for predictive models that evaluates the performance of models and determines the strength of the relationship between dependent variables and independent variables to calculate variance in the dependent variables, is detail in Eq. 2.

$$R^2 = 1 - RSS/TSS \tag{2}$$

Where: RSS means" Remaining Sum of Squares ", TSS mean" Total Sum of Squares". The value of R is between zero and one; the closer the value is to one, the more correlation between the variables [32].

- MSE (Mean square error): measuring the average of actual values and expected values, whenever the value of MSE is low indicates that the model has a high performance and is calculated by Eq. 3, [33] .

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{3}$$

- RMSE (Root Mean Square Error): one of the most widespread performance measures illustrated in Eq. 4, is calculated by calculating the square difference between the real effort and the expected effort [34].

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Ea_i - Ep_i)^2}{N}} \tag{4}$$

(Ea) Represents the real effort, and (Ep) represents the expected effort, N represents the number of elements of data [35].

- MAE (Mean absolute error): The average of the variations in the absolute values between each expected effort ($\hat{e}$) and the actual effort (e) is determined by the mean absolute error is appear in Eq. 5, where N represents the number of projects [36].

$$MAE_i = \frac{1}{N}\sum_{i=1}^{N}|e_i - \hat{e}_i| \tag{5}$$

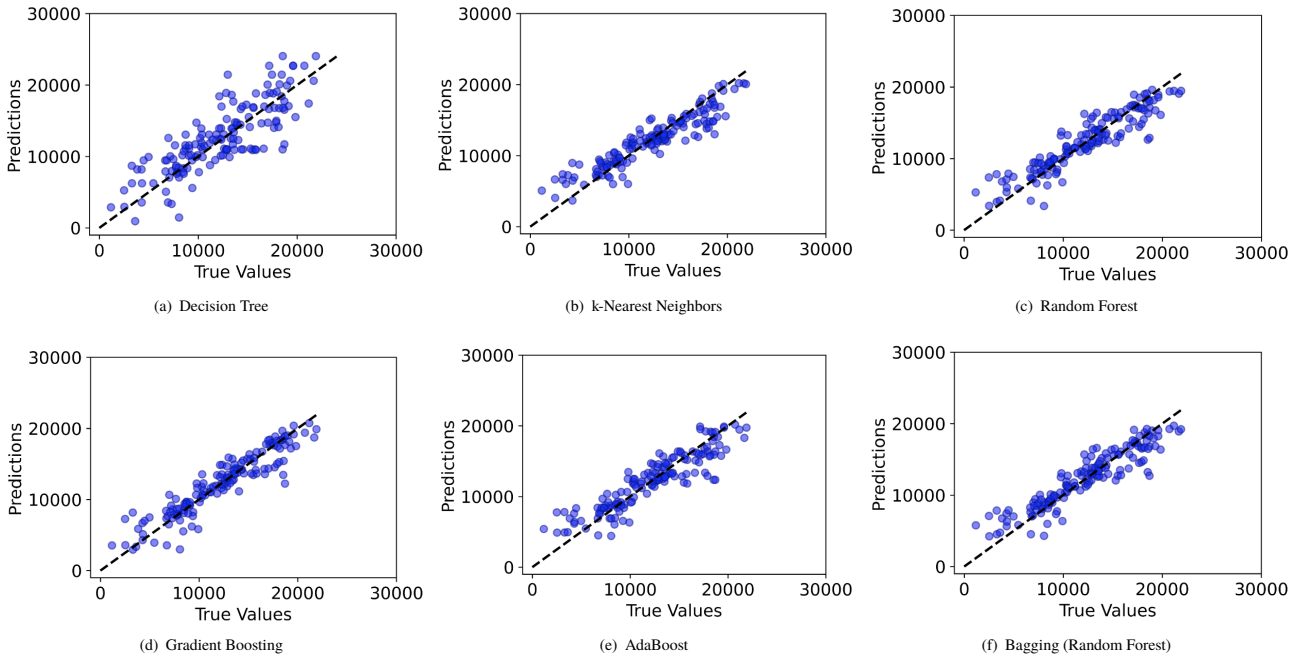
- MMRE (Mean Magnitude of Relative Error): It is a scale used to calculate relative errors of data values by calculating the absolute percentage of relative errors. The relative error is calculated by calculating the difference between actual values and estimated values using Eq. 6 [11].

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i \times 100 \tag{6}$$

where MRE represents the difference between the estimated value and the actual value and divides the result by the actual value, N represents all absolute values [37].

3. Results and discussion

This part discusses the results that appeared when applying the proposed filling model to estimate the cost of software projects using the ISBSG data set. Data was processed as mentioned in the previous section, including deleting missing data, removing redundancy, and selecting high-quality and sound projects according to classifications A and B. Data processing was completed, columns that were not related to cost estimation were removed, and a number of characteristics associated with the estimation process were identified. 17 properties were adopted. Then divided the data into training and testing, with 20% testing and 80% training, and applied the proposed method, which includes five independent models: Random Forest, Decision Tree, KNN, Ada Boosting, and Gradient Boosting. The results showed a high accuracy of up to 97%. Evaluation metrics were used. MAE, MSE, RMSE, R^2, and MMRE. To evaluate the performance of the proposed model, the results showed that the proposed model outperformed. The results of the performance measures showed the superiority of the proposed model over other models; the value of MAE (206.6208) was lower compared to other models; the lower the value, the better the results. While the MSE (4.1842) value was lower than that of individual models, this meant that the variance was lower. As for the RMSE used to measure the prevalence of errors and outliers, its value (646.858) was lower compared to the rest of the models. The last metric, R^2, is the most common to evaluate predictive models, as its value ranges between (0) and (1), measuring the amount of variance whenever its value approaches (1) Its best value when applied to the proposed model is 0.97%. These measures show the efficiency of the proposed model and its high accuracy in prediction. While the MMRE metrics to measure the absolute mean of relative error gave a lower percentage (0.1118) compared to other models, Table 2 shows the results using the mentioned metrics. Figure 8 shows the results using the scatter function, which shows the true values of effort versus the expected values of each independent model as well as the proposed model. Figure 8 shows the difference between the real values and the expected values for each independent model and the proposed model, and it was found that the results of the proposed model are closer to the values of the real effort compared to the individual models.



(a) Decision Tree (b) k-Nearest Neighbors (c) Random Forest

(d) Gradient Boosting (e) AdaBoost (f) Bagging (Random Forest)

Figure 8. Real values of effort versus the expected values.

Through the results that appeared when applying the proposed models (RF, DT, KNN, Ada boosting, and Gradient boosting) using the ISBSG data set, the RF model gave the best performance and less contrast than the rest of the models when applying evaluation metrics (MAE, MSE, RMSE, MMRE, and R^2). The results were, respectively, (205.1023, 4.1802; 646.5521, 0.1007; and 0.9752); it was adopted as a basic model in the bagging method. But the KNN model gave the lowest prediction accuracy, R^2 (0.73%), and a high variance MAE (997.0451). It was the worst among the models. While the models (DT, Ada

boosting, and Gradient boosting) had good accuracy, they gave high variance and a high relative error rate (MMRE = 0.1469, 5.1431, 0.3786), which means accuracy is lower in predicting approximate actual values compared to RF [10, 30, 37–40]. The proposed model was compared with previous studies; the relative error rate (MMRE) in the proposed method was lower compared to a number of previous studies; a closer ratio to zero indicates a higher accuracy in prediction close to the actual values. The results showed the superiority of the proposed model in the accuracy of the prediction of the real effort; the results of the comparison are shown in Table 3.

Table 2. Performance measures for proposed models.

Algorithms	Evaluation metrics				
	MAE	MSE	RMSE	MMRE	R^2
RF	205.1023	4.1802	646.5521	0.1007	0.9752
DT	383.0245	8.4624	919.918	0.1469	0.9497
KNN	997.0451	4.5414	2131.0777	0.6701	0.7302
Ada boost	225.7527	4.1337	642.942	5.1431	0.9743
Gradient boosting	355.9886	5.4043	735.144	0.3786	0.9679
Proposed model	206.6208	4.1842	646.858	0.1118	0.9755

Table 3. Comparison of the results of the proposed method using the MMRE metric with previous studies.

Research	Model	MMRE
	Proposed method	**0.1118**
Fadhil [10]	DolBat	14.576
Hasanluo [30]	(PSO) and (KNN)	12.530
Maher [37]	Stacked Ensemble	00.140
Ahmad [38]	Whale–crow optimization	0.2442
Asghari Agcheh [39]	(GA) with (BA)	34.800
Abdulmajeed [40]	Elman Neural Network	43.160

4. Conclusions

The process of estimating the cost of developing software projects is an essential and important step in the software life cycle that contributes to the success of the project and the delivery of the product within a specified time and budget. Inaccurate estimation of the cost leads in many cases to failure of work, and need to find a more effective model to estimate the cost. In this paper, a proposed bagging ensemble learning algorithm is proposed to estimate the cost of developing software projects and estimate effort. It is a more accurate and closer way to the real cost. The method was to use ISBSG data sets. After processing the data, characteristics related to the cost estimate were determined, then the data was divided into training and testing by (80%) and (20%), and performance was evaluated using the following metrics: MAE, MSE, RMSE, R^2, MMRE, The results showed the superiority of the RF model over the rest of the individual models adopted as a basic model in the bagging method, where obtained high accuracy in prediction (97%) and a relative error rate (MMRE = 0.1118), Compared to previous studies, shown in Table 3, it proved superior in prediction, and the error rate was small compared to other models (DolBat, Elman Neural Network, Whale-Crow Optimization). In Future work, we can use Deep Learning Models with stacking ensemble learning, and the work can be applied to more than one data set, as the KNN model, it performs poorly when applied to the ISBSG dataset.

Authors' contribution

All authors contributed equally to the preparation of this article.

Declaration of competing interest

The authors declare no conflicts of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] M. Abadi *et al.*, "Tensorflow: Large scale machine learning on heterogeneous distributed systems," *arXiv preprint*, 2016. [Online]. Available: https://arxiv.org/abs/1603.04467

[2] R. Sarno and J. Sidabutar, "Comparison of different neural network architectures for software cost estimation," *International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, 2015. [Online]. Available: https://doi.org/10.1109/IC3INA.2015.7377748

[3] N. Rankovic, D. Rankovic, M. Ivanovic, and L. Lazic, "A new approach to software effort estimation using different artificial neural network architectures and taguchi orthogonal arrays," *IEEE Access*, pp. 26 926–26 936, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3057807

[4] A. Mohamed, A. Omar, and S. Ibrahim, "Intrusion detection network attacks based on whale optimization algorithm," *Ingenierie des Systemes d'Information*, vol. 27, no. 3, 2022. [Online]. Available: https://doi.org/10.18280/isi.270310

[5] K. Khashan, S. Dhamyaa *et al.*, "Comparison between the two methods of optimization: Genetic algorithm (ga) and ant colony algorithm (aco) for the propulsion system of uav," *Al-Qadisiyah Journal for Engineering Sciences*, vol. 16, no. 2, 2023. [Online]. Available: https://doi.org/10.30772/qjes.2023.180482

[6] M. D. H. Almawlawe *et al.*, "A gwo-pid controller with advanced optimization features for dc-dc converters," *Signal Processing (DSP)*, vol. 11, p. 13, 2023. [Online]. Available: https://doi.org/10.30772/qjes.2023.180482

[7] Zhejiang University SEL Laboratory, *Docker Containers and Container Cloud*. Beijing: People's Post Electronic Publishing House, 2016.

[8] J. Li, *TensorFlow Technology Analysis and Practice*. Beijing: People's Posts and Telecommunications Publishing House, 2017.

[9] B. Marapelli, "Software development effort duration and cost estimation using linear regression and k-nearest neighbors machine learning algorithms," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 2, pp. 1043–1047, 2019. [Online]. Available: https://doi.org/10.35940/ijitee.K2306.129219

[10] A. A. Fadhil, G. A. Rasha, and M. A. Atica, "Software cost estimation based on dolphin algorithm," *IEEE Access*, vol. 8, pp. 75 279–75 287, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.2988867

[11] M. A. Nawaz *et al.*, "A methodology for software cost estimation using machine learning techniques," *International Conference on Recent Trends in Artificial Intelligence, IOT, Smart Cities and Applications (ICAISC)*, 2020. [Online]. Available: https://ssrn.com/abstract=3647639

[12] M. A. Shah *et al.*, "Ensembling artificial bee colony with analogy-based estimation to improve software development effort prediction," *IEEE Access*, vol. 8, pp. 58 402–58 415, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.2980236

[13] B. Khan, R. Naseem, M. Binsawad, M. Khan, and A. Ahmad, "Software cost estimation using flower pollination algorithm," *Journal of Internet Technology*, vol. 21, no. 5, pp. 1243–1251, 2020. [Online]. Available: https://doi.org/10.3966/160792642020092105002

[14] H. D. P. De Carvalho, R. Fagundes, and W. Santos, "Extreme learning machine applied to software development effort estimation," *IEEE Access*, vol. 9, pp. 92 676–92 687, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3091313

[15] S. KR, "Software cost and effort estimation using ensemble duck traveler optimization algorithm (edto) in earlier stage," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 13, pp. 3300–3311, 2021. [Online]. Available: https://doi.org/10.17762/turcomat.v12i13.914

[16] M. S. Khan *et al.*, "Optimizing deep learning model for software cost estimation using hybrid meta-heuristic algorithmic approach," *Computational Intelligence and Neuroscience*, 2022. [Online]. Available: https://doi.org/10.1155/2022/3145956

[17] J. Rashid, S. Kanwal, Wasif *et al.*, "An artificial neural network-based model for effective software development effort estimation," *Computer Systems Science and Engineering*, 2022. [Online]. Available: https://doi.org/10.32604/csse.2023.026018

[18] R. Marco, S. S. Syed Ahmad, and S. Ahmad, "Bayesian hyperparameter optimization and ensemble learning for machine learning models on software effort estimation," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 3, 2022.

[19] D. Sudigyo *et al.*, "Literature study of stunting supplementation in indonesian utilizing text mining approach," *Procedia Computer Science*, vol. 216, pp. 722–729, 2023.

[20] A. Omar and A. El-Hafeez, "Optimizing epileptic seizure recognition performance with feature scaling and dropout layers," *Neural Computing and Applications*, pp. 1–18, 2023. [Online]. Available: https://doi.org/10.1007/s00521-023-09204-6

[21] A. Burkov, *The Hundred-Page Machine Learning Book*. Quebec City, QC, Canada: Andriy Burkov, 2019, vol. 1. [Online]. Available: https://doi.org/10.1080/15228053.2020.1766224

[22] X.-S. Yang, *Introduction to Algorithms for Data Mining and Machine Learning*. London: Academic Press, 2019, vol. 1.

[23] E. Rodriguez S., F. Eduardo *et al.*, "Effort and cost estimation using decision tree techniques and story points in agile software development," *Mathematics*, vol. 11, no. 6, p. 1477, 2023. [Online]. Available: https://doi.org/10.3390/math11061477

[24] I. C. Suherman and R. Sarno, "Implementation of random forest regression for cocomo ii effort estimation," in *2020 International Seminar on Application for Technology of Information and Communication (iSemantic)*, 2020. [Online]. Available: https://doi.org/10.1109/iSemantic50169.2020.9234269

[25] A. Kanneganti, "Using ensemble machine learning methods in estimating software development effort," Master's thesis, DiVA Portal, 2020. [Online]. Available: https://www.diva-portal.org/

[26] S. Raschka and V. Mirjalili, *Python Machine Learning*. Birmingham: Packt, 2017.

[27] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018. [Online]. Available: https://doi.org/10.1002/widm.1249

[28] X. Dong *et al.*, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, pp. 241–258, 2020. [Online]. Available: https://doi.org/10.1007/s11704-019-8208-z

[29] N. Valinejad, "Maintenance cost estimation for load handling equipment: Maintenance cost estimation for reachstackers using gradient boost regressor estimator," Master's thesis, Tampere University, 2020. [Online]. Available: https://urn.fi/URN:NBN:fi:tuni-202009086926

[30] M. Hasanluo and S. F., "Software cost estimation by a new hybrid model of particle swarm optimization and k-nearest neighbor algorithms," *Journal of Electrical and Computer Engineering Innovations (JECEI)*, vol. 4, no. 1, pp. 49–55, 2016. [Online]. Available: https://doi.org/10.22061/jecei.2016.556

[31] A. Pandey and A. Jain, "Comparative analysis of knn algorithm using various normalization techniques," *International Journal of Computer Network and Information Security*, vol. 11, no. 11, 2017. [Online]. Available: https://doi.org/10.5815/ijcnis.2017.11.04

[32] F. Ahmad and M. Laheeb, "Software development effort estimation techniques: A survey," *Education Science Journal*, vol. 2, p. 23, 2022. [Online]. Available: https://doi.org/10.33899/edusj.2021.129868.1156

[33] K. P. Murphy, *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.

[34] J. Kaur *et al.*, "Neural network - a novel technique for software effort estimation," *International Journal of Computer Theory and Engineering*, vol. 2, no. 1, p. 17, 2010.

[35] A. Nassif, H. Danny *et al.*, "Towards an early software estimation using log-linear regression and a multilayer perceptron model," *Journal of Systems and Software*, vol. 86, no. 1, pp. 144–160, 2013. [Online]. Available: https://doi.org/10.1016/j.jss.2012.07.050

[36] A. B. Nassif *et al.*, "Software development effort estimation using regression fuzzy models," *Computational Intelligence and Neuroscience*, vol. 2019, 2019. [Online]. Available: https://doi.org/10.1155/2019/8367214

[37] M. Maher and S. Jamal, "An ensemble model for software development cost estimation," in *2022 5th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, 2022. [Online]. Available: https://doi.org/10.1109/ISRITI56927.2022.10052861

[38] S. W. Ahmad and G. R. Bamnote, "Whale-crow optimization (wco)-based optimal regression model for software cost estimation," *Sadhana*, vol. 44, pp. 1–15, 2019. [Online]. Available: https://doi.org/10.1007/s12046-019-1085-1

[39] A. S. Asghari and S. Farhad, "A new approach to software cost estimation by improving genetic algorithm with bat algorithm," *Journal of Computer and Robotics*, vol. 11, no. 2, pp. 17–30, 2018.

[40] A. A. Abdulmajeed, A. A. Marwa, and M. T. Tawfeeq, "Predict the required cost to develop software engineering projects by using machine learning," in *Journal of Physics: Conference Series*, vol. 1897, no. 1, 2021, p. 012029. [Online]. Available: https://doi.org/10.1088/1742-6596/1897/1/012029

How to cite this article:
Nedaa T. Qassem and Ibrahim A. Saleh (2026). 'Software cost estimation technique based on bagging ensemble learning algorithm', Al-Qadisiyah Journal for Engineering Sciences, 19(2), pp. 001- 194. https://doi.org/10.30772/qjes.2024.147128.1138