Received 21 September 2024, accepted 5 October 2024, date of publication 14 October 2024, date of current version 25 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3480829

RESEARCH ARTICLE

Enhancing Software Effort Estimation in the Analogy-Based Approach Through the Combination of Regression Methods

TAGHI JAVDANI GANDOMANI[1], (Senior Member, IEEE), MAEDEH DASHTI[2], HAZURA ZULZALIL[3], (Member, IEEE), AND ABU BAKAR MD SULTAN[3]

[1]Department of Computer Science, Faculty of Mathematical Sciences, Shahrekord University, Shahrekord 8818634141, Iran
[2]Data Science Research Group, Department of Computer Science, Faculty of Mathematical Sciences, Shahrekord University, Shahrekord 8818634141, Iran
[3]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, Selangor 43400, Malaysia

Corresponding author: Taghi Javdani Gandomani (javdani@sku.ac.ir)

This work was supported in part by the Universiti Putra Malaysia.

ABSTRACT The success of software projects is closely linked to accurate effort estimation, driving continuous efforts by researchers to refine estimation methods. Among various techniques, the analogy-based approach has emerged as a widely-used method for software effort estimation. However, there is still a need to improve its accuracy and reliability. This study aims to enhance software effort estimation in analogy-based methods by introducing a hybrid approach that combines multiple regression methods with feature weighting. The proposed approach evaluates various regression models, integrating them with analogy-based estimation using a weighted combination of project features. The objective is to improve the precision of effort estimation by optimizing similarity functions and project attribute weights. Experimental results demonstrate that the hybrid model significantly outperforms traditional analogy-based methods, achieving superior accuracy across various software project datasets. The findings highlight the potential of this approach to offer a more dependable foundation for software effort estimation, contributing to the success of software projects.

INDEX TERMS Software effort estimation, analogy-based software estimation, regression methods, machine learning.

I. INTRODUCTION

Software effort estimation is a matter of great importance, as it can play a crucial role in the success or failure of such projects. Accurate effort estimation is crucial in determining the outcome of software projects, significantly impacting their success or failure [1]. Its significance spans various domains such as project financial planning, strategic decision-making, and project evaluation and control. The accuracy of initial cost estimates holds the key to ascertaining the availability of funds required for project completion. Additionally, it aids in determining the project's economic feasibility facilitating informed resource allocation decisions.

Factors such as project size, duration, and team structure wield substantial influence over effort estimation [2]. The correlation between a project's size and complexity underscores the necessity of early-stage cost estimates for larger projects [3], [4]. Moreover, the nature of the project and organizational strategies employed during planning and management significantly impact effort estimation accuracy. Notably, some researchers advocate that precise effort estimation bolsters stakeholder confidence and fosters stronger business relationships [5].

However, software projects commonly grapple with discrepancies between estimated and actual costs due to fluctuating requirements, estimation inaccuracies, technical challenges, and resource fluctuations [6]. The inherent reliance on available information sometimes leads to significant disparities between estimated and actual costs,

The associate editor coordinating the review of this manuscript and approving it for publication was Claudia Raibulet.

owing to incomplete or restricted access to pertinent information.

Despite differing perspectives on software effort estimation, Analogy-based estimation (ABE) has gained traction as a multifaceted technique in software effort estimation [7], [8], [9], [10]. ABE involves leveraging past projects with similarities to derive effort estimates for current projects [11]. This method relies heavily on the selection and weighting of features, which are the attributes or characteristics of the software projects. Feature weighting determines the relative importance of each feature in the estimation process. The goal is to assign appropriate weights to features so that the most relevant ones have a greater influence on the estimation outcome. While feature selection involves identifying the most relevant features to use in the estimation process. The aim is to improve model performance by eliminating irrelevant or redundant features. Yet, its seemingly straightforward approach faces challenges arising from the non-normal distribution of data in software development [12]. In response to the unique nature of software projects, researchers have explored approaches aiming to enhance estimation accuracy. One prevalent strategy involves assigning weights to project attributes within the framework of the analogy-based method [13]. This study aims to evaluate regression methods and assess their collective impact on project outcomes by assigning varying weights to key project characteristics.

The article progresses with the introduction of the analogy-based method in Section II, followed by an overview of research within the analogy-based method in Section III. Section IV introduces the proposed model, while Sections V and VI delve into experimental design and analysis of results, respectively. Finally, the article ends with Sections VII and VIII presenting limitations and conclusion and key findings, respectively.

II. ANALOGY-BASED ESTIMATION (ABE)

The ABE method stands out for its straightforward and pragmatic approach. Unlike formula-driven methods, this technique relies on comparative analysis. Estimating effort for a new project involves comparing it with completed projects deemed similar. This process, which is shown in Figure 1, hinges on information gathered from previous projects, housed within historical datasets or repositories. The methodology encompasses four key components [14]:

1) Historical dataset: Establishing a historical dataset forms the bedrock of this approach, derived from either real or artificial datasets.
2) Identification of project characteristics: Gathering the traits of a new project aligns with the characteristics observed in the historical dataset.
3) Utilization of defined similarity functions: Employing predefined similarity functions, such as Euclidean or Manhattan distance criteria, aids in identifying projects akin to the new project in terms of shared attributes.
4) Project effort estimation: Calculating and evaluating the effort required for the new project entails employing solution functions like mean and median.

The subsequent sections will delve into a brief exploration of each component within the ABE system.

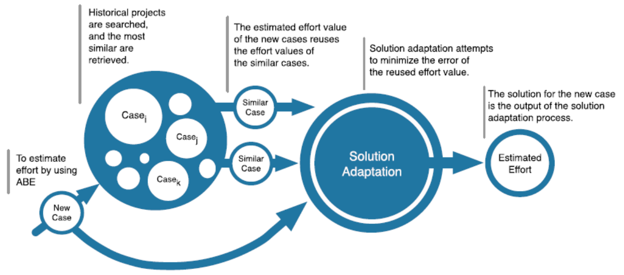
Historical projects are searched, and the most similar are retrieved.
The estimated effort value of the new cases reuses the effort values of the similar cases.
Solution adaptation attempts to minimize the error of the reused effort value.
The solution for the new case is the output of the solution adaptation process.
To estimate effort by using ABE
Case$_i$
Case$_j$
Case$_k$
Similar Case
Similar Case
Solution Adaptation
Estimated Effort
New Case

FIGURE 1. The ABE process [15].

A. SIMILARITY FUNCTION

In ABE, the function of similarity plays an important role. This function is applied to determine how similar two different projects are [16]. The standard formulation of the similarity function is as follows:

$$sim\left(p,p'\right)=f\left(Lsim\left(f_1,f_1'\right).Lsim\left(f_2,f_2'\right)....Lsim\left(f_n,f_n'\right)\right) \quad (1)$$

In this equation, p and ṕ represent the new and existing projects in the repository, respectively. The terms, f_i,and f_i' denote the values of the i^{th} feature for these projects, while n is the total number of features considered in each project. The function $Lsim()$ computes the similarity between corresponding features of the projects. Both $Lsim()$ and $f()$ functions define the overarching structure of the similarity function.

To elaborate, the similarity function begins by assessing each feature pair from the projects. For every feature i, the function $Lsim\left(f_{\mathrm{i}},f_{\mathrm{i}}'\right)$ measures how similar the feature value f_i of the new project is to the feature value f_i' of the old project. This process is repeated for all n features. The results are then aggregated using the function $f()$, which synthesizes these individual similarity measures into a single similarity score for the entire project pair. This comprehensive score helps in determining how closely the new project resembles the existing projects, thereby aiding in estimation and decision-making processes based on past project data.

In our study, we applied several similarity functions in order to enhance the accuracy of effort estimation for software. The difference is that, in our research, a feature weighting method was added and two other parameters, w and δ were introduced. The weight assigned to each feature is w, and it takes a value between 0 and 1. Another parameter is δ, which is introduced as a very small number to avoid a zero denominator. Each of these similarity functions has seen considerable usage in the literature, and their applications are detailed below to provide context and justification for their inclusion in our study [17].

1) EUCLIDEAN DISTANCE

This distance measure is based on the geographical distance between two points in multidimensional space. Each feature is considered a dimension, and the closer these two points are to each other, the more similar the projects look. The Euclidean distance criterion formula between two samples, A and B, is shown in equation (2)

$$d\,(A,B) = \sqrt{\sum\nolimits_{i=1}^{n} w_i (A_i - B_i)^2 + \delta} \tag{2}$$

In this formula, A_i and B_i represent different features in dimension i and, n is the number of features. The weight assigned to the i^{th} feature is denoted as w_i, and set δ to 0.0001. This applies to all similarity functions

The Euclidean distance is one of the most popular similarity measures applied to the software effort estimation domain. It calculates the straight-line distance between two points in a multidimensional space. Prior research, including the work of Shepperd and Schofield [16], justified the capability of the Euclidean distance function to estimate the effort, because of its capability to be relatively stable even in the presence of numerical data and its easy implementation.

2) MANHATTAN DISTANCE

In this distance measure, the distance between two samples is equal to the sum of the absolute values of the difference in their corresponding characteristics. The Manhattan distance criterion formula between two samples A and B is shown in equation (3).

$$d\,(A,B) = \sqrt{\sum\nolimits_{i=1}^{n} w_i \left|A_i - B_i\right| + \delta} \tag{3}$$

In this formula, A_i and B_i represent different features in dimension i,and n is the number of features.

The Manhattan distance, also known as the L1 norm, simply adds up the absolute differences between the coordinates of two points. This function has been employed in several studies; for example, Dolado [18] used this to capture the cumulative differences in attributes and found it useful in cases where outlier influence needs to be reduced.

3) MAXIMUM-DISTANCE SIMILARITY

This function calculates the similarity between two vectors or data points based on the maximum distance between them along any dimension or feature in their respective coordinates. This is a measure of dissimilarity, where a higher value indicates greater dissimilarity between vectors. Mathematically, the similarity function of the maximum distance between two vectors A and B can be calculated as follows (equation (4)):

$$\begin{aligned} &MXD\,(A,B) \\ &\quad = \max\left(w_1 \left|A_1 - B_1\right| .w_2 \left|A_2 - B_2\right| \ldots .w_n \left|A_n - B_n\right|\right) \end{aligned} \tag{4}$$

where A and B are the two vectors to be compared, and n is the total number of dimensions or features in the vectors.

4) MAHALANOBIS

Mahalanobis similarity function calculates the Mahalanobis distance between two data points A and B, taking into account the correlation between features (through the covariance matrix) and the independence of scale (through the inverse covariance matrix). Calculating this similarity function is given in equation (5).

$$Mahalanobis\,(A,B) = \sqrt{(A-B)^T . \sum -1. (A-B) + \delta} \tag{5}$$

where A and B are two data points that are compared. $\sum -1$ Indicates the inverse of the covariance matrix, which shows the correlation between features.

Mahalanobis distance takes into account the correlations of the data set and is useful for identifying outliers. This function has been applied in several advanced estimation models, including the work by Briand et al. [19], who utilized it to improve the accuracy of effort estimation by considering the variance and covariance of project attributes.

5) AKRITEAN SIMILARITY FUNCTION

This function, known as the heuristic distance function, is an innovative approach that combines elements of both the Euclidean (EUC) and Manhattan (MHT) distance functions. This combination is designed to take advantage of the strengths of each distance metric, making it adaptable to different scenarios, effectively covering both mostly empty and filled areas in a dataset. It is calculated as equation (6).

$$AKritean\,(A.B) = \omega_1 .EUC(A,B) + \omega_2 .MHT(A,B) \tag{6}$$

where A and B are two data points. ω_1 and ω_2 are weights assigned to Euclidean and Manhattan distances, respectively.

In our study, each of the considered similarity functions has been applied to the comparison of software projects with respect to their attributes, like lines of code, functional points, and complexity metrics. The choice of the appropriate similarity function is one of the important factors that can noticeably affect the performance of the ABE method. Our effort estimation model aims at drawing on the respective strengths of these different similarity functions applied in our effort, so that it turns out to be more robust. The selection process for each of these similarity functions was based on their tested effectiveness found and cited in the literature.

B. K-NEAREST NEIGHBORS (KNN)

The k-Nearest Neighbors (KNN) algorithm plays a significant role in ABE. Once the distances are calculated, KNN identifies the 'k' nearest projects (analogies) that are most similar to the new project. KNN, a sample-based learning approach, proves to be a robust tool in pattern recognition and classification tasks. It is especially well-suited for ABE because it intuitively fits the notion of identification and exploiting analogies from past projects to estimate effort for new projects.

One of the main reasons that ABE is vastly dominated by KNN is the simplicity and effectiveness of the latter. There are no assumptions made in KNN regarding the underlying distribution of data, which may turn out quite advantageous if one considers the variety of software projects. Proximity is the basic principle on which KNN works, and makes it very intuitive and easy to implement. This simplicity does not come at a cost related to its performance. Overall, KNN has generally demonstrated the capability of returning accurate and trustworthy estimates in various domains.

The selection of the parameter 'k' is a crucial factor in achieving accurate results. A small 'k' can lead to classifications that are highly sensitive to noise or outliers, while a large 'k' may result in overly generalized decisions that overlook local nuances in the data. Therefore, determining an optimal 'k' value is essential to strike a balance between capturing meaningful patterns and avoiding undue influence from noise. This adaptability in choosing 'k' underlines the versatility of KNN and makes it rather appropriate for various contexts within software effort estimation.

Other classification methods, such as Decision Tree (DT), Support Vector Regression (SVR), Random Forest (RF), or neural networks, would not usually be used within ABE. These are techniques that introduce richer model structures and assumptions that clearly do not match the basic principle of analogy-based reasoning. Unlike KNN, which makes direct use of the notion of similarity to past projects, such alternative methods were designed for predictive modeling based on feature importance or separation of classes and thus deviate from the core idea of ABE.

The literature, however, has dramatically favored KNN with this regard, since it is capable of directly and interpretably identifying analogies. While other methods bring along their strengths if they are situated in different contexts, they generally increase unnecessary complexities in an ABE process. As such, KNN remains the best choice for ABE, providing a simple, yet reliable and effective way of identifying similar software projects and hence effort estimation based on past experiences.

By keeping a focus on KNN and its benefits, we will make sure our approach stays very close to conventional ABE; it gives a blatant reason as to why it should still be used and also shows its efficiency in this particular domain.

C. SOLUTION FUNCTION

Solution functions play a pivotal role in determining how effort is estimated from selected historical projects. These functions encompass methods such as closest analogy, inverse weighted mean (IWM), mean, and median, each influencing how the estimated effort aligns with chosen similar projects [20]. The IWM approach stands out by assigning distinct weights to each similar project, ensuring that their impact on the final estimated effort is proportionate to their similarity with the new project. This weighted aggregation reflects the significance of each similar project in contributing to the estimation result. The calculation of the inverse weighted average is illustrated in equation (7):

$$\widehat{C_p} = \sum\nolimits_{k=1}^{n} \frac{Sim(P.P_k)}{\sum_{i=1}^{n} Sim(P.P_k)} C_{pk} \tag{7}$$

In this equation, P represents the project for which the cost will be estimated. P_k represents the kth similar project. $Sim\,(P, P_k)$ Denotes the similarity between projects P and P_k. C_{pk} Indicates the cost of the most similar project to P_k. The IWM approach intricately balances similarity and cost considerations, offering a nuanced perspective in estimating project efforts based on historical data.

D. REGRESSION METHODS IN EFFORT ESTIMATION

Regression, a fundamental statistical method, is employed to model the relationship between a dependent variable and one or more independent variables. The primary goal of these methods is to predict the value of the dependent variable based on the given independent variables. Noteworthy among these methods are RF, DT, SVR, and Linear Regression (LR), each briefly explained below:

1) DECISION TREE (DT)

A DT is a predictive model that employs a tree-like structure for decision-making. In this method, the dataset is partitioned based on feature values, creating branches that culminate in predictions at leaf nodes. Constructed using historical data from software projects, a DT incorporates project characteristics such as size, complexity, and development methodology as attributes. The resulting tree can then predict the cost of new projects based on their characteristics [21], [22].

2) RANDOM FOREST (RF)

RF stands out as an ensemble learning method that utilizes multiple DTs during training and prediction. By combining predictions from various trees, this method aims to enhance accuracy and mitigate overfitting. Each tree is trained on a random subset of data and features, and the final prediction usually relies on the average (regression) or majority vote (classification) of all trees. For software effort estimation, RF can be employed by training the model on project features. The resulting model can predict effort or cost for new projects, taking into account the interactions between attributes [23], [24], [25].

3) SUPPORT VECTOR REGRESSION (SVR)

SVR, a regression technique within the Support Vector Machine (SVM) framework, aims to find a hyperplane that best fits data points while minimizing margin violations. Particularly effective for non-linear relationships between variables, SVR can be used to estimate software cost. It models the relationship between project characteristics and cost, excelling in handling nonlinearity and complex relationships in data pertinent to software effort estimation [26], [27].

4) LINEAR REGRESSION (LR)

LR, a basic yet powerful technique, models the relationship between a dependent variable and one or more independent variables as a linear equation. The objective is to find coefficients that minimize the difference between actual and predicted values. Applied to software project characteristics, LR predicts cost based on these attributes, assuming a linear relationship. It offers a straightforward approach to understanding the linear connections between features and cost [28], [29].

In essence, these regression methods provide a diverse toolkit for software effort estimation, with each method possessing unique strengths suitable for various scenarios.

E. ENSEMBLE METHODS IN SOFTWARE EFFORT ESTIMATION

Ensemble methods in software effort estimation are strategic approaches that involve integrating the results of multiple individual models to enhance the accuracy and reliability of forecasting the effort required for software development. These methods stem from the concept that combining predictions from diverse models often yields superior performance compared to relying on a single model. Ensemble methods are broadly categorized into homogeneous and heterogeneous groups.

1) HOMOGENEOUS ENSEMBLE METHODS

These methods combine the outputs of learners from the same base but with different structures or parameters. This could involve using a singular machine learning technique with varied configurations or applying different parameters within a single technique, such as negative correlation checks, binning methods, or stochastic subspace strategies. The underlying principle is to aggregate outputs from a single method but with various adjustments or modifications.

2) HETEROGENEOUS ENSEMBLE METHODS

In contrast, heterogeneous methods combine the outputs of different machine learning techniques to introduce diversity in predictions generated by distinct algorithms. Rather than focusing on variations of a single technique, this approach integrates outputs from entirely different machine learning methods.

An extensive analysis of 24 research articles spanning 2000 to 2016 revealed a predominant emphasis on homogeneous Ensemble Software Effort Estimation methods in the literature, constituting 17 of the 24 studies [30]. These studies commonly utilized machine learning techniques as base learners. The integration of outputs from these base learners involved applying a diverse range of aggregation rules, totaling 12 in number. The choice of aggregation rules depended on the nature of the machine learning tasks, whether classification or regression, as well as the specific combination approach, whether homogeneous or heterogeneous. In 2023, Idri et al. [30] updated the 2016 systematic review with new findings from studies published between 2016 and 2020 [31]. Following established guidelines, they appraised the 2016 review's methods and created an updated protocol. From 3,682 retrieved papers, 30 papers were selected for detailed analysis. Their findings confirm that machine learning remains the most common technique for constructing Effort Estimation Models, with ensemble techniques outperforming individual models.

Moreover, some innovative methods combine Ensemble learning with feature selection techniques. For instance, Kocaguneli et al. [32] introduced a heterogeneous ensemble approach that incorporated nine distinct machine learning techniques alongside sequential forward selection (SFS), a feature selection algorithm. In this method, various combination rules such as weighted mean, ranked mean, and inverse were applied to merge the results of these techniques. Similarly, Hosni et al. [33] investigated the impact of two filter feature selection methods on a heterogeneous set comprising four separate machine learning techniques, aiming to identify the effect of these feature selection strategies on the set's performance.

III. RELATED WORK

In the area of software effort estimation, several techniques have been proposed and enhanced over time. In this section, a review of the latest related and most relevant works concerning the problem our research tackles is provided, along with a comparative evaluation underlining the improvements brought in by our proposed approach.

The ABE is still very much in use as a rather intuitively appealing and effective approach [8]. The primary technique within ABE is the KNN algorithm, which has been extensively studied and validated.

More recent developments include improving feature selection and weighting and incorporating a range of dissimilarity metrics to provide better similarity measures [34], [35]. One seminal work by Albrecht and Gaffney [36] introduced the concept of function points for software estimation and provided one of the earliest applications of analogy-based methods. Based on this foundation, our approach is built with the use of more sophisticated similarity functions and feature weighting mechanisms in order to achieve increased accuracy. Shepperd and Schofield [16] also used KNN in estimation of software projects and proved it effective in real-world situations. In this paper, we further this work by investigating optimal values of 'k' and refining the process of measuring similarities, making the estimation more robust and flexible. While the ABE approach may appear straightforward in the realm of software development, its practical implementation faces intricate challenges arising from the unique characteristics of data distribution.

Some studies have integrated conventional estimation techniques with machine learning methods to leverage their individual strengths. Meenakshi and Pareek [37] contributed a detailed systematic review on estimation of software effort using deep learning techniques. Their study showed the potential of deep learning methods in improving effort estimation by identifying intricate patterns in software engineering data. In this vein, Li et al. [38] integrated deep learning into software effort estimation by extracting semantic features from project descriptions to increase the accuracy of predictions. Deep-SE applies deep learning for feature extraction; our approach is oriented to enhancing analogy-based estimation using optimized KNN. This retains the interpretability and ease of implementation without the computational complexity of the deep models. Similarly, Chen et al. [39] exploited a pre-trained GPT-2 model for the semantic relationships in software project data and obtained state-of-the-art results in effort estimation. In particular, our method complements the strengths of GPT2SP since we offer a much simpler, but still very effective approach that does not require significant computational resources and is therefore more feasible for smaller projects or smaller organizations with weaker computational capacity.

Researchers have shown that dynamically adjusting the feature weights during the effort estimation process can significantly improve the accuracy and reliability of the results [10], [40]. The inherent complexity of software projects, influenced by factors extending beyond mere similarity, necessitates a nuanced approach to weight evaluation for ensuring method effectiveness. Indeed, optimization techniques have emerged as invaluable methods for refining and enhancing the feature weighting approach in software estimation models. These techniques play a pivotal role in calibrating the importance and influence of different features within the model, thereby elevating the accuracy and reliability of the estimation process. Moreover, they underscore the dynamic nature and adaptability of software estimation, enabling developers and stakeholders to tailor model behavior to the intricacies of the dataset and project scope. By meticulously adjusting the weights assigned to various features, these optimization methods contribute to a more precise and context-aware estimation process.

A diverse array of optimization techniques is employed in the field, and notable algorithms include Genetic Algorithms (GAs) [41], [42], [43] and Differential Evolution (DE) [4], [44], both drawing inspiration from natural selection. Within software effort estimation, GAs have proven pivotal in enhancing the accuracy and reliability of cost forecasting. The process commences by representing potential solutions as a set of parameters defining software project features. An initial population of potential solutions is generated, and their fitness is assessed using a predefined fitness function, often based on historical data or other pertinent criteria. The most suitable solutions, those closely aligned with desired outcomes, are selected for reproduction through the crossover operation, simulating genetic recombination. To maintain genetic diversity, random mutations are introduced. Over successive generations, GAs guide the algorithm towards an optimal or near-optimal parameter set, refining the accuracy of effort estimation by selecting features and weights that best estimate software project costs. Although GAs and DE are powerful tools for optimization and could produce suitable results when used for feature weighting, they come with limitations such as slow convergence, premature convergence, sensitivity to parameter settings, high computational costs, and scalability issues.

Swarm intelligence algorithms contribute significantly to the optimization of software estimation models, with notable examples including Particle Swarm Optimization (PSO) algorithms [45], [46] and the Ant Colony Optimization (ACO) algorithm [47]. In PSO, potential solutions representing project features are conceptualized as particles moving through a parameter space. These particles dynamically adjust their properties based on their best-known position and the best-known global position. The movement of particles, guided by a fitness function, aims to minimize the disparity between predicted and actual costs. PSO effectively navigates the solution space, iteratively refining project characteristics to enhance the accuracy of software cost estimates. Conversely, the ant colony algorithm draws inspiration from the search behavior of ants. Potential solutions are envisioned as paths through project features. Ants, mimicking routing behavior, repetitively select features, leaving synthetic pheromones along their chosen paths. The intensity of these pheromones is influenced by the quality of the solution. Through iterative convergence, these algorithms identify optimal or near-optimal sets of project characteristics. This iterative process significantly improves the accuracy of software cost estimates, providing valuable support for informed decision-making in project planning and management. While swarm intelligence algorithms like PSO and ACO offer robust mechanisms for optimizing feature weights, their application in ABE is constrained by issues such as parameter sensitivity, computational cost, scalability, premature convergence, and the complexity of implementation. Addressing these limitations often requires hybrid approaches, adaptive parameter tuning, and careful empirical validation to ensure accurate and reliable effort predictions.

Hybrid methods have been studied in which ABE is combined with other techniques in a process making use of multiple estimation strategies. Satapathy and Rath [48] combined ABE with regression models for the enhancement of the estimation process by using a hybrid approach to address the limitations of each individual method. Conforming to a strictly analogical approach, our approach significantly refines both the processes of similarity assessment and analogy identification to achieve high accuracy with a model that is also simple and easy to interpret. On their part, Jorgensen and Shepperd [49] gave an overview of different software effort estimation methods and some of their strengths and weaknesses, including ABE and other approaches. Our work takes into consideration the findings presented here, and we have targeted the weaknesses of ABE identified, most prominently in measurement of similarity and selection of analogies.

We have tried to propose a new approach introducing a number of key improvements over the existing approaches. We increase the robustness and flexibility of the estimation process by including multiple similarity functions and optimizing the selection of ‘k’. In this way, this multifaceted measurement of similarity is more adaptable to different project characteristics, thus ensuring better performance across diverse datasets. We further adopt feature weighting to make sure that only the most relevant project attributes have priority in their use for the task of increasing the accuracy of analogy identification. This refactoring gives better emphasis on the most impacting features and thus warrants a more accurate and reliable estimate. Our approach is found to be of wider application among users, without much loss in accuracy, when compared with deep learning models that are overwhelmed by computational complexity. Such a balance between simplicity and high performance makes our approach suitable for small- and large-scale projects without large computational resources. On the whole, we are oriented to present an approach that shall SCE under traditional KNN-based ABE with visible enhancements in similarity measurement and feature weighting.

IV. THE PROPOSED MODEL

In this section, we propose a hybrid model that integrates multiple regression techniques (RF, DT, SVR, and LR) with the ABE framework. The novelty of our approach lies in the following key aspects:

Weighted Combination of Regression Methods: Unlike traditional ensemble methods, which focus on combining learners from similar classes (homogeneous) or entirely different machine learning techniques (heterogeneous), our hybrid model blends regression methods through an optimized weighted averaging technique. The weights for each regression model are dynamically adjusted based on historical project data, ensuring a more precise and context-sensitive estimation process.

Feature Weighting Mechanism: A major limitation of existing ensemble methods is the lack of effective feature weighting within analogy-based estimation frameworks. Our approach introduces a novel feature weighting technique that assigns varying degrees of importance to different project features (e.g., lines of code, functional points, and complexity metrics). This ensures that the most relevant project characteristics significantly influence the estimation, improving the model’s accuracy and interpretability.

Dual-Phase Optimization: The hybrid model operates in two distinct phases—training and testing. The first phase, model training, involves presenting a set of training data to the model. In this phase, the analogy-based method and regression methods are employed to assign weights to control parameters. Subsequently, in the testing phase, the trained model is utilized for effort estimation on test data.

In the training phase, our objective is to identify the most optimal combination of similarity functions, solution functions, and other controllable parameters within the analogy-based method. We aim to automate the majority of parameter selection through machine learning methods. To achieve this, we employ the weighted combination method of regression models. These models are trained using the Leave-One-Out Cross-Validation (LOOCV) method on the training dataset. LOOCV is used to evaluate the predictive performance of the ABE approach rigorously. By using each project in the dataset as a test case while the rest of the data serve as the training set, LOOCV ensures that the model is tested on every available data point.

This method allows for detailed error analysis, as it provides an error estimate for each project. This can help in understanding how the model performs across different types of projects and identify any patterns in the errors. LOOCV helps validate the model by simulating a real-world scenario where the model has to predict the effort for unseen projects. This validation step is crucial to ensure that the model generalizes well to new data and is not overfitting. Subsequently, the trained model is scrutinized in the testing phase using testing data for comprehensive evaluation.

This dual-phase model aligns with the ABE approach, leveraging both historical data and regression models to enhance the accuracy and reliability of software effort estimation. The integration of machine learning techniques in the training phase adds a layer of adaptability and optimization, allowing the model to adjust to the unique characteristics of different software projects dynamically. In the testing phase, the model’s robustness is assessed, providing insights into its effectiveness in real-world applications.

A. TRAINING PHASE

The training phase, illustrated in Figure 2, operates based on the structure of the analogy-based method, utilizing a set of historical projects as the training dataset for effort estimation. The initial step in training is the selection of an appropriate similarity function. This article employs five available similarity functions. Notably, the selection of the similarity function is the only step that cannot be automated. Once a similarity function is selected, machine learning algorithms optimize other parameters. These include feature weights, initially chosen randomly within the range of zero and one. The number of k nearest neighbors, a critical parameter in analogy-based methods, ranges between 1 and 9, with the optimal value determined by the proposed model. The last parameter, the solution function, uses three well-known functions, and its optimization is delegated to machine learning methods.

The proposed model repeats the process using a weighted regression model for the number of similarity functions (5 times) to obtain the best combination for the ABE approach. The model generates a continuous vector with length N+4, with the first N cells representing project features, and then four cells are added to this vector, where cell N+1

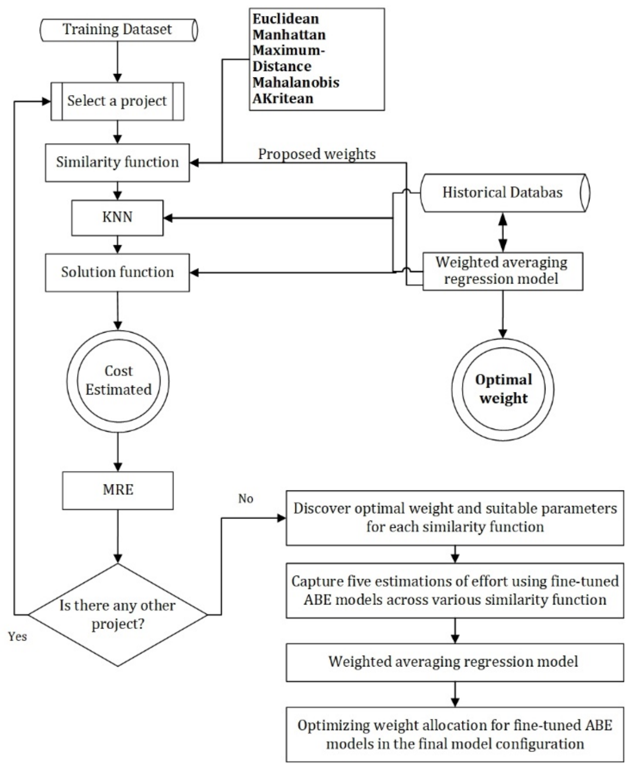
Training Dataset
Euclidean
Manhattan
Maximum-Distance
Mahalanobis
AKritean
Select a project
Similarity function
Proposed weights
Historical Databas
KNN
Weighted averaging regression model
Solution function
Cost Estimated
Optimal weight
MRE
No
Discover optimal weight and suitable parameters for each similarity function
Capture five estimations of effort using fine-tuned ABE models across various similarity function
Is there any other project?
Yes
Weighted averaging regression model
Optimizing weight allocation for fine-tuned ABE models in the final model configuration

FIGURE 2. Training phase architecture.

represents the number of K. The nearest neighbor, cell N+2, represents the solution function, which can have an integer value between 1 to 3 (because three solution functions are used in this paper). The remaining two cells, namely N+3 and N+4, respectively, correspond to the parameters that we need to obtain for different similarity functions. Figure 3 provides more details.

To calculate the final effort for a project, the proposed method combines the results of the five optimized ABE models, assigning weights based on a weighted average formula (Equation 8):

$$\begin{aligned}\text{Final Effort} &= W_{EUC} \times \text{effort}_{EUC} + W_{MHT} \times \text{effort}_{MHT} + W_{AKR} \\ &\quad \times \text{effort}_{AKR} + W_{MXD} \times \text{effort}_{MXD} + W_{MHL} \times \text{effort}_{MHT}\end{aligned} \tag{8}$$

This equation represents a composite of efforts estimated by the five optimized ABE models, emphasizing the significance of $w_{\text{simlarityfunction}}$ coefficients. These coefficients determine the relative impact or contribution of each ABE approach to the final estimate. The optimization process ensures that each weight falls within the range [0, 1], with the crucial constraint that the sum of all weights equals 1: $W_{EUC} + W_{MHT} + W_{MXD} + W_{MHL} + W_{AKR} = 1$. This rigorous optimization aims to determine the most effective combination of weights, ensuring accurate and comprehensive effort estimation for a given project in the proposed model.

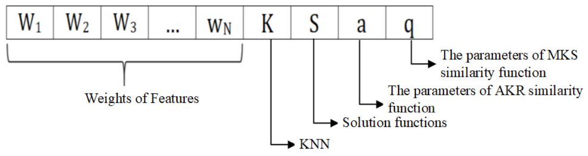
W1
W2
W3
...
WN
K
S
a
q
The parameters of MKS similarity function
Weights of Features
The parameters of AKR similarity function
Solution functions
KNN

FIGURE 3. Initial encoding of a workable solution for the Proposed method in training stage.

B. THE PROCESS OF COMBINING BASIC REGRESSION MODELS

Figure 4 illustrates the weighted averaging process used to determine the final output for a new sample, calculated as equation (9):

$$O_t = \sum_{i=1}^{4} w_i . o_i \tag{9}$$

In this equation, i denotes the index of the base regression model, w_i represents the weight of the base model number i, o_i Is the output of the base model number i for the given input, and O_t Signifies the final output for the input sample. Notably, w_i Is a value between 0 and 1 ($0 \leq w_i \leq 1$), and the condition of the sum of weights equaling 1 holds true in this equation ($\sum_{i=1}^{4} w_i = 1$).

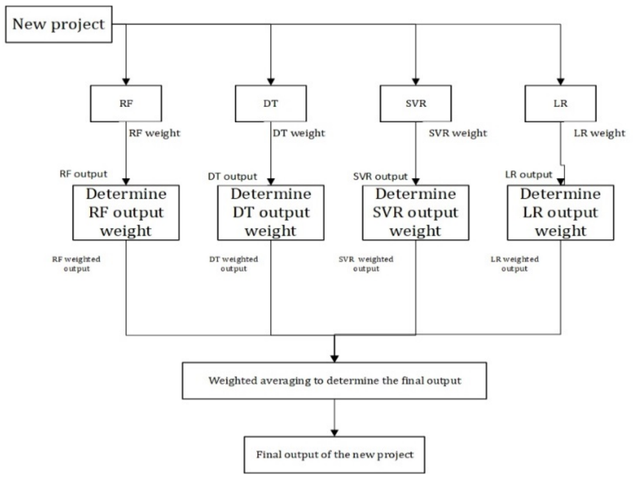
New project
RF
DT
SVR
LR
RF weight
DT weight
SVR weight
LR weight
RF output
DT output
SVR output
LR output
Determine RF output weight
Determine DT output weight
Determine SVR output weight
Determine LR output weight
RF weighted output
DT weighted output
SVR weighted output
LR weighted output
Weighted averaging to determine the final output
Final output of the new project

FIGURE 4. The process of combining basic regression models.

C. TESTING PHASE

The testing phase involves the rigorous evaluation of the proposed model's effectiveness. Its primary objective is to assess the model's capability to estimate the effort required for new and unseen projects accurately. This phase encompasses a systematic evaluation, utilizing values obtained in the training phase, to estimate effort for projects in the testing dataset. The estimation process involves a comparative analysis between the selected project in the testing dataset and the entire dataset.

Effort estimation for each unseen project is initiated using the optimized ABE model with its specific similarity function. The optimized coefficients from the training process, along with the optimal ABE model-specific feature weighting, are applied. This approach ensures that each ABE model maximizes its potential, capturing details identified during the optimization process. Subsequently, efforts for test dataset projects are generated using each of the six optimized ABE models. The weighted average method combines these results, providing a comprehensive and optimal effort estimate for the test project.

The estimation process is iteratively executed for all projects in the test dataset, yielding effort estimates for each project. The final step involves measuring the model's performance through various metrics. These metrics collectively offer a comprehensive assessment of the model's reliability and accuracy in project effort estimation. The testing stage, illustrated in Figure 5, shows the steps, from applying optimized ABE models and aggregating results to calculating performance metrics.

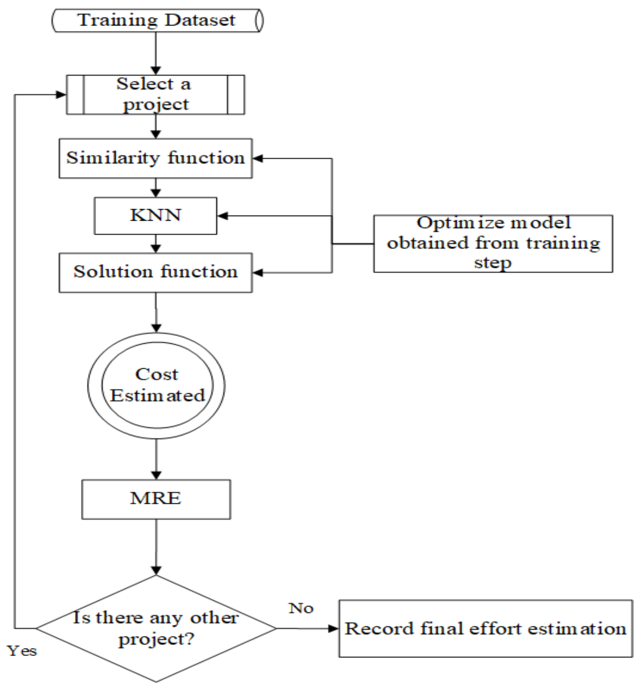
Training Dataset
Select a project
Similarity function
KNN
Solution function
Optimize model obtained from training step
Cost Estimated
MRE
Is there any other project?
Yes
No
Record final effort estimation

FIGURE 5. Testing phase.

V. EXPERIMENT DESIGN

In this section, the methodology for evaluating the proposed model, including datasets, evaluation criteria, and the testing process, is outlined.

A. DATASET

A diverse set of datasets has been meticulously selected to comprehensively evaluate the proposed model, ABE with feature weighting. These datasets encompass a variety of software projects, each distinguished by unique characteristics such as lines of code, functional requirements, and complexity. The diversity ensures the model's performance is assessed across different software domains and project features. The following datasets are briefly explained:

1) ALBRECHT DATASET

Albrecht dataset Introduced by Albrecht and Gaffney in 1983, this dataset originates from IBM software projects and is crucial for function point analysis (FPA) and software project estimation. It explores the relationship between software size and the effort required for development [36]. This dataset contains software project data from IBM, including attributes such as the number of function points, source lines of code (SLOC), and actual effort in person-hours. It is primarily used for function point analysis and software effort estimation.

2) COCOMO DATASET

The "Constructive Cost Model" (COCOMO) dataset, particularly the "Cocomo81" dataset, is associated with the original COCOMO model developed by Barry Boehm in 1981. It contains historical data from various software projects, serving as a benchmark for calibrating and validating software effort estimation models [50]. It contains attributes like project size (measured in lines of code), effort multipliers, and actual effort in person-months.

3) CHINA DATASET

This dataset comprises data from projects developed by Chinese companies, featuring 19 distinct features and 499 records [51]. The independent variables focus on functional components, including input, output, query, file, and interface. These components contribute to calculating function points, a key metric in software effort and size estimation.

4) KEMERER DATASET

Kemerer dataset Widely used to study different aspects of software development, the Kemerer dataset provides information about software projects, including lines of code (SLOC), function points, and development effort [52].

5) MAXWELL DATASET

This dataset offers insights into industrial software projects developed in Finland by large commercial banks. With details from 62 projects, it considers several key independent features for analysis [53]. Key attributes include project size, complexity metrics, and actual effort in person-hours.

These datasets were chosen for the research community and had a number of problems with accessing them. Most of the commercial datasets are pretty large in size, and hence it is pretty hard to get them for there are strict privacy concerns that govern the use of such data, coupled with the proprietary nature of the datasets. Our study did not use any industrial dataset, thereby making our study more reproducible and transparent by using publicly available data. Some of the other historical datasets used as benchmarks in software effort estimation research are the Albrecht, COCOMO, and

Maxwell datasets. These datasets will provide scope for consistent comparison with previous studies, thus helping in validation and comparisons of the methods proposed by us.

Source datasets are carefully chosen, correctly documented, and widely recognized to be of high quality and relevance. They were already validated and used substantially by previous research, which means that our findings will be based upon reliable data.

B. EVALUATION CRITERIA

To quantitatively assess the accuracy and effectiveness of the proposed model, a set of evaluation criteria, including the Magnitude of Relative Error (MRE), has been adopted. These metrics provide a comprehensive understanding of how well the estimated effort aligns with the actual effort for each project. Below is an explanation of the MRE criterion and its calculation.

1) MAGNITUDE OF RELATIVE ERROR (MRE)

MRE measures the relative difference between the estimated value and the actual value, typically expressed as a percentage. The Relative Error (RE), a fundamental component in MRE calculation, represents the difference between the estimated (or calculated) value and the actual value in relation to the actual value. Equation (10) illustrates how to compute the Relative Error.

$$Relative\ Error\ \ (RE) = \frac{(Estimated\ Value - Actual\ Value)}{Actual\ Value} \tag{10}$$

The MRE specifically refers to the absolute value of the relative error, providing a measure of the difference irrespective of direction. Equation (11) details the calculation of MRE for each project.

$$MRE_i = \frac{|\,(Estimated\ Value_i - Actual\ Value_i)\,|}{AActual\ Value_i} \tag{11}$$

A lower MRE indicates a closer match between the estimated value and the actual value, signifying higher accuracy in the estimation process. The MRE criterion is fundamental in evaluating the model's performance by quantifying the discrepancies between estimated and actual efforts.

2) MEAN MAGNITUDE OF RELATIVE ERROR (MMRE) AND ITS VARIATIONS

Mean Magnitude of Relative Error (MMRE) and its Variations**:** MMRE is a crucial metric for assessing the performance of estimation methods, models, or algorithms. It provides an average estimation error across all records in a dataset, offering insights into how well the estimated values align with the true values. The calculation of MMRE is defined in equation (12), where 'i' represents the index of each sample, and 'N' is the total number of samples in the dataset.

$$MMRE = \frac{1}{N}\sum\nolimits_{i=1}^{N} MRE_i \tag{12}$$

A lower MMRE signifies that, on average, the estimated values are closer to the actual values, indicating better performance of the estimation method or model. MMRE serves as a comprehensive benchmark to understand the estimation method's efficacy in various scenarios.

Recognizing MMRE's potential imbalance, two additional criteria, the Best Minimum Magnitude of Relative Error (BMMRE) and the Best Inverse Magnitude of Relative Error (BIMMRE), have been introduced. These metrics aim to address the limitations of MMRE by normalizing error rates. Equations (13) and (14) depict the calculation for BMMRE and BIMMRE, respectively.

$$BMMRE = \frac{1}{N}\sum\nolimits_{i=1}^{N} \frac{|\text{Estimated Value} - \text{Actual Value}|}{\text{Min(Estimated Value, Actual Value)}} \tag{13}$$

$$BIMMRE = \frac{1}{N}\sum\nolimits_{i=1}^{N} \frac{|\text{Estimated Value} - \text{Actual Value}|}{\text{Max(Estimated Value, Actual Value)}} \tag{14}$$

Both BMMRE and BIMMRE offer ways to normalize error rates, providing a more nuanced understanding of the differences between estimated and actual efforts. These metrics enhance the evaluation process by accounting for specific features in the error distributions.

C. SETUP AND IMPLEMENTATION

The test setup is carefully designed to ensure reliable and consistent results. The proposed model is applied to each dataset and the estimated efforts are generated for all projects. These estimates are then combined with actual effort values to calculate the aforementioned evaluation metrics. By repeating this process for each data set, a comprehensive assessment of the accuracy of the technique in different scenarios is obtained.

The validation approach employed in this study uses a rigorous process to ensure the integrity of the results, and the Leave One Out (LOO) cross-validation technique is used, dividing the dataset into training and testing subsets. This protects against overfitting and provides a realistic estimate of the technique's performance on unseen data. This approach also takes into account potential variations in data set characteristics and project features, thus strengthening the generalizability of the findings.

In summary, the experimental design implemented in this study was thoughtfully designed to evaluate the accuracy and robustness of the proposed model. Using diverse datasets, comprehensive evaluation metrics, a rigorous testing process, and a rigorous validation approach, we ensure that the obtained results provide valuable insights into the potential of the technique to increase the accuracy of software effort estimation.

VI. RESULTS AND ANALYSIS

A. RESULTS

In this section, we present the outcomes derived from the amalgamation of four fundamental regression methods across five distinct datasets. All simulations and evaluations were conducted under uniform conditions. The primary objective is to discern the impact of combining these basic regression methods. We scrutinize the results of each method based on the dataset used, employing various evaluation criteria.

1) ALBRECHT DATASET RESULTS

The results from the Albrecht dataset are illustrated in Table 1.

TABLE 1. Results from the albrecht dataset.

Method	MMRE	BMMRE	BIMMRE	MdMRE	PRED (0.25)
RF	1.0322	1.1738	0.3447	0.2920	0.3750
DT	1.1774	1.473	0.4284	0.4655	0.2083
SVR	1.9753	2.7743	0.4777	0.62666	0.2083
LR	1.3431	0.4891	1.1153	0.5151	0.2083
Hybrid	0.9513	1.0730	0.3408	0.2978	0.4583

Table 1 provides a comparative analysis of the performance of different regression methods, assessed by various evaluation criteria. Notably, the "hybrid method" emerges as the top-performing approach across all criteria. It achieved the lowest values for Median Magnitude of Relative Error (MdMRE), BIMMRE, BMMRE, and MMRE, signifying its superior predictive accuracy. In contrast, LR and DT exhibit similar performance with relatively higher errors than the hybrid method". Conversely, the results for SVR are notably high, indicating elevated values of MdMRE and BMMRE, suggesting potential ineffectiveness in this dataset. Although RF demonstrates acceptable performance, it falls short of the accuracy achieved by the "hybrid method". These findings highlight the significance of selecting an appropriate regression technique, with the 'hybrid method' demonstrating its potential as an effective choice.

2) COCOMO81 DATASET RESULTS

Table 2 displays the results obtained by applying the proposed model to the COCOMO81 dataset.

TABLE 2. Results from the Cocomo81 dataset.

Method	MMRE	BMMRE	BIMMRE	MdMRE	PRED (0.25)
RF	4.2835	4.4449	0.6461	2.5300	0.1111
DT	1.3972	2.2522	**0.5369**	**0.7505**	0.1111
SVR	3.0258	4.7938	0.6090	0.9565	0.1428
LR	16.7701	5.1769	12.4902	3.9790	0.0952
Hybrid	4.2326	2.3887	2.2657	1.6043	0.0793

The evaluation criteria result for the Cocomo81 dataset across different regression methods unveil significant trends and distinctions. Notably, DT exhibits the lowest MdMRE and deviation in BMMRE, suggesting its precision in predictions. However, the context of the dataset should be considered, as the LR method yields notably high errors across all measures. RF and SVR methods present competitive results with performance levels close to each other, though slightly lower than the accuracy achieved by DT. It is crucial to highlight the consistent performance of the hybrid method", striking a balance between accuracy and complexity. Interestingly, the LR method demonstrates an exceptionally high deviation in BIMMRE and MMRE values, indicating a substantial bias in its predictions. This analysis underscores the importance of method selection in the context of the Cocomo81 dataset, with DT emerging as a promising choice for precise predictions, while the hybrid method offers a well-balanced alternative.

3) MAXWELL DATASET RESULTS

Table 3 displays the results obtained by applying the proposed model to the Maxwell dataset.

TABLE 3. Results from the maxwell dataset.

Method	MMRE	BMMRE	BIMMRE	MdMRE	PRED (0.25)
RF	0.6979	0.7518	**0.3142**	**0.3011**	**0.4354**
DT	0.6097	0.7650	0.3391	0.3717	0.3709
SVR	1.4138	1.9082	0.4857	0.5513	0.1935
LR	1.0015	0.4936	0.7817	0.5236	0.2741
Hybrid	0.6337	0.69839	0.3212	0.3678	0.4032

The detailed analysis of the Maxwell dataset across various regression methods, as presented in Table 3, offers valuable insights into the prediction accuracy of each method. RF stands out as the top performer, showcasing the lowest MdMRE and BMMRE among all methods, underscoring its exceptional accuracy in predictions. While the DT method remains competitive, it slightly trails RF in accuracy, albeit with fewer errors compared to other methods. Conversely, SVR exhibits a noticeable bias in BIMMRE and MMRE values, indicating potential bias and higher error rates in its predictions. Intriguingly, the "hybrid method" strikes a commendable balance between accuracy and complexity, rendering it a pragmatic choice for this dataset. In summary, the Maxwell dataset analysis highlights RF as the superior-performing method, delivering the most accurate predictions, while SVR might warrant further investigation due to higher bias and error rates.

4) KEMERER DATASET RESULTS

Table 4 shows the results obtained by applying the proposed model to the Kemerer dataset.

The examination of the Kemerer dataset through various regression methods, detailed in Table 4, illuminates their predictive performance, with SVR emerging as a notable standout among the methods. SVR demonstrates remarkably low MdMRE and BMMRE, indicating unparalleled accuracy in predictions. DT and RF methods also yield competitive

TABLE 4. Results from the kemerer dataset.

Method	MMRE	BMMRE	BIMMRE	MdMRE	PRED (0.25)
RF	0.9180	1.1889	0.4122	0.5531	0.2000
DT	0.8681	1.2345	0.4374	0.4379	0.2000
SVR	**0.0666**	0.5482	**0.4643**	1.5154	1.0828
LR	1.3303	**0.4687**	0.8871	0.9088	0.2000
Hybrid	0.7239	0.9534	**0.3547**	**0.3902**	**0.4000**

results, displaying relatively lower errors compared to alternative techniques. Conversely, LR exhibits significant bias in BIMMRE and MMRE values, signifying substantial bias and prediction errors. Intriguingly, the ''hybrid method'' strikes a balance between accuracy and complexity, making it a pragmatic choice for this dataset. Notably, this method achieves the highest PRED value among its counterparts. In summary, the Kemerer dataset analysis underscores the exceptional predictive accuracy of the hybrid method'', closely trailed by DT and RF, while the higher bias and error rate of LR warrant further investigation.

5) CHINA DATASET RESULTS

The China dataset undergoes meticulous scrutiny employing various regression methods to assess their predictive prowess, as detailed in Table 5. In this dataset, RF emerges as a standout performer, boasting the lowest values in most evaluation criteria. These results underscore the exceptional accuracy of RF in predicting this dataset. DT and SVR also deliver competitive performance, exhibiting relatively lower errors compared to alternative methods. Conversely, LR demonstrates higher deviation in BIMMRE and MMRE values, signifying potential bias and prediction errors in its outcomes. Interestingly, the hybrid method'' closely aligns with LR in the results, with a notable improvement in the PRED criterion. In summary, the analysis of the China dataset accentuates RF as the superior performing method, closely trailed by DT and SVR, while the higher bias and error rate of LR merits further investigation.

TABLE 5. Results from the china dataset.

Method	MMRE	BMMRE	BIMMRE	MdMRE	PRED (0.25)
RF	**1.7481**	**1.9855**	**0.4798**	**0.6371**	**0.1883**
DT	2.0153	2.8702	0.5350	0.7352	0.1743
SVR	2.2136	3.0341	0.7624	0.5419	0.1683
LR	1.9565	2.1755	0.4967	0.6488	0.1903
Hybrid	1.9184	2.1575	0.4923	0.6410	**0.2044**

In this study, we benchmarked the performance of the proposed hybrid model against established methods using widely recognized metrics, including MMRE, BMMRE, BIMMRE, MdMRE, and PRED. The observed improvements across all datasets consistently demonstrate the hybrid model's superiority. Given the substantial and consistent differences, we believe that these results provide clear evidence of the model's practical advantages, following standard comparison practices in the field.

VII. DISCUSSION

A. IN-DEPTH ANALYSIS

This research primarily deals with the assessment of the effectiveness of regression methods in software effort estimation. The reason for using these four regression methods—RF, DT, SVR, and LR—is that these methods are popular, well-known, and have been proved effective in many scenarios. The purpose behind the orientation in these methods is to demonstrate that the solo application of these regression techniques can be clubbed to improve their prediction accuracy by following a hybrid approach.

This research has, therefore, been limited to regression methods to enable a deep analysis, unlike many methods proposed in literature consideration, including deep learning models, genetic algorithms, and many others from the area of machine learning. The reason is a desire to study any possible synergies among regression techniques and to develop a clear and easily interpretable framework that can be implemented in a straightforward manner without the computational complexity associated with more advanced models.

Additional experiments were conducted for the complete evaluation of results through an analysis of computational overhead introduced by each regression method, sensitivity to various training settings, and the quality of the hybrid model in contrast to individual models. More specifically, computational overhead for every regression method was measured in terms of training and prediction times on different datasets. These datasets were drawn with a view toward their diversity and representativeness for different software projects.

Results indicated that although RF and SVR were more computationally intensive, their accuracy justifies their use in the hybrid model. In particular, RF performed well with large datasets containing a large number of features due to its ensemble nature that avoids overfitting. On the other hand, SVR did very well in capturing complex and nonlinear relationships within data, which is very important in ensuring the accuracy for effort estimation, especially for heterogeneous software projects.

Various training settings, such as the number of trees used in RF, the depth of DT, or the kernel functions in SVR, have been changed to verify how they impact prediction accuracy. For example, increasing the number of trees generally improved the accuracy of RF at the cost of raising computational time. Similarly, the depth of DT could be adjusted with respect to the balance of model interpretability and performance, and kernel functions provided different insights into the adaptability of SVR during different data distributions. Results indicated that fine-tuning these parameters can have a strong impact on model performance.

This robustness of the hybrid model was tested through varying weights assigned to each regression method. Results revealed that, in the majority of evaluation criteria, the hybrid model outperformed all of the individual methods. For example, in the Albrecht dataset, the MMRE obtained by the Hybrid Model was 0.9513, which was very low in comparison to results when single methods were used. Further analysis was carried out to study the role of feature selection and weighting within the hybrid model. It could have weighted project features according to their relevance and focused on the most important attributes of the lines of code, functional points, and complexity metrics. The weighting of features hence not only improved the accuracy of the model developed but also increased comprehending ability with respect to the factors that influence software effort estimation.

In addition to accuracy, hybrid model stability and generalizability were tested. To rigorously test its performance in each dataset, the LOOCV method was applied. Since this method involved using every single data point to validate the model, it comprehensively checked the predictive capability of the model. The LOOCV results proved the high accuracy rate of the hybrid model in different project types, hence very strong and versatile. These extra experiments provide further evidence that the suggested hybrid model performs better compared with the solitary regression methods. Thus, this paper further underpins the view that a combination of these regression methods gives more accurate and reliable effort estimates for software, with clearly presented details of the computational overhead, the impact of training settings, and the importance of feature weighting.

Our results, obtained from five commonly used software project datasets (Albrecht, COCOMO81, Maxwell, Kemerer, and China), demonstrate that the hybrid model consistently outperforms traditional ensemble methods. The introduction of feature weighting and optimized regression combinations significantly improves the accuracy of effort estimation. Specifically, the hybrid model reduces the Mean Magnitude of Relative Error (MMRE) and achieves better predictive performance compared to individual regression models and previously proposed ensemble approaches.

B. TRADE-OFFS BETWEEN MODEL COMPLEXITY AND PRACTICAL APPLICABILITY

The hybrid model proposed in this study offers significant improvements in accuracy for software effort estimation by integrating multiple regression methods and feature weighting techniques. However, it is important to acknowledge that increased model complexity might limit its practical applicability, particularly for smaller organizations with limited computational resources.

1) COMPUTATIONAL COMPLEXITY VS. ACCURACY

The hybrid model leverages multiple regression techniques and dynamically adjusts feature weights, making it computationally more intensive than simpler models such as single regression methods or traditional analogy-based approaches. While this complexity allows the model to handle diverse and large datasets more effectively, it may pose challenges for small-scale organizations or resource-constrained environments where computational resources are limited. Despite this, the model maintains a balance by reducing the Mean Magnitude of Relative Error (MMRE) and providing superior predictive accuracy, making it feasible for use in situations where accuracy is prioritized over computational simplicity.

2) INTERPRETABILITY VS. PERFORMANCE

Another trade-off lies in the interpretability of the model. The hybrid model's structure, combining multiple techniques and weighing features, adds layers of complexity that may reduce its interpretability for end users, especially those unfamiliar with machine learning techniques. In comparison, simpler models like Linear Regression (LR) or Decision Trees (DT) offer more straightforward, interpretable results but may sacrifice accuracy in complex scenarios. To mitigate this, the proposed model still prioritizes intuitive methods like analogy-based estimation, maintaining some level of interpretability without compromising performance.

3) PRACTICAL FEASIBILITY IN RESOURCE-CONSTRAINED ENVIRONMENTS

For smaller organizations, practical feasibility is a key concern. While the hybrid model is designed to optimize performance across diverse datasets, it remains essential to assess its computational requirements in real-world applications. To address this, the model can be scaled down by adjusting certain parameters, such as limiting the number of regression methods used or optimizing fewer feature weights, to reduce computational overhead. Additionally, for organizations dealing with smaller datasets, a reduced version of the model could still offer significant performance improvements without requiring substantial computational resources.

4) BALANCING COMPUTATIONAL FEASIBILITY AND ACCURACY

The balance between computational feasibility and accuracy was achieved by testing the hybrid model on diverse datasets with varying levels of complexity. Through Leave-One-Out Cross-Validation (LOOCV), the model was optimized to ensure that its computational demands did not outweigh the performance gains, particularly when applied to medium and large datasets. This approach ensures that the model remains practical for a wide range of applications while still offering improvements in effort estimation accuracy.

While the hybrid model presents increased complexity, the trade-offs between model performance, interpretability, and computational requirements were carefully considered. The model is designed to remain adaptable, making it applicable in real-world scenarios, with potential adjustments for resource-constrained environments.

VIII. LIMITATIONS

In any research study, it is essential to acknowledge the limitations that may affect the validity and generalizability of the findings. This study explores enhancing software effort estimation using an analogy-based approach and regression methods. Despite its contributions to the field, there are several limitations that must be considered.

One significant limitation of this study, like similar studies, is the reliance on a limited number of commonly used datasets for software development effort estimation and the lack of access to large-scale commercial datasets from industry projects. Unlike recent studies leveraging extensive industrial datasets, our research is based on publicly available datasets due to proprietary restrictions and privacy concerns, which restricts our ability to use such data [54], [55]. Consequently, the datasets used in this study, while well-established and reliable, are smaller compared to those used in some recent works like Deep-SE and GPT2SP, which utilized datasets with over 23,000 issues [38]. The widespread use of these datasets is primarily for comparing the efficiency of software estimation models. However, this smaller size use may restrict the generalizability of the findings. The specific characteristics and contexts of these datasets may not fully capture the broader spectrum of software development projects. Consequently, the model developed in this study might perform well on these specific datasets but may not generalize effectively to other datasets with different characteristics.

Another key limitation is the absence of comprehensive features related to human factors within the all existing datasets used. Software effort estimation is significantly influenced by human elements such as team experience, individual productivity, and management practices. However, the lack of detailed information on these factors in the datasets limits the reliability and accuracy of the effort estimation. Human factors play a crucial role in determining the actual effort required, and their omission can lead to incomplete and potentially misleading estimates. To address this limitation, future studies should aim to include more detailed information on human factors. This can be achieved through the design of surveys, collection of additional meta-data, and incorporation of qualitative data from interviews or questionnaires. Including these aspects would provide a more comprehensive understanding of the factors influencing software effort estimation, leading to more accurate and reliable models.

In this work, the computational overhead has been checked and the impact of various training settings assessed. A more detailed discussion about the methodological choices and additional experiments is available in Subsection B in Section VI. This subsection is a proper evaluation where computational overhead related to regression methods, different training settings, and detail on how the hybrid model improves over the individual models are discussed. In this regard, we would like to take note of these limitations to provide a balanced view about the contributions from the study and areas where further research is needed. Such limitations can only be addressed in future work dealing with the development of more robust and generalizable software effort estimation models.

IX. CONCLUSION AND FUTURE WORK

Accurate software effort estimation is crucial for project success, driving researchers and experts to refine estimation methods. The analogy-based approach is a popular choice, yet efforts to enhance it continue, as the quest for improved analogy-based methodologies persists.

This study examines the performance of ‘‘hybrid method’’ that powered by the four fundamental regression methods, i.e. RF, DT, SVR, and LR, across the most popular datasets in this field including Albrecht, COCOMO81, Maxwell, Kemerer, and China. The primary objective was to assess the impact of combining these basic regression methods and identify the most effective approach.

Data analysis and performance evaluation showed that the ‘‘hybrid method’’ consistently outperforms the individual regression methods across multiple evaluation criteria, such as MdMRE, BIMMRE, BMMRE, and MMRE, demonstrating its superior predictive accuracy and effectiveness.

In the Albrecht dataset, the ‘‘hybrid method’’ achieved the lowest error values, indicating its ability to provide the most accurate predictions compared to the other regression techniques. In the COCOMO81 dataset, DT method exhibited the highest precision, with the lowest MdMRE and BMMRE values, while the ‘‘hybrid method’’ maintained a well-balanced performance. For the Maxwell dataset, the RF method emerged as the top performer, demonstrating the lowest MdMRE and BMMRE, highlighting its exceptional accuracy in predictions.

In the Kemerer dataset, SVR method displayed remarkable accuracy, with remarkably low MdMRE and BMMRE values, while the ‘‘hybrid method’’ achieved the highest PRED (0.25) score. Regarding the China dataset, RF method again demonstrated superior performance, with the lowest error values across most evaluation criteria, underscoring its exceptional predictive capability for this dataset.

Overall, the study showed that the ‘‘hybrid method’’ consistently delivers superior predictive accuracy and performance compared to the individual regression techniques, making it a promising and effective choice for practical applications. The ‘‘hybrid method’’ consistently struck a balance between accuracy and complexity, rendering it a pragmatic choice in various scenarios. This versatility positions it as an advantageous option for projects with diverse characteristics. The findings underscore the nuanced nature of selecting regression methods tailored to specific datasets, emphasizing the significance of an informed choice to ensure precise and reliable software project effort estimation. The novelty of our hybrid model lies in its ability to dynamically adapt to project characteristics through weighted combinations of regression methods and a novel feature weighting mechanism. This approach addresses the limitations of existing ensemble methods by optimizing both similarity functions and feature importance within the analogy-based effort estimation framework. The results indicate that this hybrid model offers a significant improvement in the accuracy of software effort estimation, making it a robust and reliable tool for project management and planning.

Future work could explore integrating machine learning techniques to handle dynamic changes in software projects, such as using real-time data to continuously refine effort estimations. Additionally, incorporating deep learning methods for better pattern recognition and feature selection may further improve accuracy. Expanding the application of the model to other domains and developing user-friendly tools for broader use in industry settings are also promising directions for future research.

REFERENCES

[1] Z. Zia, A. Rashid, and K. U. Zaman, "Software cost estimation for component-based fourth-generation-language software applications," *IET Softw.*, vol. 5, no. 1, pp. 103–110, 2011.

[2] M. Agrawal and K. Chari, "Software effort, quality, and cycle time: A study of CMM level 5 projects," *IEEE Trans. Softw. Eng.*, vol. 33, no. 3, pp. 145–156, Mar. 2007.

[3] J. Hihn and H. Habib-Agahi, "Cost estimation of software intensive projects: A survey of current practices," in *Proc. 13th Int. Conf. Softw. Eng.*, vol. 16, 1991, pp. 276–287.

[4] A. K. Bardsiri, "An intelligent model to predict the development time and budget of software projects," *Int. J. Nonlinear Anal. Appl.*, vol. 11, no. 2, pp. 85–102, 2020.

[5] G. Pfajfar, A. Shoham, A. Małecka, and M. Zalaznik, "Value of corporate social responsibility for multiple stakeholders and social impact—Relationship marketing perspective," *J. Bus. Res.*, vol. 143, pp. 46–61, Apr. 2022.

[6] A. Jadhav, M. Kaur, and F. Akter, "Evolution of software development effort and cost estimation techniques: Five decades study using automated text mining approach," *Math. Problems Eng.*, vol. 2022, pp. 1–17, May 2022.

[7] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Inf. Softw. Technol.*, vol. 54, no. 1, pp. 41–59, Jan. 2012, doi: 10.1016/j.infsof.2011.09.002.

[8] A. Idri, F. A. Amazal, and A. Abran, "Analogy-based software development effort estimation: A systematic mapping and review," *Inf. Softw. Technol.*, vol. 58, pp. 206–230, Feb. 2015.

[9] A. Idri, M. Hosni, and A. Abran, "Improved estimation of software development effort using classical and fuzzy analogy ensembles," *Appl. Soft Comput.*, vol. 49, pp. 990–1019, Dec. 2016.

[10] M. Dashti, T. J. Gandomani, D. H. Adeh, H. Zulzalil, and A. B. M. Sultan, "LEMABE: A novel framework to improve analogy-based software cost estimation using learnable evolution model," *PeerJ Comput. Sci.*, vol. 7, p. 800, Jan. 2022, doi: 10.7717/peerj-cs.800.

[11] F. Walkerden and R. Jeffery, "An empirical study of analogy-based software effort estimation," *Empirical Softw. Eng.*, vol. 4, pp. 135–158, Jun. 1999.

[12] V. Khatibi Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, "A flexible method to estimate the software development effort based on the classification of projects and localization of comparisons," *Empirical Softw. Eng.*, vol. 19, no. 4, pp. 857–884, Aug. 2014.

[13] B. Sigweni and M. Shepperd, "Feature weighting techniques for CBR in software effort estimation studies: A review and empirical evaluation," in *Proc. 10th Int. Conf. Predictive Models Softw. Eng.*, Sep. 2014, pp. 32–41.

[14] I. Abnane and A. Idri, "Improved analogy-based effort estimation with incomplete mixed data," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, Sep. 2018, pp. 1015–1024.

[15] P. Phannachitta, J. Keung, A. Monden, and K. Matsumoto, "A stability assessment of solution adaptation techniques for analogy-based software effort estimation," *Empirical Softw. Eng.*, vol. 22, no. 1, pp. 474–504, Feb. 2017.

[16] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Trans. Softw. Eng.*, vol. 23, no. 11, pp. 736–743, Nov. 1997.

[17] N.-H. Chiu and S.-J. Huang, "The adjusted analogy-based software effort estimation based on similarity distances," *J. Syst. Softw.*, vol. 80, no. 4, pp. 628–640, Apr. 2007, doi: 10.1016/j.jss.2006.06.006.

[18] J. J. Dolado, "On the problem of the software cost function," *Inf. Softw. Technol.*, vol. 43, no. 1, pp. 61–72, Jan. 2001.

[19] L. C. Briand, T. Langley, and I. Wieczorek, "A replicated assessment and comparison of common software cost modeling techniques," in *Proc. 22nd Int. Conf. Softw. Eng.*, 2000, pp. 377–386.

[20] L. Angelis and I. Stamelos, "A simulation tool for efficient analogy based cost estimation," *Empirical Softw. Eng.*, vol. 5, pp. 35–68, Mar. 2000.

[21] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown, "An introduction to decision tree modeling," *J. Chemometrics, A J. Chemometrics Soc.*, vol. 18, no. 6, pp. 275–285, Jun. 2004.

[22] A. Najm, A. Zakrani, and A. Marzak, "Systematic review study of decision trees based software development effort estimation," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 7, pp. 542–552, 2020.

[23] X. Li, "Using 'random forest' for classification and regression," *Chin. J. Appl. Entomol.*, vol. 50, no. 4, pp. 1190–1197, 2013.

[24] Z. Abdelali, H. Mustapha, and N. Abdelwahed, "Investigating the use of random forest in software effort estimation," *Proc. Comput. Sci.*, vol. 148, pp. 343–352, Jan. 2019.

[25] A. G. P. Varshini, A. K. Kumari, and V. Varadarajan, "Estimating software development efforts using a random forest-based stacked ensemble approach," *Electronics*, vol. 10, no. 10, p. 1195, May 2021.

[26] A. L. I. Oliveira, "Estimation of software project effort with support vector regression," *Neurocomputing*, vol. 69, nos. 13–15, pp. 1749–1753, Aug. 2006.

[27] M. Awad, R. Khanna, M. Awad, and R. Khanna, "Support vector regression," in *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. NY, USA: Springer, 2015, pp. 67–80.

[28] Z. A. Khalifelu and F. S. Gharehchopogh, "Comparison and evaluation of data mining techniques with algorithmic models in software cost estimation," *Proc. Technol.*, vol. 1, pp. 65–71, Jan. 2012.

[29] X. Su, X. Yan, and C.-L. Tsai, "Linear regression," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 4, no. 3, pp. 275–294, 2012.

[30] A. Idri, M. Hosni, and A. Abran, "Systematic literature review of ensemble effort estimation," *J. Syst. Softw.*, vol. 118, pp. 151–175, Aug. 2016.

[31] J. T. H. A. Cabral, A. L. I. Oliveira, and F. Q. B. da Silva, "Ensemble effort estimation: An updated and extended systematic literature review," *J. Syst. Softw.*, vol. 195, Jan. 2023, Art. no. 111542.

[32] E. Kocaguneli, T. Menzies, and J. W. Keung, "On the value of ensemble effort estimation," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1403–1416, Nov. 2012.

[33] M. Hosni, A. Idri, and A. Abran, "On the value of filter feature selection techniques in homogeneous ensembles effort estimation," *J. Softw., Evol. Process*, vol. 33, no. 6, p. e2343, Jun. 2021.

[34] M. Karimi, T. J. Gandomani, and M. Mosleh, "An integrated approach for estimating software cost estimation using adaptive neuro-fuzzy inference system and the grey wolf optimization algorithm," in *Proc. 14th Int. Conf. Inf. Knowl. Technol. (IKT)*, Dec. 2023, pp. 229–234.

[35] M. Afshari and T. J. Gandomani, "Enhancing software effort estimation with ant colony optimization algorithm and fuzzy-neural networks," in *Proc. 3rd Int. Conf. Distrib. Comput. High Perform. Comput. (DCHPC)*, May 2024, pp. 1–6.

[36] A. J. Albrecht and J. E. Gaffney, "Software function, source lines of code, and development effort prediction: A software science validation," *IEEE Trans. Softw. Eng.*, vol. SE-9, no. 6, pp. 639–648, Nov. 1983.

[37] M. Pareek, "Software effort estimation using deep learning: A gentle review," in *Proc. Int. Conf. Sustain. Innov. Solutions Current Challenges Eng. Technol.* Singapore: Springer, 2023, pp. 351–364.

[38] Y. Li, Z. Ren, Z. Wang, L. Yang, L. Dong, C. Zhong, and H. Zhang, "Fine-SE: Integrating semantic features and expert features for software effort estimation," in *Proc. IEEE/ACM 46th Int. Conf. Softw. Eng.*, Feb. 2024, pp. 1–12.

[39] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, "Generative pretraining from pixels," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1691–1703.

[40] B. B. Sigweni, "An investigation of feature weighting algorithms and validation techniques using blind analysis for analogy-based estimation," Doctoral dissertation, Dept. Comput. Sci., Brunel Univ. London, U.K., 2016.

[41] S.-J. Huang and N.-H. Chiu, "Optimization of analogy weights by genetic algorithm for software effort estimation," *Inf. Softw. Technol.*, vol. 48, no. 11, pp. 1034–1045, Nov. 2006.

[42] Y. F. Li, M. Xie, and T. N. Goh, "A study of genetic algorithm for project selection for analogy based software cost estimation," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage.*, Dec. 2007, pp. 1256–1260.

[43] Y. F. Li, M. Xie, and T. N. Goh, "Optimization of feature weights and number of neighbors for analogy based cost estimation in software project management," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage.*, Dec. 2008, pp. 1542–1546.

[44] T. R. Benala and R. Mall, "DABE: Differential evolution in analogy-based software development effort estimation," *Swarm Evol. Comput.*, vol. 38, pp. 158–172, Feb. 2018, doi: 10.1016/j.swevo.2017.07.009.

[45] M. Azzeh, A. B. Nassif, S. Banitaan, and F. Almasalha, "Pareto efficient multi-objective optimization for local tuning of analogy-based estimation," *Neural Comput. Appl.*, vol. 27, no. 8, pp. 2241–2265, Nov. 2016.

[46] D. Wu, J. Li, and C. Bao, "Case-based reasoning with optimized weight derived by particle swarm optimization for software effort estimation," *Soft Comput.*, vol. 22, no. 16, pp. 5299–5310, Aug. 2018.

[47] S. Ranichandra, "Optimizing non-orthogonal space distance using ACO in software cost estimation," *Mukt. Shabd. J.*, vol. 9, no. 4, pp. 1592–1604, 2020.

[48] S. M. Satapathy and S. K. Rath, "Empirical assessment of machine learning models for agile software development effort estimation using story points," *Innov. Syst. Softw. Eng.*, vol. 13, nos. 2–3, pp. 191–200, Sep. 2017.

[49] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 33–53, Jan. 2007.

[50] B. W. Boehm, *Software Engineering Economics*. Berlin, Germany: Springer, 2002.

[51] J. Li, G. Ruhe, A. Al-Emran, and M. M. Richter, "A flexible method for software effort estimation by analogy," *Empirical Softw. Eng.*, vol. 12, no. 1, pp. 65–106, Jan. 2007.

[52] C. F. Kemerer, "An empirical validation of software cost estimation models," *Commun. ACM*, vol. 30, no. 5, pp. 416–429, May 1987.

[53] K. Maxwell, L. Van Wassenhove, and S. Dutta, "Performance evaluation of general and company specific models in software development effort estimation," *Manage. Sci.*, vol. 45, no. 6, pp. 787–803, Jun. 1999.

[54] V. Tawosi, R. Moussa, and F. Sarro, "Agile effort estimation: Have we solved the problem yet? Insights from a replication study," 2022, *arXiv:2201.05401*.

[55] E. M. D. B. Fávero, D. Casanova, and A. R. Pimentel, "SE3M: A model for software effort estimation using pre-trained embedding models," *Inf. Softw. Technol.*, vol. 147, Jul. 2022, Art. no. 106886.

TAGHI JAVDANI GANDOMANI (Senior Member, IEEE) received the joint bachelor's and master's degrees in computer engineering (software) from the Isfahan University of Technology and Isfahan University in Isfahan, Iran, and the Ph.D. degree from Universiti Putra Malaysia (UPM), in 2014. Currently, he is an Associate Professor of software engineering with Shahrekord University, Shahrekord, Iran. In addition to his teaching and research duties, he leads the Data Science Research Group, Shahrekord University. His research interests include agile software development, software process improvement, software project management, software cost estimation, empirical software engineering, and the application of artificial intelligence in software engineering.

MAEDEH DASHTI received the M.Sc. degree in software engineering from Islamic Azad University, Isfahan Branch, Iran, in 2020. She is currently a Research Assistant with the Data Science Research Group, Shahrekord University, Shahrekord, Iran. She has contributed to national and international research projects, focusing on AI fairness and the ethical implications of AI systems. Her research interests include software cost estimation, machine learning, AI ethics, and empirical software engineering.

HAZURA ZULZALIL (Member, IEEE) received the Ph.D. degree in software engineering from University Putra Malaysia (UPM), Malaysia, in 2011. She is an Associate Professor with the Faculty of Computer Science and Information Technology, UPM. Her research interests include agile software development, software requirement engineering, software measurement, and empirical software engineering. She is a Committee Member of Malaysian Standard for NSC 07/TC 11.

ABU BAKAR MD SULTAN received the B.Sc. degree from the National University of Malaysia (UKM), and the master's and Ph.D. degrees in software engineering from Universiti Putra Malaysia (UPM). He is a Professor with the Faculty of Computer Science and Information Technology, UPM. He has published many journals in the field of software engineering. His current research focuses on software security and empirical software engineering. His research projects are supported by a number of Malaysian government research funding agencies.