


Recommendation of Machine Learning Techniques for Software Effort Estimation using Multi-Criteria Decision Making

Ajay Kumar

(Department of Information Technology, KIET Group of Institutions,
Delhi-NCR, Ghaziabad-201206, India

 <https://orcid.org/0000-0003-0126-7172>, ajaygarg100@gmail.com)

Abstract: For the development of the software industry, Software Effort Estimation (SEE) is one of the essential tasks. Project managers can overcome budget and time overrun issues by accurately estimating a software project's development effort in the software life cycle. In prior studies, a variety of machine learning methods for SEE modeling were applied. The outcomes for various performance or accuracy measures are inconclusive. Therefore, a mechanism for assessing machine learning approaches for SEE modeling in the context of several contradictory accuracy measures is desperately needed. This study addresses selecting the most appropriate machine learning technique for SEE modeling as a Multi-Criteria Decision Making (MCDM) problem. The machine learning techniques are selected through a novel approach based on MCDM. In the proposed approach, three MCDM methods- Weighted Aggregated Sum Product Assessment (WASPAS), Technique for Order Preference by Similarity to Ideal Solution (TOPSIS), and ViseKriterijumska Optimizacija I Kompromisno Resenje (VIKOR) were applied to determine the ranking of machine learning techniques on SEE performance based on multiple conflicting accuracy measures. For validating the proposed method, an experimental study was conducted over three SEE datasets using ten machine-learning techniques and six performance measures. Based on MCDM rankings, Random Forest, Support Vector Regression, and Kstar are recommended as the most appropriate machine learning techniques for SEE modeling. The results show how effectively the suggested MCDM-based approach can be used to recommend the appropriate machine learning technique for SEE modeling while considering various competing accuracy or performance measures altogether.

Keywords: Software Effort Estimation (SEE); Multi-Criteria Decision Making (MCDM); WASPAS; TOPSIS; VIKOR

Category: D.2, D.2.9

DOI: 10.3897/jucs.110051

1 Introduction

In the process of development of software, the goal of estimating effort is to estimate the labor and time required to complete a project successfully within the specified budget and time [Ali and Gravino 2019]. Usually, SEE is expressed in terms of person-hours or person-months. Because of the continuously increasing need for complex and large-scale software systems, managers have identified SEE as one of the essential tasks directly linked to the failure or success of the entire software development process. Accurate estimation of a software project's development effort assists project managers in overcoming budget and time overrun issues. Because of several problems in

achieving reliable SEE results, software effort estimation research has been ongoing since the 1960s.

Initially, expert judgment and analogy-based methods were used to perform software effort estimation. By consulting experts who have previously finished similar projects, the effort estimate is done in the expert judgment method. When estimating by analogy, the current projects are compared to previously finished similar projects for estimating software efforts. Later, researchers were interested in using machine learning models for estimating software effort, which has been widely utilized since 1991 [Ali and Gravino 2019].

[Rijwani and Jain 2016] proposed the usage of ANNs for SEE modeling. They conducted an experimental study using three software effort datasets to validate the proposed model. [Tayyab et al. 2018] developed a model for estimating software efforts using the multilayer perceptron technique. [Arslan 2019] reviewed machine learning techniques for software effort estimation. The authors used thirteen machine learning techniques over two software effort datasets in the study to develop SEE models. Machine learning models for software effort estimation have been thoroughly investigated and found useful for software project management as per the study conducted by [Varshini et al. 2021]. [Akhbardeh and Reza 2021] focused on the difficulties of software effort estimation using traditional approaches. For constructing SEE models, they emphasized the usage of machine learning techniques. [Mahmood et al. 2022] conducted a review study to examine several machine learning algorithms for modeling software effort estimation. Apart from these, described above, various researchers have spent a lot of effort demonstrating the use and efficiency of machine learning techniques for software effort estimation (as described in detail in section 2).

The predictive capabilities of machine learning approaches vary significantly for various performance measures across software effort datasets of different software systems. No single predictive model exists for a particular application domain that performs better than other models considering all performance measures as per the No Free Lunch Theorem [Wolpert and Macready 1995]. Thus, it becomes challenging for a software project manager to choose which machine learning technique to employ for SEE modeling. So, it is necessary to propose a method for selecting the most appropriate machine learning technique for SEE modeling, considering various performance measures.

This work suggests a novel MCDM-based approach for recommending the best appropriate machine learning technique for SEE modeling, considering many competing performance or accuracy measures. MCDM is one of the most well-known subfields of decision-making. MCDM finds the most suitable choice among the many possibilities accessible, taking into account multiple competing criteria [Thakkar 2022]. Distinct criteria may conflict with one another since they signify different aspects of the options. For instance, cost and profit may conflict, etc. As multiple contradictory accuracy or performance measures are involved in assessing a machine learning technique, the problem of choosing the most appropriate approach for SEE modeling can be formulated as an MCDM problem.

The machine learning techniques are selected through a novel MCDM-based approach. In the proposed approach, three MCDM methods (WASPAS, VIKOR, and TOPSIS) were used to determine ranking scores of machine learning techniques on SEE performance based on multiple conflicting accuracy measures. The following is a summary of this paper's key contribution.

- This research proposes an MCDM-based method for recommending machine learning techniques for developing models for software effort estimation considering several conflicting accuracy measures altogether.
- In the proposed approach, three MCDM methods- WASPAS, VIKOR, and TOPSIS were employed to determine the individual ranking score of machine learning techniques for SEE modeling. Next, the machine learning technique, ranked first by all three MCDM methods, is recommended as the best machine learning technique for SEE modeling.
- This study conducts an experimental study to validate the proposed approach using ten machine learning techniques and six conflicting performance measures over three software effort datasets (open-source).

The following is how the remaining part of the paper is organized. Section 2 describes the relevant work, whereas Section 3 describes the proposed method, selected machine learning techniques, MCDM methods, and performance measures used in this study. The datasets used in this paper and the experimental procedure used to validate the proposed approach are discussed in section 4. The results, discussion, and the MCDM-based ranking of machine learning techniques are presented in Section 5. Section 6 presents the practical implications of the study, Section 7 concludes the paper, and future research directions are presented in section 8.

2 Related Work

This section highlights previous research on SEE modeling by various researchers. [Seo et al. 2013] developed a model for software effort estimation using multiple linear regression with recursive data partitioning. [Pai et al. 2013] focused on using Artificial Neural networks (ANN) in software effort estimation modeling. [Fedotova et al. 2013] conducted a review study on Multiple Linear Regression (MLR) in software effort estimation modeling. They compared the software effort estimation capability of MLR with that of the traditional expert judgment method in terms of performance metrics MMRE and PRED (0.25). They concluded that MLR-based models are superior at estimating software effort than the standard expert judgment method.

[Anandhi and Chezian 2014] developed two software effort estimation models using machine learning techniques M5 and linear regression. The COCOMO dataset was used in the experimental study. They determined that the M5 algorithm outperformed the linear regression based on the results of two performance metrics, MMRE and MdMRE. [Satapathy et al. 2016] employed the random forest for estimating software development efforts. They compared the software effort prediction capability of random forest with that of the other four machine learning techniques, namely, Stochastic Gradient Boosting (SGB), Radial Base Function Network (RBFN), Multi-layer Perceptron (MLP), and Log-linear regression (LLR) based on two performance measures MMRE and PRED. For the software effort estimation, [Sharma and Singh 2018] employed three machine learning techniques: multilayer perceptron, random forest, and support vector machines. These three machine-learning techniques were examined on four software effort datasets. After analyzing the results, the authors conclude that random forest is better than the other two techniques.

[Amaral et al. 2019] developed a model for estimating software efforts using the decision tree. They evaluated the proposed model over four software effort datasets in terms of two performance measures PRED (0.25) and MMRE. [Sakhrawi et al. 2020] used four machine-learning techniques for developing SEE models. Four machine learning techniques are Gradient Boosting Regressor (GBR), SVR, Ada Boost Regressor, and random forest. They compared the results of software effort predictions produced by these four machine learning techniques for two performance measures, MAE and MSE. The results of the experimental study show that random forest outperforms the other three machine learning techniques.

[Sakhrawi et al. 2021] et al., in their survey study, underlined the importance of choosing the most appropriate machine learning technique for estimating software effort among various available machine learning techniques. They reviewed roughly thirty research publications on machine learning-based software effort estimation models that were published between 1995 and 2020. [Goyal 2021] conducts an empirical study for the evaluation of machine learning techniques for software effort estimation. The author compares the performance of generalized linear regression, support vector regressor, and artificial neural network for estimating software efforts on five software effort datasets. The author uses the two performance measures namely, MMRE and mean absolute error (MAR) for the comparative study. However, both accuracy measures were used separately.

[Mahmood et al. 2022] conducted a review study to examine several machine learning algorithms for modeling software effort estimation. They focused on single machine learning techniques and a machine learning-based ensemble strategy for predicting software effort in the study. The study included two parts: first, they used machine learning techniques to investigate state-of-the-art in the domain of SEE modeling. Second, they employed the commonly known performance measurements MMRE and PRED (0.25) to assess various machine learning algorithms.

[Sharma and Vijayvargiya 2022] perform a comparative study to evaluate the software effort estimation performance of four soft computing-based models- GEHO-based NFN (GEHO-NFN), Adaptive GA-based neural network (AGANN), Neuro-fuzzy logic (NFL), and Localized Neighborhood Mutual Information based neural network (LNI-NN). Five different datasets, including the promise database's cocomo81, cocomonasa1, and cocomonasa2 were used to test and validate each of the four modelling approaches. Four performance measures- RMSE, PRED, MdMMRE, and MMRE were used for the comparative study. However, they have not considered four performance measures altogether.

[Kumar and Srinivas 2023] suggest a model namely, accurate analogy-based software effort estimation model (AA-SEE) based on machine learning techniques and hybrid optimization to further improve effort estimate. They evaluate their proposed model through different software effort datasets using various performance measures such as MMER, MMRE, MdMMRE, and MdMMER. However, they have considered one performance measure one at a time. [Abnane et al. 2023] proposed a model for improving software effort estimation by replacing missing values using ensemble imputation techniques. They evaluate the proposed model on various datasets using five performance metrics- logarithmic standard deviation (LSD), mean inverted balanced relative error (MIBRE), mean balanced relative error (MBRE), Pred (0.25), and standardized accuracy (SA). However, they have not considered these five performance measures taken into account altogether. [Sanchez et al. 2023] developed software effort

estimation models using four machine learning techniques- ensemble learning, adaboost, random forest, and decision tree. They compared the performance of these four machine learning techniques for software effort estimation using the performance measures- MSE, MMRE, MdMRE, and Pred (0.25). However, they have not considered these four performance measures altogether.

After a thorough review of research done in software effort estimation modeling, the following observations can be concluded.

- Various researchers have spent a lot of effort demonstrating the use and efficiency of machine learning techniques for software effort estimation.
- In some cases, researchers have used only single performance measures to evaluate the machine learning techniques. Some studies consider multiple performance measures to evaluate machine learning techniques for software effort estimation, but these studies do not consider the simultaneous optimization of all performance measures.
- In previous studies, no one has emphasized selecting the most suitable machine learning technique for software effort estimation modeling considering various performance measures taken into account altogether using MCDM approach.

This paper proposes an MCDM based approach for recommending most suitable machine learning technique for software effort estimation modeling by taking into account more than one conflicting performance measure considering simultaneous optimization of all performance measures. To the best of the authors' knowledge, no other research has considered the problem of evaluating machine learning techniques for SEE modeling in the presence of more than one conflicting performance or accuracy measure using the MCDM approach.

3 Research Methodology

This section describes the proposed method, selected machine learning techniques, performance measures, and MCDM methods used in this study.

3.1 Proposed Method

This paper suggests an MCDM-based method for recommending the best machine learning technique for estimating software effort in the presence of many competing accuracy measures. The proposed approach used three MCDM methods to rank ten machine learning techniques on SEE performance considering six performance measures. An experimental study was conducted over three open-source SEE datasets to validate the proposed approach. Fig. 1 provides an overview of the proposed method.

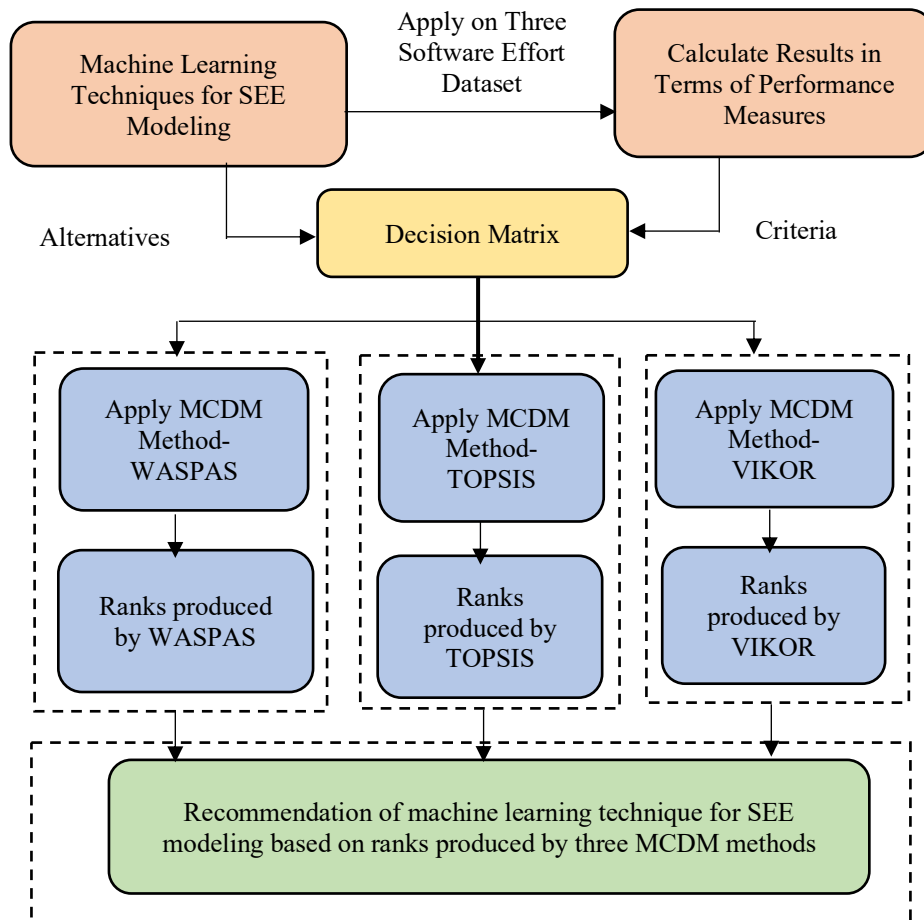


Figure 1: The process of generating a ranking index for SEE models using the MCDM-based approach

3.2 Machine Learning Techniques

Considering a large number of machine learning techniques, it is not possible to take all machine learning techniques into consideration for the validation of the proposed approach. This study has applied ten machine learning techniques used in previous studies (as discussed in the related work section) for building SEE models. These machine learning techniques are Multiple Linear Regression (MLR), Isotonic Regression (IR), Pace Regression (PR), K-Nearest Neighbors (KNN), Support Vector Regression (SVR), KSTAR, Decision Table (DT), M5Rules, M5P, and Random Forests (RF).

3.3 Performance Measures

This study chooses six performance measures to evaluate machine learning techniques for SEE modeling. Depending on the type of performance measure, the MCDM methodology classifies these six performance measures as six conflicting criteria (cost criteria or benefits criteria). A performance measure for which a minimum value is desired can be considered a cost criterion. On the other hand, performance measures where the maximum value is desired can be considered as benefit criteria. All six performance measures are listed in Table 1, followed by a brief description of each performance measure.

Performance Measures (as criteria)	Abbreviation	Type of Criteria
Root Mean Square Error	RMSE	Cost Criteria
Mean Magnitude of Relative Error	MMRE	Cost Criteria
Median of Absolute Residual Error	MDARE	Cost Criteria
Mean Balanced Relative Error	MBRE	Cost Criteria
Pearson Correlation Coefficient	r	Benefit Criteria
PRED (0.25)	PRED (0.25)	Benefit Criteria

Table 1: Performance Measures

Given n is the total number of observations. For i^{th} observation, u_i represents actual effort, and v_i represents estimated effort. All six performance measures can be explained as follows:

- RMSE represents the root mean square error and can be calculated as follows:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^n (u_i - v_i)^2}{n}} \quad (1)$$

- MMRE [Menziez 2014] represents the mean value of the magnitude of relative error and can be calculated as follows:

$$\text{MMRE} = \frac{1}{n} \sum_{i=1}^n \frac{|u_i - v_i|}{u_i} \quad (2)$$

- MDARE is the median value of all absolute residual error (ARE i), where ARE i can be calculated as follows:

$$\text{ARE}_i = |u_i - v_i| \quad (3)$$

- MBRE [Kumar 2023] represents the mean value of balanced relative error and can be calculated as:

$$\text{MBRE} = \frac{1}{n} \sum_{i=1}^n \frac{|u_i - v_i|}{\min(u_i, v_i)} \quad (4)$$

- Pearson Correlation Coefficient (r) [Sheskin 2011] measures the linear correlation between actual values and predicted values. This performance measure can be calculated as follows:

$$r = \frac{n(\sum_{i=1}^n u_i v_i) - (\sum_{i=1}^n u_i)(\sum_{i=1}^n v_i)}{\sqrt{\left[n \sum_{i=1}^n u_i^2 - (\sum_{i=1}^n u_i)^2 \right] \left[n \sum_{i=1}^n v_i^2 - (\sum_{i=1}^n v_i)^2 \right]}} \quad (5)$$

- PRED (0.25) [Menzies 2014] represents the fraction of predicted values with a magnitude of relative error (MRE) less than 0.25. Where MRE can be calculated as follows:

$$\text{MRE}_i = \frac{|u_i - v_i|}{u_i} \quad (6)$$

3.4 MCDM Methods

When decisions must be made based on conflicting criteria, a variety of MCDM methods are available. There are benefits and drawbacks to every MCDM method. There is currently no approach exist that permits a specific MCDM method to be selected. Many MCDM methods will produce a more trustworthy ranking of machine learning techniques than a single MCDM method. In this study, we have chosen three MCDM approaches for evaluating machine learning models in SEE modeling: WASPAS, TOPSIS, and VIKOR.

3.4.1 WASPAS

WASPAS method [Zavadskas et al. 2012] combines the results of two different MCDM methods, namely the weighted product model (WPM) and the weighted sum model (WSM). A detailed stepwise description of WASPAS method is given below.

Step1: Construction of Decision Matrix as the input for WASPAS method.

Prepare a decision matrix, $D_{a \times c}$; here, a denotes the number of machine learning techniques for SEE modeling as alternatives, and c represents the number of performance measures as criteria. In this research, the value of c is six, and the value of a is ten. In the matrix $D_{a \times c}$, each entry d_{ij} denotes the value of the j^{th} accuracy measure for the corresponding i^{th} machine learning technique.

Step2: Normalization of Decision Matrix

Calculate the normalized value of each entry d_{ij} of the decision matrix $D_{a \times c}$, using Eq. (7) and Eq. (8).

(For Beneficiary criteria, where the maximum value is desired):

$$d_{ij}^* = \frac{d_{ij}}{\max_i d_{ij}}; i = 1 \text{ to } a, j = 1 \text{ to } c \text{ if criteria } j \text{ is benefit criteria} \quad (7)$$

(For Cost criteria, where the minimum value is desired):

$$d_{ij}^* = \frac{\min_i d_{ij}}{d_{ij}}; i = 1 \text{ to } a, j = 1 \text{ to } c \text{ if criteria } j \text{ is cost criteria} \quad (8)$$

Step3: Compute the WSM factor (Q_i^{wsm}) and WPM factor (Q_i^{wpm}) for each alternative using Eq. (9) and Eq. (10), respectively.

$$Q_i^{wsm} = \sum_{j=1}^c d_{ij}^* w_j ; i = 1 \text{ to } a \quad (9)$$

$$Q_i^{wpm} = \prod_{j=1}^c (d_{ij}^*)^{w_j} ; i = 1 \text{ to } a \quad (10)$$

Where $w_j = [w_1, w_2, \dots, w_c]$ represents the weights of performance measures. In this study, we have considered equal weightage of each criterion (performance measure).

Step4: Compute the Aggregation measure Q_i for each alternative using Eq. (11).

$$Q_i = \frac{(Q_i^{wsm} + Q_i^{wpm})}{2}; i = 1, 2, \dots, a \quad (11)$$

Step5: Rank the alternatives in decreasing order of aggregation measure. The higher the value of the aggregation measure higher will be the rank.

3.4.2 TOPSIS

TOPSIS [Hwang and Yoon 1981] is a well-known MCDM technique for rating available methods to address a decision problem with competing criteria. This technique chooses the alternative that is closest to the ideal alternative. An ideal alternative is defined as the alternative with the best possible criterion value. It is the Euclidean distance that is utilized for distance measure. Below is a step-by-step procedure.

Step1: Construction of Decision Matrix as the input for TOPSIS method.

[Description] same as described in step1 of previous section 3.4.1.

Step2: Normalized decision matrix $V_{a \times c}$ is obtained by using Eq. (12); here, each entry v_{ij} denotes the normalized value of d_{ij} .

$$v_{ij} = \frac{d_{ij}}{\sqrt{\sum_{i=1}^a d_{ij}^2}} ; j=1,2,\dots,c \quad (12)$$

Step3: Weighted normalized decision matrix $T_{a \times c}$ is obtained by using Eq. (13), where t_{ij} represents the weighted normalized value of d_{ij} .

$$t_{ij} = v_{ij} \times w_j \quad (13)$$

Here w_j is the weight assigned to criteria j . The values of weights are the same as in the case of WASPAS.

Step4: Calculate Ideal Solutions $[PIS]_{c \times 1}$ and $[NIS]_{c \times 1}$

The best value each criterion may achieve is determined as the positive ideal solution (PIS). The least/worst value each criterion can achieve is used to calculate the negative ideal solution (NIS). They can be calculated using Eq (14) and Eq. (15).

$$PIS = \left\{ (\max T_{ij} / j \in z), (\min T_{ij} / j \in z') \text{ for } i=1,2,\dots,a \right\} \quad (14)$$

$$= \{T_1^+, T_2^+, T_3^+, \dots, T_c^+\}$$

$$NIS = \left\{ (\min T_{ij} / j \in z), (\max T_{ij} / j \in z') \text{ for } i=1,2,\dots,a \right\} \quad (15)$$

$$= \{T_1^-, T_2^-, T_3^-, \dots, T_c^-\}$$

Where z is related to beneficial criteria and z' is related to cost criteria.

Step5: Euclidean Distance

For each alternative Euclidean distance ED^+ from PIS and Euclidean distance ED^- from NIS are calculated using (16) and (17), respectively.

$$ED_i^+ = \left\{ \sqrt{\sum_{j=1}^c (T_{ij} - T_j^+)^2} \text{ for } i=1 \text{ to } a \right\} \quad (16)$$

$$ED_i^- = \left\{ \sqrt{\sum_{j=1}^c (T_{ij} - T_j^-)^2} \text{ for } i=1 \text{ to } a \right\} \quad (17)$$

Step6: Find Relative Closeness $[RC]_{a \times 1}$

Relative closeness for each alternative with respect to the NIS and PIS can be calculated using Eq. (18).

$$RC_i = \frac{ED_i^-}{(ED_i^- + ED_i^+)} \quad (18)$$

Step7: Selection of the best alternative

Rank the alternatives (in this study, machine learning techniques for SEE modeling) according to the value of relative closeness obtained in step 6. The machine learning technique with the highest value of relative closeness (RC) will be recommended as the most appropriate alternative.

3.4.3 VIKOR

VIKOR is one of the most popular MCDM methods. This method produces the ranking index of alternatives based on a particular measure of closeness to the ideal solution in the presence of conflicting criteria. Following is the detailed procedure of the VIKOR method [Opricovic and Tzeng 2004].

Step1: Construction of Decision Matrix as the input for VIKOR method.

[Description] same as described in step1 of WASPAS and TOPSIS in previous sections 3.4.1 and 3.4.2, respectively.

Step2: Find the best d_j^+ and worst d_j^- values for each criterion by using the following formula.

$$d_j^+ = \max_i d_{ij}, d_j^- = \min_i d_{ij}; i = 1 \text{ to } a, j = 1 \text{ to } c\}, \text{ For benefit criteria} \quad (19)$$

$$d_j^+ = \min_i d_{ij}, d_j^- = \max_i d_{ij}; i = 1 \text{ to } a, j = 1 \text{ to } c\}, \text{ For cost criteria} \quad (20)$$

Step3: The utility measure S_i and regret measure R_i can be computed as follows:

$$S_i = \sum_{j=1}^c \frac{w_j(d_j^+ - d_{ij})}{(d_j^+ - d_j^-)}; i = 1 \text{ to } a, j = 1 \text{ to } c \quad (21)$$

$$R_i = \max_j \left[\frac{w_j(d_j^+ - d_{ij})}{(d_j^+ - d_j^-)} \right]; i = 1 \text{ to } a, j = 1 \text{ to } c \quad (22)$$

Where w_j is the weight of criteria j . The values of weights are the same as in the case of WASPAS and TOPSIS.

Step4: Calculate the values (S^*, S^-) and (R^*, R^-) by using the following relations.

$$S^* = \min_i S_i, S^- = \max_i S_i; i = 1 \text{ to } a \quad (23)$$

$$R^* = \min_i R_i, R^- = \max_i R_i; i = 1 \text{ to } a \quad (24)$$

Step5: Now compute the value of VIKOR index Q_i for each alternative as follows:

$$Q_i = \frac{1}{2} \left[\frac{(S_i - S^*)}{(S^- - S^*)} + \frac{(R_i - R^*)}{(R^- - R^*)} \right]; i = 1, 2, \dots, a \quad (25)$$

Step6: Rank the alternatives in order of Q_i value, with a smaller Q_i value indicating a higher rank.

4 Experimental Study

This section is further divided into two subsections. Subsection 4.1 discusses the brief description of software effort datasets used in this study. Subsection 4.2 presents the detailed experimental procedure for validating the proposed method, as described in section 3.1.

4.1 SEE Datasets

In this study, we used three datasets easily available in the public domain and have been widely used in previous studies [Ali and Gravino 2019, Rijwani and Jain 2016, Anandhi and Chezian 2014, Satapathy et al. 2016, Sharma and Singh 2018, Arslan 2019, Sakhravi et al. 2020, Sharma and Vijayvargiya 2022] for software effort estimation modeling. Three SEE datasets are: cocomo81, cocomonasa1, and cocomonasa2, taken from the software engineering data repository PROMISE [Promise Repository]. Details of the datasets are as follows:

- The cocomo81 dataset contains information about 63 software projects. This dataset has seventeen attributes. All seventeen attributes are numeric, in which fifteen attributes are the effort multiplier, one attribute is lines of code (LOC), and one attribute is the actual effort (dependent variable).
- The cocomonasa1 dataset contains information about 60 software projects. This dataset also has seventeen attributes (all numeric), including fifteen effort multiplier attributes, one attribute for LOC, and one is the dependent variable (actual effort).
- The cocomonasa2 dataset contains information about 93 software projects. This dataset has twenty-four attributes. Fifteen attributes are standard discrete effort multipliers, seven attributes describe the projects, one attribute is for LOC, and one is the actual effort (dependent variable).

4.2 Experimental Design

Machine learning techniques create SEE models based on the data describing projects completed in the past. In this study, we made the assumption that historical datasets of previously completed projects at a software company are available to estimate the software efforts for the new project.

The following procedure is used for experimental design.

Input: Three software effort estimation datasets as described in the previous section (section 4.1).

Output: Ranking Index for machine learning techniques for SEE modeling.

Step 1: Three datasets described in section 4.1 are preprocessed to select the relevant features. Feature selection is a preprocessing operation used to find and eliminate unnecessary and irrelevant data from datasets. Correlation-based Feature Selection (CFS) [Hall 1999], a well-known feature selection algorithm, is used in this study.

According to the CFS process, a feature is advantageous if it has a strong correlation with the dependent variable but not with other features.

Step 2: Apply ten machine learning techniques as described in section 3.2 to build ten SEE models for each dataset. Open-source tool Weka version 3.8.3 [Hall et al. 2009] was used to implement these ten machine-learning techniques.

Step 3: Results of six performance measures are obtained for ten machine learning techniques used for SEE modeling. The results for each dataset are stored in a 10×6 matrix.

Step 4: Use the 10×6 matrix for each dataset obtained from step 3 as the decision matrix to apply three MCDM methods, WASPAS, TOPSIS, and VIKOR.

Step 5: A 10×1 matrix is obtained as the output of each MCDM method representing the ranking score of ten machine learning techniques for SEE modeling for each dataset.

Step 6: Recommend the machine learning technique for SEE modeling based on the rank produced by all three MCDM methods.

The graphical representation of the experimental design is shown in Fig. 2.

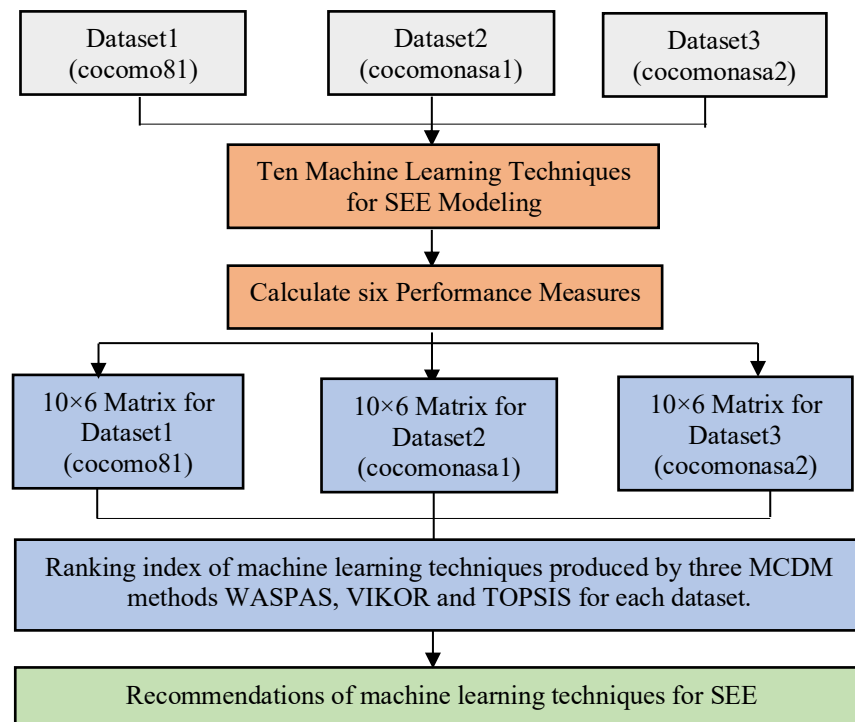


Figure 2: Graphical Representation of the Experimental Study

5 Results and Discussion

There are two subparts to this section. In the first part, we review the software effort estimation results of ten machine-learning techniques for each dataset. In the second part, the proposed MCDM-based method ranks the machine learning techniques for SEE modeling.

5.1 SEE Results

Table 2 to Table 4 display the results of the ten machine learning techniques for SEE modeling in terms of the six performance measures as described in section 3.3.

Machine Learning Technique	RMSE	MMRE	MDARE	MBRE	r	PRED (0.25)
MLR	0.2144	6.8414	0.1070	2.4122	0.7979	0.2459
IR	0.2557	2.6348	0.0690	3.3705	0.7048	0.2295
PR	0.2095	6.8142	0.1140	2.3206	0.8081	0.2459
SVR	0.2183	3.4522	0.0770	1.7052	0.7898	0.2131
KNN	0.2964	2.6744	0.0790	4.6908	0.6590	0.0984
KSTAR	0.2524	1.9772	0.0600	2.3773	0.7291	0.1639
DT	0.3329	2.3050	0.1280	2.9554	0.4554	0.2131
M5Rules	0.2144	6.8414	0.1070	2.4122	0.7979	0.2459
M5P	0.2144	6.8414	0.1070	2.4122	0.7979	0.2459
RF	0.2299	2.1056	0.0670	2.3966	0.7629	0.2295

Table 2: Results of Dataset1 (cocomo81)

Machine Learning Technique	RMSE	MMRE	MDARE	MBRE	r	PRED (0.25)
MLR	0.1095	0.4476	0.0450	0.5226	0.9466	0.4915
IR	0.1377	0.4191	0.0290	0.5200	0.9160	0.5254
PR	0.1092	0.4455	0.0430	0.4987	0.9468	0.4746
SVR	0.1055	0.3235	0.0280	0.3619	0.9511	0.5593
KNN	0.1546	0.3606	0.0280	0.4798	0.8939	0.5593
KSTAR	0.1424	0.4551	0.0270	0.5364	0.9292	0.4576
DT	0.1756	0.8435	0.0340	1.0376	0.8636	0.4237
M5Rules	0.1078	0.4207	0.0390	0.4322	0.9482	0.4746
M5P	0.1075	0.4180	0.0450	0.4279	0.9486	0.4915
RF	0.1125	0.3898	0.0310	0.4201	0.9459	0.5932

Table 3: Results of Dataset2 (cocomonasa1)

Machine Learning Technique	RMSE	MMRE	MDARE	MBRE	r	PRED (0.25)
MLR	0.1917	2.0065	0.0670	1.8013	0.8170	0.2967
IR	0.2617	1.1522	0.1140	1.6820	0.6290	0.2088
PR	0.1924	2.0150	0.0670	1.8301	0.8156	0.2857
SVR	0.2105	1.6595	0.0970	1.7831	0.7753	0.3736
KNN	0.2323	2.5644	0.0430	4.6168	0.7514	0.3956
KSTAR	0.2183	1.3567	0.0730	1.6331	0.7889	0.3626
DT	0.2638	1.6885	0.0620	2.0522	0.6367	0.3187
M5Rules	0.1997	1.9831	0.0780	1.7468	0.8003	0.2747
M5P	0.1983	1.9870	0.0830	1.7550	0.8031	0.2747
RF	0.2229	2.1609	0.0510	2.5223	0.7520	0.3187

Table 4: Results of Dataset3 (cocomonasa2)

- Table 2 shows that for the cocomo81 dataset, the SEE model PR performs the top for the accuracy measures PRED (0.25), r , and RMSE. For performance measures, MMRE and MDARE, SEE model KSTAR performs the best. SEE model SVR is best for performance measure MBRE. The most appropriate SEE models for accuracy measure PRED (0.25) are M5P, M5Rules, and MLR.
- Table 3 shows that, for the cocomonasa1 dataset, the SEE model SVR performs the top for accuracy measures MBRE, RMSE, r , and MMRE. For performance measure, MDARE, SEE model KSTAR is the best. For accuracy measure PRED (0.25), SEE model RF outperforms all other SEE models.
- Table 4 shows that, for the cocomonasa2 dataset, the SEE model MLR is most appropriate for performance measures r and RMSE. For performance measure, MMRE, SEE model IR is the best. SEE model KNN is most appropriate for performance measures MDARE and PRED (0.25). For performance measure, MBRE, SEE model KSTAR is the best.

According to the above findings, no one machine learning technique for SEE modeling can be chosen as the optimal machine learning technique for any dataset when six performance measures are considered together. Consequently, this motivates us to evaluate machine learning techniques for software effort estimation considering several performance metrics using the MCDM-based approach.

5.2 MCDM Ranking

Three MCDM methods, WASPAS, VIKOR, and TOPSIS (as described in section 3.4), are used for generating the ranking index for ten machine learning techniques for software effort estimation considering multiple conflicting performance measures. Table 5-7 shows the ranking index for machine learning techniques for SEE modeling for three datasets cocomo81, cocomonasa1, and cocomonasa2.

Machine Learning Techniques	Ranking Generated by Three MCDM Methods					
	WASPAS		TOPSIS		VIKOR	
	WASPAS-Score	WASPAS-Rank	TOPSIS-Score	TOPSIS-Rank	VIKOR-Score	VIKOR-Rank
MLR	0.7231	6	0.5213	6	0.7092	6
IR	0.7844	4	0.6793	4	0.3547	4
PR	0.7265	5	0.5210	9	0.7052	5
SVR	0.8519	2	0.8054	2	0.0693	2
KNN	0.6159	10	0.4419	10	0.9730	9
KSTAR	0.8477	3	0.7660	3	0.3108	3
DT	0.6522	9	0.5307	5	1.0000	10
M5Rules	0.7231	6	0.5213	6	0.7092	6
M5P	0.7231	6	0.5213	6	0.7092	6
RF	0.8870	1	0.8529	1	0.0000	1

Table 5: Ranking index of machine learning techniques for SEE modeling for dataset cocomo81

Machine Learning Techniques	Ranking Generated by Three MCDM Methods					
	WASPAS		TOPSIS		VIKOR	
	WASPAS-Score	WASPAS-Rank	TOPSIS-Score	TOPSIS-Rank	VIKOR-Score	VIKOR-Rank
MLR	0.7940	9	0.6702	9	0.6879	9
IR	0.8328	5	0.7569	5	0.3115	3
PR	0.8015	8	0.6921	8	0.6129	7
SVR	0.9844	1	0.9617	1	0.0000	1
KNN	0.8599	3	0.7726	3	0.4686	5
KSTAR	0.8077	7	0.7013	7	0.5535	6
DT	0.6065	10	0.1481	10	1.0000	10
M5Rules	0.8422	4	0.7618	4	0.4553	4
M5P	0.8321	6	0.7207	6	0.6637	8
RF	0.9146	2	0.8838	2	0.0470	2

Table 6: Ranking index of machine learning techniques for SEE modeling for dataset cocomonasal

Machine Learning Techniques	Ranking Generated by Three MCDM Methods					
	WASPAS		TOPSIS		VIKOR	
	WASPAS-Score	WASPAS-Rank	TOPSIS-Score	TOPSIS-Rank	VIKOR-Score	VIKOR-Rank
MLR	0.8031	2	0.6996	2	0.2068	2
IR	0.7104	10	0.5561	9	1.0000	10
PR	0.7947	4	0.6896	4	0.2289	3
SVR	0.7962	3	0.6556	6	0.3859	6
KNN	0.7305	9	0.4278	10	0.8027	8
KSTAR	0.8604	1	0.7884	1	0.0000	1
DT	0.7464	8	0.6935	3	0.8479	9
M5Rules	0.7727	5	0.6575	5	0.3237	4
M5P	0.7666	6	0.6411	8	0.3314	5
RF	0.7618	7	0.6443	7	0.4458	7

Table 7: Ranking index of machine learning techniques for SEE modeling for dataset cocomonasa2

The following inferences are drawn from Table 5-7.

- For the dataset cocomo81, it is observed that Random Forest (RF) is the most appropriate machine learning technique for SEE modeling after optimization of all conflicting accuracy measures.
- Support Vector Regression (SVR) is suggested as the most appropriate machine learning technique for SEE modeling on the cocomonasa1 data set after optimization of all competing performance measures.
- After taking into account and optimizing all conflicting accuracy measures, it is found that KSTAR is the most appropriate machine learning technique for SEE modeling on the cocomonasa2 dataset.

From the above inferences, it can be observed that the advantage of applying the proposed MCDM-based approach is that the proposed method can be used as an efficient tool for selecting the most appropriate machine learning techniques for SEE modeling by optimizing various contradictory performance measures taken into account altogether. Although most of the researchers in previous studies [Varshani et al. 2021, Fedotova et al. 2013, Anandhi and Chezian 2014, Sharma and Singh 2018, Arslan 2019, Sakhrawi et al. 2020, Sharma and Vijayvargiya 2022, Kumar and Srinivas 2023] took into account multiple performance measures to assess the various available SEE models, it can be observed that in their study, they presented the best SEE model by simply taking into consideration one performance measure at a time.

6 Practical Implication

Successful project management involves precise software development effort estimation. Several types of machine learning techniques for estimating software development efforts have been examined in prior studies. There may be inconsistencies in the software effort estimation modeling performance of machine learning techniques. As a result, the choice of the optimum machine learning technique becomes challenging for a software manager since different machine learning techniques have inconsistent prediction capability with respect to multiple performance measures. If a software manager has a reason for emphasizing a particular performance measure, he/she may choose the machine learning technique more likely to perform best for this measure. In contrast, if it is unclear which performance measure should be emphasized, he/she can be interested in a solution that offers a decent trade-off between many performance measures. The proposed MCDM-based method provides a unique framework to aid the software manager in selecting the most suitable machine learning technique for SEE modeling, considering various performance measures and taking them into account altogether.

7 Conclusion

Many studies have focused on developing various models for accurate software effort estimation using machine learning approaches, as described in the related work section of this paper. However, the selection of the most appropriate machine-learning technique for SEE is still in the infant stage. In this study, we propose an MCDM-based framework to address the issue of selecting a machine-learning technique for SEE modeling using MCDM. To validate the proposed approach, we have chosen three datasets from PROMISE repository, which are freely available and widely used by various researchers in the past for SEE modeling (described in detail in section 4.1).

In the proposed method, the values of six performance metrics for ten machine learning techniques on each dataset are obtained first. For any dataset, according to the results of six performance metrics for the applied machine learning techniques, as discussed in section 5.1, no machine learning technique can be recommended considering all performance metrics. Consequently, it is necessary to evaluate machine learning techniques by optimizing all six-performance metrics. In the proposed approach, three MCDM methods are implemented to produce individual ranking scores of machine learning techniques based on six performance metrics. Three datasets were used for analysis to validate the proposed method. Experimental results indicate that machine learning technique RF is best suited for the cocomo81 dataset, machine learning technique SVR is the most appropriate for the cocomonasa1 dataset, and machine learning technique KSTAR is best suited for the cocomonasa2 dataset.

8 Future Research Direction

As an extension, the proposed MCDM-based approach can be used to recommend the most appropriate machine learning technique for solving various types of software cost and quality prediction problems in software engineering, such as software defect prediction problems, etc. Furthermore, the proposed method can be extended to solve

various types of decision-making problems in other domains related to software engineering. For example, during the various stages of the software development life cycle, the proposed approach may be used to choose the best software testing technique from among the various testing approaches that are available. As the proposed approach is based on the MCDM methods, using hybrid MCDM methods may be another future work for solving various decision problems in the context of software engineering. This study can also be further extended to a large number of software effort estimation datasets.

References

- [Abnane et al. 2023] Abnane, I., Idri, A., Chlioui, L., Abran, A.: "Evaluating ensemble imputation in software effort estimation"; *Empirical Software Engineering*, 28, 2, (2023).
- [Akhbardeh and Reza 2021] Akhbardeh, F., Reza, H.: "A Survey of Machine Learning Approach to Software Cost Estimation"; Proc. International Conference on Electro Information Technology (EIT), IEEE (2021), 405-408.
- [Ali and Gravino 2019] Ali, A., Gravino, C.: "A systematic literature review of software effort prediction using machine learning methods"; *Journal of software: evolution and process*, 31, 10 (2019).
- [Amaral et al. 2019] Amaral, W., Rivero, L., Junior, G. B., Viana, D.: "Using Machine Learning Technique for Effort Estimation in Software Development"; Proc. XVIII Brazilian Symposium on Software Quality, (2019), 240-245.
- [Anandhi and Chezian 2014] Anandhi, V., Chezian, R. M.: "Regression techniques in software effort estimation using cocomo dataset"; Proc. International Conference on Intelligent Computing Applications, IEEE (2014), 353-357.
- [Arslan 2019] Arslan, F.: "A review of machine learning models for software cost estimation"; *Review of Computer Engineering Research*, 6, 2 (2019) 64-75.
- [Fedotova et al. 2013] Fedotova, O., Teixeira, L., Alvelos, H.: "Software Effort Estimation with Multiple Linear Regression: Review and Practical Application"; *Journal of Information Science and Engineering*, 29, 5 (2013), 925-945.
- [Goyal 2021] Goyal, S.: "Comparative Analysis of Machine Learning Techniques for Software Effort Estimation"; in *Proceeding of ICTSES*, Springer Nature Singapore, 2021.
- [Hall 1999] Hall, M. A.: "Correlation-based feature selection for machine learning"; (1999).
- [Hall et al. 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H.: "The WEKA Data Mining Software: An Update"; *SIGKDD Explorations*, 11, 1 (2009).
- [Hwang and Yoon 1981] Hwang, C. L., Yoon, K.: "Methods for Multiple Attribute Decision Making"; Proc. Multiple Attribute Decision Making, Springer, Berlin, Heidelberg, (1981), 58–191.
- [Kumar 2023] Kumar, A.: "Recommendation of Regression Models for Real Estate Price Prediction using Multi-Criteria Decision Making"; *Journal of Communications Software and Systems*, 19, 3 (2023) 220-229.
- [Kumar and K. Srinivas 2023] Kumar, K. H., Srinivas, K.: "An accurate analogy-based software effort estimation using hybrid optimization and machine learning techniques"; *Multimedia Tools and Applications*, (2023).

- [Mahmood et al. 2022] Mahmood, Y., Kama, N., Azmi, A., Khan, A. S., Ali, M.: “Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation”; *Software: Practice and Experience*, 52, 1 (2022), 39-65.
- [Menzies 2014] Menzies, T., Kocaguneli, E., Turhan, B., Minku, L., Peters, F.: “Sharing data and models in software engineering”; Morgan Kaufmann (2014).
- [Opricovic and Tzeng 2004] Opricovic, S., Tzeng, G. H.: Compromise solution by MCDM methods: “A comparative analysis of VIKOR and TOPSIS”; *European journal of operational research*, 156, 2 (2004), 445-455.
- [Pai et al. 2013] Pai, D. R., McFall, K. S., Subramanian, G. H.: “Software effort estimation using a neural network ensemble”; *Journal of Computer Information Systems*, 53, 4 (2013), 49-58.
- [Promise Repository] <http://promise.site.uottawa.ca/SERepository>
- [Rahman et al. 2023] Rahman, M., Roy, P. P., Ali, M., Gonc, T., Sarwar, H.: “Software Effort Estimation using Machine Learning Technique”; *International Journal of Advanced Computer Science and Applications*, 14, 4(2023), 822-827.
- [Rijwani and Jain 2016] Rijwani, P., Jain, S.: “Enhanced software effort estimation using multi layered feed forward artificial neural network technique”; *Procedia Computer Science*, 89 (2016), 307-312.
- [Sakhrawi et al. 2020] Sakhrawi, Z., Sellami, A., Bouassida, N.: “Software Enhancement Effort Estimation using Machine Learning Regression Methods”; *International Journal of Computer Information Systems and Industrial Management Applications*, 12 (2020), 412-423.
- [Sakhrawi et al. 2021] Sakhrawi, Z., Sellami, A., Bouassida, N.: “Software Enhancement Effort Prediction Using Machine-Learning Techniques: A Systematic Mapping Study”; *SN Computer Science*, 2, 6 (2021), 1-15.
- [Sanchez et al. 2023] Sanchez, E. R., Santacruz, E. F. V., Maceda, H. C.: “Effort and Cost Estimation Using Decision Tree Techniques and Story Points in Agile Software Development,” *Mathematics*, 11, 6, (2023).
- [Satapathy et al. 2016] Satapathy, S. M., Acharya, B. P., Rath, S. K.: “Early-stage software effort estimation using random forest technique based on use case points”; *IET Software*, 10, 1 (2016), 10-17.
- [Seo et al. 2013] Seo, Y. S., Bae, D. H., Jeffery, R.: “AREION: Software effort estimation based on multiple regressions with adaptive recursive data partitioning”; *Information and Software technology*, 55, 10 (2013), 1710-1725.
- [Sharma and Singh 2018] Sharma, P., Singh, J.: “Machine Learning Based Effort Estimation using Standardization”; *Proc. International Conference on Computing, Power and Communication Technologies*, IEEE (2018), 716-720.
- [Sharma and Vijayvargiya] Sharma, S., Vijayvargiya, S.: “Modeling of software project effort estimation: a comparative performance evaluation of optimized soft computing-based methods”; *International Journal of Information Technology*, 14, 5, (2022), 2487–2496.
- [Sheskin 2011] Sheskin, D. J.: “Handbook of parametric and nonparametric statistical procedures”; Chapman and Hall/CRC (2011).
- [Tayyab et al. 2016] Tayyab, M. R., Usman, M., Ahmad, W.: “A Machine Learning Based Model for Software Cost Estimation”; *Proc. SAI Intelligent Systems Conference*, Springer (2016), 402-414.

[Thakkar 2022] Thakkar, J. J.: Multi-Criteria Decision Making, 1st ed. Singapore, Singapore: Springer, 2022.

[Varshini et al. 2021] Varshini, A. P., Kumari, K. A., Janani, D., Soundariya, S.: “Comparative analysis of Machine learning and Deep learning algorithms for Software Effort Estimation”; *Journal of Physics: Conference Series*, 1767, 1 (2021), 012019.

[Wolpert and Macready 1995] Wolpert, D. H., Macready, W. G.: “No free lunch theorems for search”; *Technical Report SFI-TR-95-02-010*, Santa Fe Institute, 10 (1995).

[Zavadskas et al. 2012] Zavadskas, E. K., Turskis, Z., Antucheviciene, J., Zakarevicius, A.: “Optimization of weighted aggregated sum product assessment”; *Electronics and Electrical Engineering*, 122, 6 (2012), 3-6.