Applied Soft Computing 131 (2022) 109737

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A feature selection model for software defect prediction using binary Rao optimization algorithm

Karpagalingam Thirumoorthy [a,*,1], Jerold John Britto J. [b,1]

[a] Department of Computer Science and Engineering, Mepco Schlenk Engineering College, Sivakasi 626005, Tamilnadu, India
[b] Department of Mechanical Engineering, Ramco Institute of Technology, Rajapalayam 626117, Tamilnadu, India

ARTICLE INFO

Article history:
Received 10 March 2022
Received in revised form 5 September 2022
Accepted 14 October 2022
Available online 26 October 2022

Keywords:
Feature selection model
TOPSIS
Hybrid Rao optimization algorithm
Software defect prediction

ABSTRACT

In this digital world, using software has become an important part of daily life and business. The software must be rigorously tested in order to avert a financial crisis. The defect-free software enhances the functionality of the business. Predicting software defects in advance is a crucial task in the software industry. The aim of Software Defect Prediction (SDP) is to locate the possible software bugs. This paper proposes a hybrid feature selection (filter–wrapper) approach based on the multi-criteria decision making (MCDM) method and the Rao optimization method for selecting the more informative features to improve the software defect prediction rate. The proposed work measures the fitness of the candidate solution by using the defect prediction rate and the feature selection ratio. The performance of the proposed method is evaluated using three popular benchmark NASA datasets (PC5, JM1, and KC1) and compared with the state-of-the-art methods. The proposed feature subset selection scheme identifies the most significant feature subset for defect prediction with an average accuracy of 95% on the benchmark datasets. According to the experimental results, the proposed hybrid approach outperforms the standard strategy in terms of defect prediction rate.

1. Introduction

In this century, internet users are using various software in the form of web applications and mobile applications, in which the quality of software is considered to be one of the most significant components in software development [1,2]. In computer programming, a software defect is an error that has an incorrect impact and reduces the software's reliability. Software defects can be caused by logic errors, requirement errors, or design errors [3,4]. Finding defects in a software module can be extremely challenging for a software tester. Software Defect Prediction (SDP) is an essential part in the software development process. During the process of software testing, the automated software defect prediction system provides assistance to the software professional. The supervised machine learning algorithms are used to develop the software defect prediction system. It uses the supervised dataset to build the software defect prediction model. The feature space of the dataset is high. The high dimensional feature space reduces the performance of the defect prediction system. The high dimensional feature space may contain the redundant and non-informative features. In order to improve the performance of the fault prediction system, the feature selection is one of the dimension reduction(feature selection) techniques which improve the performance of the prediction system [5]. Basically, feature selection strategies are grouped into the following,

- Filter method: The filter-based approach [6] applies the scoring methodologies such as Information Gain, Gini index, etc. to all the feature. It computes the significance score of each feature. Finally, it uses the significance rank to pick the top-rated features.
- Wrapper method: The main goal of the wrapper-based strategy is to select the best subset of features [7]. The wrapper based approach depends on a specific supervised learning algorithm.

1.1. Software description record representation

The software description dataset S is a set of software modules. Let software dataset contains 'k' software modules, and is denoted as $S = \{s_1, s_2, s_3, \ldots, s_k\}$. The software description record is described by various software metrics. The set of software metrics is denoted as $M = [m_1, m_2, m_3, \ldots, m_l]$, where l is the total number of software metrics. Each software description record s_i is represented as a vector $\langle w_{i1}, w_{i2}, w_{i3}, \ldots, w_{im}\rangle$ where w_{ij} denotes the software metric value of m_j which is taken on the software s_i. Table 1 shows a typical software description record dataset in terms of feature(software metric) vector.

* Corresponding author.
E-mail address: kthirumoorthy@mepcoeng.ac.in (K. Thirumoorthy).
[1] All authors contributed equally and significantly in writing this article.

https://doi.org/10.1016/j.asoc.2022.109737
1568-4946/

Table 1
Software metric matrix.

Software description record	Software metrics			
	m_1	m_2	$\ldots m_i \ldots$	m_l
s_1	w_{11}	w_{12}	w_{1i}	w_{1l}
s_2	w_{21}	w_{22}	w_{2i}	w_{2l}
$\vdots$				
s_k	w_{k1}	w_{k2}	w_{ki}	w_{kl}

1.2. Pre-processing

Normalization enriches the data mining process. In order to standardize software metric values, the normalization technique is applied. In this study, we have used z-score normalization [8] techniques which is based on the software metric mean and standard deviation. The z-score transformation is formulated in Eq. (1)

$$w'_{ij} = \frac{w_{ij} - \mu_{m_j}}{\sigma_{m_j}} \tag{1}$$

1.3. Contribution and organization

The key features of the proposed TOPSIS based Hybrid Rao Optimization(THRO) method are

1. The multi-criteria decision making (MCDM) method called TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) is employed as filter method.
2. Self-Adaptive — The proposed method automatically calculates the population size of each iteration
3. Multi-population — The proposed method split the entire population into several sub-populations, and the hybrid Rao optimization principle has been applied to each sub-population.
4. Elitism — The proposed method placed the elitist candidate solution for the next generation.
5. Crossover and Mutation — The proposed method used the genetic operators to enhance the search ability

The remaining sections of this article are structured as follows: Various state-of-the-art software defect prediction methods are presented as related work in Section 2. The working principles of the proposed hybrid elitism based self-adaptive Rao optimization based software defect prediction technique is described in Section 3. Section 4 describes the dataset and evaluation metrics employed in this study. In Section 5, the experimental result is described. The validity threats of the proposed method is presented in Section 6. Finally, the conclusion of the article is given in Section 7.

2. Literature study

This section outlines some of the current research work related to SDP model construction using machine learning techniques, feature selection techniques, and optimization techniques.

2.1. Machine learning algorithms based SDP models

The software defect prediction model is constructed using various machine learning strategies, such as supervised learning, unsupervised learning, and association mining techniques [9]. In order to discover software defects, researchers have employed a wide variety of machine learning techniques, such as Naive Bayes (NB) [10], Bayesian Network [11], Decision Tree (DT) [12, 13], K-nearest neighbours (KNN) [14], support vector machine (SVM) [15,16], and K-means [17,18].

Abaei and Selamat [19] proposed fuzzy clustering based SDP model. They used eight NASA benchmark dataset. X-means clustering algorithm is proposed for SDP by the author Park and Hong in [20]. A complete survey of machine learning techniques based SDP model is presented by Pandey et al. in [21]. They reviewed 154 research articles. They highlighted that machine learning model is better than classical statistical models for software fault prediction. A comparative analysis of SDP model is presented in [22]. The author compared the various supervised learning algorithms and ensemble methods. Also, they highlighted the superiority of random forest classifier using 10 NASA benchmark dataset. Pandey et al. [23] applied ensemble techniques for software bug prediction. Assim et al. [24] presented the review study of supervised learning algorithms for predicting the software defects. The review article [25] highlighted the superiority of the machine learning algorithm for predicting the software defects.

2.2. Filter-based feature selection approach

In the literature, numerous filter-based feature selection schemes have been proposed. Chi-square [26], Information Gain [27], and Mutual Information [28] are prominent filter based approaches.

Kowshalya et al. [29] proposed new feature selection algorithm based on the Reverse Piece-wise Correlation and Shuffled Piece-wise Correlation. Khourdi and Bahaj [30] devised the Breast Cancer Prediction system. They used Fast Correlation-Based Filter method to select the informative feature for cancer prediction. Hancer et al. [31] proposed differential evolution feature selection method based on the ReliefF score, mutual information, and Fisher Score. In [32], the author proposed an iterative Relief-based algorithm for selecting informative features from the high dimensional feature space. Finally, they provided the comparative summaries of Relief-based algorithms. Bennasar et al. [33] proposed the Joint Mutual information Maximization(JMIM) to improve the performance of mutual information. They highlighted the performance of JMIM method using eleven public dataset. Liang et al. [34] proposed a method called CMIFSI (Conditional Mutual Information based Feature Selection considering Interaction). They used the conditional mutual information to identify the feature redundancy. In [6], the author presented the comparative study about the mutual information based feature selection methods. Alsaeedi and Khan [22] proposed the ensemble method for feature selection. They combined the information gain, relief, Gini index, and chi-square methods for ranking the features. They demonstrate the superiority of the ensemble method using fourteen UCI dataset. Kaur and Patil [35] proposed a filter method called weighted Minimum Redundancy Maximum Relevance (wMRMR). They ranked the features by using a three stage filter approach. They selected the informative features from the high-dimensional protein sequence data.

2.3. Meta-heuristics based software defects prediction

In recent years, the metaheuristic techniques are used in various application such as text classification [36], tumor classification [37], and intrusion detection system [38].

Malhotra et al. [39] used particle swarm optimization algorithm to develop the SDP model. They used Min–Max normalization techniques to normalize the dataset. Tung et al. [40] devised the software defect prediction model using teaching learning based optimization technique. They applied the optimal feature subset selection techniques to build the SDP model. A hybrid SDP model is developed in [41]. The author combined the machine

learning techniques and genetic algorithm. They used the decision tree based classification model. In [42], the author applied the Island Moth Flame Optimization technique for software defect prediction. They used Twenty-one public software datasets for evaluation. They used the flame optimization algorithm for feature selection. They trained the SVM classification model by using the selected optimal feature subset. In [43], the author applied particle swarm optimization algorithm for fault prediction. Panda and Azar [44] highlighted the hybrid grey wolf optimization algorithm for software bug prediction. Anbu and G S [45] used the firefly optimization techniques for SDP model construction. They used the SVM, K-NN, and NB classifier for prediction. Kiran Kumar and Narsimha [46] highlighted the use of Ant Colony Optimization for defect classification. They used eight open source benchmark dataset for evaluation.

2.4. Multi-criteria decision making approach

In recent years, the Multi-criteria decision making methods are used in various application such as material selection [47], Supplier Selection [48,49], Software Package Selection [50], task scheduling [51,52] and Cloud Service Selection [53].

In [54], the author used MCDM for ranking the classification algorithm based on 10 performance criteria. Singh et al. [55] applied the MCDM for feature selection. They used TOPSIS for selecting significant features in the network traffic dataset. Kou et al. [56] ranked the feature subset selection approach for document classification based on the 9 evaluation measure. They compared 5 MCDM method. Hashemi et al. [57] employed the TOPSIS for feature selection process. They used ridge regression algorithm to construct the decision matrix. They evaluated using by the medical dataset. In [58], the author highlighted the MCDM based feature selection approach. They used VIKOR method to rank the features. Bishnu and Bhattacharjee [17] proposed TOPSIS based feature selection scheme for anomaly detection.

Despite the fact that many research on feature selection have been proposed in the literature, the researcher focuses on improving the quality of the feature selection approach. In this study, we deployed the TOPSIS based hybrid Rao optimization technique for software defect prediction.

2.5. Basic principles of MCDM & TOPSIS

In this decade, Multi-Criteria Decision Making (MCDM) approaches are becoming more popular and are used in various real-world applications. MCDM techniques aid decision makers in dealing with complex problems. The decision maker needs to consider multiple factors/criteria/indicator during decision-making process. Each criterion has a varied impact (varying level of influence) on the decision-making process. The primary objective of the MCDM technique is to rank the alternatives in a preference order according to the criteria.

Technique for order preference similarity to ideal solution (TOPSIS) is a well-known multi-criteria decision making method proposed in [59]. The TOPSIS method ranks the alternatives by using multiple benefit criteria and cost criteria. TOPSIS assigns the score to the alternatives based on the positive ideal solution (PIS) and the negative ideal solution (NIS). PIS increases the benefit criteria/indicator and reduces the cost criteria/indicator. NIS increases the cost criteria/indicator and reduces the benefit criteria/indicator. The process of the TOPSIS method is described in Fig. 1.

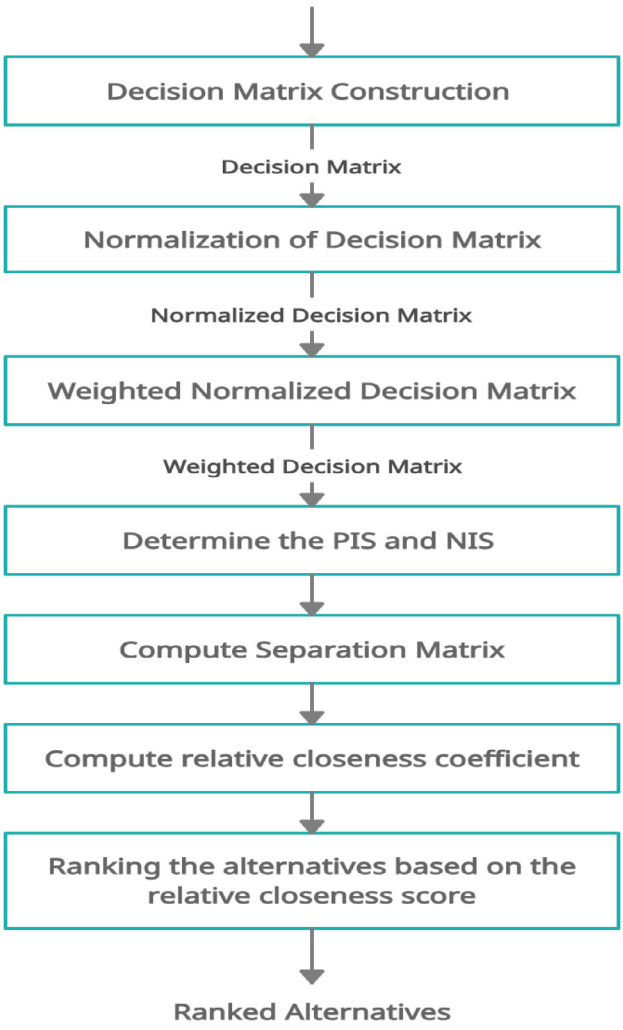


Fig. 1. Flowchart of TOPSIS.

2.6. Overview of Rao optimization algorithm

Rao optimization algorithm is one of the meta-heuristic optimization methods and was proposed by Professor Venkata Rao in [60]. Algorithm 1 describes the basic working principles of the Rao optimization method. The best candidate solution and worst candidate solutions are used to seek for a global optimal solution in the Rao optimization algorithm. In each iteration, the position of candidate solution is modified as follows,

$$\Delta'_{i,j} = \Delta_{i,j} + \alpha|\Delta_{best,j} - \Delta_{worst,j}| \quad (2)$$

where α is a random number.

Unlike other meta-heuristic methods such as Genetic Algorithm (GA), Grey Wolf Optimization (GWO), Firefly Optimization (FFY), and Cuckoo Search (CS), the Rao optimization algorithm does not require any specific control parameters.

3. Proposed work

In this decade, optimization techniques have been applied to many research problems. In this study, we applied optimization techniques to pick the optimal informative feature subset for classification. The framework of the proposed model is depicted in Fig. 2. Table 2 lists the symbols used in this work.

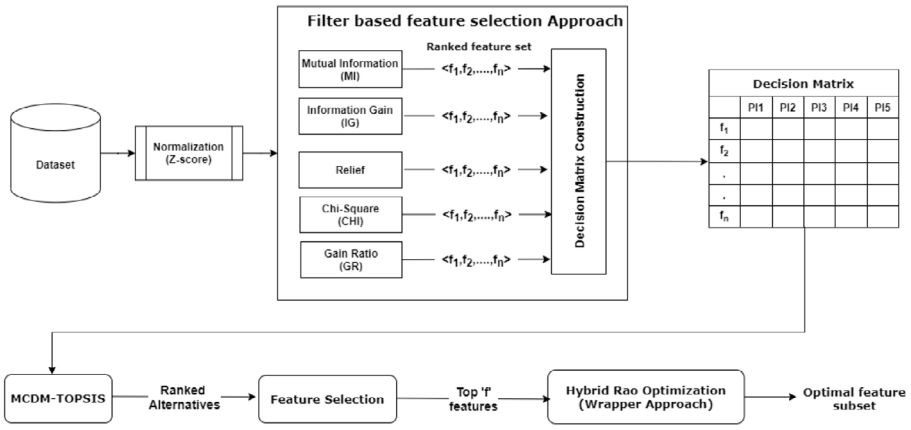


Fig. 2. Architecture of proposed work.

Algorithm 1: Pseudocode of Rao Optimization algorithm [60]

Input:
N : population Size
d : number of design variables
T_{max} : Termination Criteria (maximum iteration limit)
Output: best candidate - global optimal solution

Design the objective function
Generate initial population of 'N' candidate solution with 'd' dimension
Calculate the fitness of each candidate solution by using objective function
Identify the *best* and *worst* candidate solution from the population
while $t < T_{Max}$ **do**
  for *each candidate solution* Δ_i *i=1 to N* **do**
    Find new candidate solution $\Delta_i^{'}$ from the old solution Δ_i by using best and worst solution // Using by Eq. (2)
    compute the new candidate solution's fitness score
    if *new solution* $\Delta_i^{'}$ *is better than old solution* Δ_i **then**
      Ignore the old solution by keeping the new candidate solution
    else
      Keep the old solution and ignore the new solution
    end
  end
  update the best and worst candidate solution
end
return the best candidate solution

Table 2
List of notation.

Notation	Description
Γ	Decision Matrix
RF	Ranked Features
RA	Ranked Alternatives (ranked features)
i	index value of candidate solution
j	design variable index
t	iteration index
POP_{size}	population size
T_{max}	Termination Criteria (maximum no. of iteration)
Υ_i	ith candidate solution
$\Upsilon_{i,j}$	jth position of Solution body Υ_i
$\Upsilon_{i,fitness}$	*fitness* score of ith candidate solution Υ_i
Υ_{best}	*best* solution
$\Upsilon_{best,fitness}$	*fitness* score of best candidate solution
Υ_{worst}	*worst* solution
$\Upsilon_{worst,fitness}$	*fitness* score of worst candidate solution
α	random number [0-1]
β	population control rate

Table 3
Decision matrix for TOPSIS method.

Alternatives	Criteria				
	MI	IG	Relief	CHI	GR
f_1	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}
f_2	m_{21}	m_{22}	m_{23}	m_{24}	m_{25}
f_3	m_{31}	m_{32}	m_{33}	m_{34}	m_{35}
⋮					
f_n	m_{51}	m_{52}	m_{53}	m_{54}	m_{55}

3.1. TOPSIS based filter approach

Initially, the software module description dataset (S) is normalized by the Z-score normalization technique. Let the dataset contain 'n' features. In the preliminary screening, five well-known filter approaches are applied: Mutual Information (MI), Information Gain (IG), Relief, Chi-Square (CHI), and Gain Ratio (GR). Each filter method assigns a significant score to each feature. The final score of each feature is calculated based on five independent scores computed by five different filter methods. In order to compute the final score of each feature, we used the multi-criteria decision making method. In this work, we used TOPSIS to compute the final significant score based on the five filter method scores (criteria).

According to the five filter method scores, the decision matrix for the TOPSIS is generated. Table 3 shows the decision matrix, which has the following components,

- n alternatives i.e. ($f_1, f_2, f_3, \ldots$and f_n)
- 5 criteria i.e. filter methods (MI, IG, Relief, CHI, GR)
- criteria weights (w_1, w_2, w_3, w_4, w_5)

where m_{ij} denotes the significant score of ith alternative's (feature) assigned by jth filter method. In this study, we consider the equal weights for all criteria ($w_1 = w_2 = w_3 = w_4 = w_5 = 0.2$).

The TOPSIS approach is then employed on the decision matrix. The output of the TOPSIS is the ranked alternatives. From that, the top 'f' features (F_{TOPSIS}) are selected.

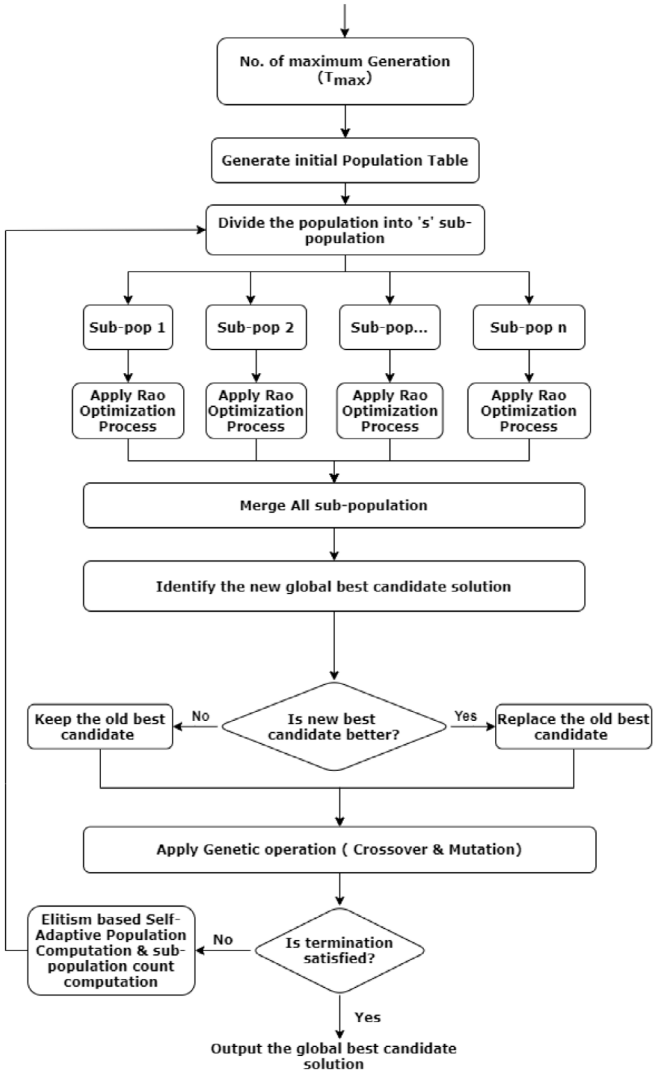


Fig. 3. Flowchart of hybrid Rao Optimization algorithm.

3.2. Hybridization of Rao optimization technique

The hybrid Rao optimization based wrapper approach would further examine the combined features. The wrapper approaches are not suited to a large feature set containing hundreds/thousands of features. Owing to the previous dimension/feature reduction method, the wrappers can now be applied with less computational effort. In this study, we integrate the Rao optimization search process with genetic functionality such as crossover and mutation schemes to improve the performance of the Rao optimization algorithm. The working principle of the proposed hybrid Rao optimization algorithm is shown in Fig. 3.

3.3. Encoding scheme and initial population

In this work, we used the binary coding scheme. A binary vector of size 'n'(size of $F_{combined}$: total no. of features extracted in the TOPSIS based filter approach) is used to represent the candidate solution. Each position of the candidate solution can be 1 or 0, which indicates the corresponding feature is selected or discarded. Let the sample candidate solution is [0, 0, 1, 1, 0, 0, 1, 1, 1, 0]. It specifies that the features appeared in the index 3, 4, 7, 8 *and* 9 are selected. similarly, the features appeared in the index 1, 2, 5, 6, *and* 10 are unselected.

In the search space, the set of candidate solutions is called Population. The size of F_{TOPSIS} defines the initial population size (POP_{size}). It is calculated as follows,

$$N = POP_{size} = |F_{TOPSIS}| * \Omega \tag{3}$$

where Ω is the initial population size factor. In our experiment, we have tested different values for initial population size factor (Ω) (5%, 10%, 15%, 20%). It is observed that initial population size factor 10% gives the optimum result while considering the classification accuracy and computation time.

For initial population table, the set of POP_{init} (population size) candidate solutions are randomly generated. In order to convert the search space from continuous to discrete, the sigmoid transfer function is used, which is defined as follows,

$$T(\Theta) = \frac{1}{1 + e^{-\Theta}} \tag{4}$$

After converting the real value to a probability value, each design variable of the candidate solution in the population is updated to a binary value as defined in Eq. (5).

$$\Upsilon_{i,j} = \begin{cases} 1, & if \;\; T(\Upsilon_{i,j}) > random(0, 1) \\ 0, & otherwise \end{cases} \tag{5}$$

3.4. Elitism based self-adaptive multi-population scheme

The proposed THRO feature selection algorithm automatically calculates the population size for each iteration. The population size of next iteration is calculated as,

$$POP_{new} = round(POP_{old}(1 + \beta)) \tag{6}$$

where β is the population control rate. The population control rate is a real value that is chosen at random. It will be in the range of [−0.5 to 0.5]. If the population control rate is a positive value, then the population size will be increased. For example, if the old population (POP_{old}) size is 100 and the population control rate (β) is 0.2, then the size of the new population (POP_{new}) is 120. Similarly, if the population control rate is a negative value then the population size will be decreased. For example, if the old population (POP_{old}) size is 100 and the population control rate (β) is −0.2, then the size of new population (POP_{new}) is 80. If the population size of the new iteration is greater than the old iteration, then elitism is followed to add the additional (POP_{new} - POP_{old}) number of candidate solutions. If the new generation's population has fewer individuals than the previous generation ($POP_{new} < POP_{old}$), then the superior candidate solutions of the old generation will be carried over to the new iteration. The total population is initially separated into three subgroups. The hybrid Rao optimization technique is applied to each sub-population. Independently, each subgroup's candidate solutions are modified. Finally, the sub-population groups are merged to find the global best candidate. For the next iteration, the sub-population count is calculated as follows,

$$s' = \begin{cases} s + 1 & \text{if } \Upsilon^{(t+1)}_{best,fitness} \leq \Upsilon^{t}_{best,fitness} \\ s - 1 & \text{if } \Upsilon^{(t+1)}_{best,fitness} > \Upsilon^{t}_{best,fitness} \end{cases} \tag{7}$$

3.5. Objective function for evaluating candidate solution

In optimization problems, the objective function (fitness function) plays an important role in discovering an ideal solution. The fitness score is a scalar value computed by the objective function. In this study, we define the fitness function based on the error rate of the classifier and the feature selection ratio.

$$fitness(\Upsilon_i) = ErrorRate(\Upsilon_i) + \frac{f_{selected}}{|F_{TOPSIS}|} \tag{8}$$

Table 4
Single-division-point crossover operation (division location: 4).

Before crossover										
Parent-1	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**1**
Parent-2	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
After crossover										
Child-1	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
Child-2	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**1**

Table 5
Flip mutation operation (Point1:4, Point2:8).

Before Mutation	0	1	0	1	0	0	1	0	1	1
After Mutation	0	1	0	0	1	1	0	1	1	1

where $f_{selected}$ - number of features selected in the given solution Υ_i. Testing error rate of the classifier is defined as

$$ErrorRate = \frac{n_{misclassified}}{|D_{Test}|} \tag{9}$$

where $n_{misclassified}$ - number of misclassified samples in the testing set.

3.6. Updating candidate solution for next generation

Each candidate solutions of the iteration 't' population $\psi^{(t)}$ is transformed to the population of the next generation $\psi^{(t+1)}$ using by the best ($\Upsilon^{(t)}_{best}$) and worst $\Upsilon^{(t)}_{worst}$ solutions of the generation 't'.

In generation 't', the jth position of the ith solution $\Upsilon^{(t)}_{i,j}$ is modified as $\Upsilon^{(t+1)}_{i,j}$ at generation 't+1' and is formulated in Eq. (10)

$$\Upsilon^{(t+1)}_{i,j} = \Upsilon^{(t)}_{i,j} + \alpha \left| \Upsilon^{(t)}_{best,j} - \Upsilon^{(t)}_{worst,j} \right| \tag{10}$$

After that, the Sigmoid transfer function is applied to convert the search space from continuous to discrete. Finally, each position of the candidate solution is converted to a binary value as defined in Eq. (5).

3.7. Hybridization using genetic operator

Crossover: Crossover is one of the genetic operators. In evolutionary algorithms, the crossover operator creates new child candidate solutions by combining two parent candidate solutions. The single-division-point crossover scheme is used in this work. *Single-division-point crossover:* The parent solutions are separated into two halves based on the random division point (random integer in the range of 1 to n). Then, the last halves of the parent candidate solutions are swapped. For example, the functionality of the crossover genetic operator is shown in Table 4.

The crossover operation is performed by randomly selecting two of the best N/2 individuals in the population. ***Mutation:*** Mutation is one of the genetic operators. The mutation operator performs the minor random changes in the candidate solution to create a new solution. In this work, the flip mutation scheme is used. Initially, two location points (point1, point2) in the candidate solution are randomly selected. Then, invert the values (0 to 1 and vice-versa) from point 1 to point 2. The process of flip mutation is shown in the Table 5. The iterative process of the proposed work runs T_{max} times. After that, the iterative process will be stopped and the best (Υ_{best}) candidate solution will be given as the global optimal solution.

3.8. Algorithmic description of the proposed hybrid feature selection approach

Algorithm 2 describes the proposed hybrid feature selection approach. The inputs are software description dataset with class label (S), classifier (C), List of filter methods (MI, IG, Relief, CHI, and GR), weight of the each filter (Wt) and the termination criteria of the iterative process (T_{max}). The output of the proposed algorithm is the best optimal feature subset (F_{best}). Line 1 load the software description dataset (S). Line 2- normalize the software metric matrix(SMM). Line 4–7 construct the decision matrix based on the 5 filter method scores. Line 8 apply the MCDM (TOPSIS) method on the decision matrix. It returns the ranked alternatives. Line 9 selects the top ranked features based on the TOPSIS score. Line 10 modify the SMM based on the F_{TOPSIS} feature subset. Line 11 apply the hybrid feature selection approach. Finally, Line 12 returns the best optimal candidate solution.

Algorithm 2: Pseudocode of TOPSIS based Hybrid Rao Optimization

Input:
S : Software module description dataset with class label
C : Classification Algorithm
Filters:[MI, IG, Relief, CHI, GR] **//list of filter methods**
Wt:[w_1,w_2,w_3,w_4,w_5] **//weight values of each filter methods**
T_{max} : Termination Criteria (maximum iteration limit)
Output: F_{best} - optimal feature subset

1 SMM←loadSoftwareMetricDataset(S)
2 SMM ← dataPreProcessing(SMM) `// Apply Z-Score normalization`
3 $\Gamma \leftarrow \Phi$ `// Initialize the Decision Matrix`
4 **foreach** *filter method fi in Filters* **do**
5 RF_{fi}←computeSignificantScore(SMM,fi) `// assign the score to each features based on filter method fi`
6 Γ.append(RF_{fi}) `// Append the RF of filter method fi into the Decision Matrix`
7 **end**
8 RA←TOPSIS(Γ,Wt) `// Ranking the filter methods based on TOPSIS`
9 F_{TOPSIS}←selectTopRankFeatures(RA)
10 SMM←SMM[:,F_{TOPSIS}] `// Dimension reduction by filters based on TOPSIS`
11 F_{best}←applyHybridRaoOptimization(F_{TOPSIS},SMM,C,T_{max}) `// Hybrid Rao Optimization based Wrapper approach`
12 **return** F_{best}

Algorithm 6 describes an elitist self-adaptive multi-population hybrid Rao optimization based feature selection approach. The inputs are informative feature subset (F_{TOPSIS}), software description dataset (SMM), classification algorithm (C), and the termination criteria of the iterative process (T_{max}). The output is the best candidate solution (Υ_{best}). Line 1 calculates the size of the initial population table. Lines 2–8 create the initial population table. The fitness score is computed by using the algorithm 4. Line 7 indicates the fitness computation process. Lines 11–43 describe the core process of the hybrid elitist self-adaptive Rao optimization technique. Lines 24–35 describe the genetic operation functionality. The proposed algorithm automatically computes the population size for the next iteration. Line 39 computes the new population size. Lines 40–44 use the elitism method to copy

the new population table from the old population table. Line 54 returns the best optimal candidate solution.

Algorithm 3 describes the Rao optimization algorithm process. The inputs are sub-population (POP_s), filtered feature subset (F_{TOPSIS}) document term matrix (DTM), and classification algorithm(C). The output of the Algorithm 3 is the updated population table (POP'_s). Line 1 initialize the POP'_s population table. Line 2 finds the leader candidate solution. Lines 3–14 describe the follower candidate solution movements towards to the leader candidate solution. Line 13 append the updated candidate solution to the new population table. Line 15 returns the new sub-population table.

Algorithm 3: Rao Optimization Algorithm

Input:
POP_s : sub-population
F_{TOPSIS} : TOPSIS based filtered feature subset
SMM : software module description dataset
C : Classification Algorithm
Output: POP'_s: updated sub-population

1 $POP'_s \leftarrow \emptyset$ // Initialize the POP'_s
2 $\Upsilon^s_{best} = findBestCandidateSolution(POP_s)$;
3 $\Upsilon^s_{worst} = findWorstCandidateSolution(POP_s)$;
4 **foreach** *candidate solution* Υ_i *i=1 to* $|POP_s|$ **do**
5 **foreach** *position/design variable j of candidate solution* Υ_i*; j=1 to* n_s **do**
6 $\alpha \leftarrow random(0, 1)$ // random real value [0.0 to 1.0]
7 $\Phi_{i,j} \leftarrow \Upsilon_{i,j} + \alpha \left|\Upsilon_{best,j} - \Upsilon_{worst,j}\right|$
8 $\Phi_{i,j} \leftarrow binarization\big((Sigmoid(\Phi_{i,j})\big)$
9 **end**
10 $\Phi_{i,fitness} = computeFitness\,(\Phi_i, F_{TOPSIS}, SMM, C)$
11 **if** $\Phi_{i,fitness} < \Upsilon_{i,fitness}$ **then**
12 $\Upsilon_i = \Phi_i$ // replace the old solution by new (temporary) solution
13 **end**
14 $POP'_s.append(\Upsilon_i)$ // Add the candidate solution to the new population subset
15 **end**
16 *return* POP'_s

The fitness calculation process is presented in Algorithm 4. The algorithm inputs are listed in Line 1. Line 2 retrieves the features that were selected from the input candidate solution. Line 3 slices the dataset based on the selected features. From the original dataset, Line 4 separates the dataset into a training set and a testing set. Line 5 describes the classification model construction process. Line 6 calculates the error rate. Line 7 computes the fitness score of the candidate solution. Finally, Line 8 returns the fitness score.

Algorithm 5 describes the elitism process. The algorithm inputs are listed in Line 1. Line 2 initialize the new population table. Line 3 sort the entire candidate solution in the population by fitness score. Line 4 choose the top POP_{count} number of candidate solution from the sorted population table. Line 5 return the new population table.

4. Implementation details

A brief overview of the datasets, classifiers, and evaluation methods that were used in this study are presented in this section.

Algorithm 4: Fitness score calculation

Input:
Υ_{input} - input candidate solution
F_{TOPSIS} : TOPSIS based filtered feature subset
SMM - Software module description dataset
C - Classification Algorithm
Output: *fitness_score*

1 **Function** $computeFitness\,(\Upsilon_{input}, F_{TOPSIS}, SMM, C)$
2 $index \leftarrow getSelectedFeatures(\Upsilon_{input})$
3 SMM←SMM[:,index]
4 X_{train}, X_{test}, Y_{train}, Y_{test} ←train_test_split(SMM,test_size=0.3) // sklearn.model_selection Python API
5 classifier←constructClassificationModel(X_{train}, Y_{train},C) // Classification model training
6 errorRate← computeErrorRate(classifier,X_{test},Y_{test})
7 $fitness\_score = errorRate + \frac{sizeOf(index)}{|F_{TOPSIS}|}$
8 **return** *fitness_score*
9 **End Function**

Algorithm 5: Copy Best Optimal Solution from the Population Table

Input:
POP_{Main} - Old Population Table
POP_{count} - needed population count
Output: POP_{new}- New Population Table

1 **Function** $CopyBestOptimalSolution\,(POP_{Main}, POP_{count})$
2 $POP_{new} \leftarrow \emptyset$ // Initialize the POP_{New}
3 $POP_{Main} \leftarrow$ SortPopulationByFitness(POP_{Main}) // sort the population table by fitness score
4 $POP_{new} \leftarrow POP_{Main}\big[:POP_{count}\big]$ // copy the best candidate solution from the POP_{Main}
5 **return** POP_{new}
6 **End Function**

4.1. Dataset

The performance of the proposed software defect prediction scheme is evaluated using three public benchmark datasets [PC5, JM1, KC1][2] which are available in the NASA-MDP public dataset repositories. Table 6 shows summaries of the public benchmark dataset mentioned above. Table 7 shows the descriptions of the software metrics.

4.2. Supervised learning algorithms used

We employed the two most prominent supervised classification algorithms in this proposed study. A 10-fold cross-validation strategy is applied to construct the software defect prediction model.

1. Naive Bayes (NB) classifier: NB classifier [61,62] works based on the joint probabilities of the attributes. It uses the Bayes theorem to predict the class label.
2. Support Vector Machine (SVM) classifier: SVM [15,16] constructs the hyperplane that segregates the data points into the two classes. SVM separates the two classes by the maximum margin hyperplane. The support vectors define the border of the hyperplane.

[2] https://github.com/klainfo/NASADefectDataset/tree/master/OriginalData/MDP.

Algorithm 6: Feature selection using hybrid Rao optimization algorithm

Input:
F_{TOPSIS} : TOPSIS based filtered feature subset
SMM : software description dataset
C : Classification Algorithm
T_{max} :Termination Criteria (maximum no. of iteration)
Output: Υ_{best}: best candidate solution

$POP_{size} \leftarrow 0.10 * |F_{TOPSIS}|$ // compute the size of initial population table See Eqn (4.1)
// Initial Population Table Construction
foreach *candidate solution* Υ_i *i=1 to* POP_{size} **do**
  foreach *design variable/position j of candidate solution* Υ_i*; j=1 to* n_s **do**
    $\Upsilon_{i,j} = random(0,1)$ // random real value [0.0 to 1.0]
    $\Upsilon_{i,j} = binarization((Sigmoid(\Upsilon_{i,j}))$ // convert the real value to binary value
  end
  $\Upsilon_{i,fitness} = computeFitness\left(\Upsilon_i, F_{combined}, D, C\right)$ // Fitness score computation
end
$\Upsilon_{best} \leftarrow$ findBestCandidateSolution(POP_{Main}) // Identify the best candidate solution
s_{count}=3 // sub-population count: initially, POP_{Main} is divided into '3' sub-population
foreach *iteration t 1 to* T_{max} **do**
  $POP_s \leftarrow splitMainPopulationToSubPopulation(POP_{Main}, s_{count})$ // Split POP_{Main}
  foreach *sub population si in* POP_{Main} *1 to* s_{count} **do**
    $POP_{si} \leftarrow applyRaoOptimization(POP_{si}, F_{combined}, D, C)$
  end
  $POP_{Main} \leftarrow mergeMultiPopulationToMainPopulation(POP_s)$ // Merge all sub-population
  $\Upsilon'_{best} \leftarrow$ findBestSolution(POP_{Main})
  if $\Upsilon'_{best,fitness} < \Upsilon_{best,fitness}$ **then**
    $\Upsilon_{best} \leftarrow \Upsilon'_{best}$ // update the best (Υ_{best}) candidate solution
    $s_{count} \leftarrow s_{count}$-1 // update population count
  else
    $s_{count} \leftarrow s_{count}$+1 // Ignore the Υ'_{best} and update the population count
  end
  $(\Upsilon_{parent1}, \Upsilon_{parent2}) \leftarrow selectTwoParentSolution(\Upsilon, POP_{size}/2)$ // Select two parents from Population
  $\Upsilon_{child1}, \Upsilon_{child2} \leftarrow Crossover(\Upsilon_{parent1}, \Upsilon_{parent2})$
  if $\Upsilon_{child1,fitness} < \Upsilon_{best,fitness}$ **then**
    $\Upsilon_{best} = \Upsilon_{child1}$ // replace the best solution by child1 solution
  end
  if $\Upsilon_{child2,fitness} < \Upsilon_{best,fitness}$ **then**
    $\Upsilon_{best} = \Upsilon_{child2}$ // replace the best solution by child2 solution
  end
  $\Upsilon_{mutated} \leftarrow MutateSolution(\Upsilon_{best})$
  if $\Upsilon_{mutated,fitness} < \Upsilon_{best,fitness}$ **then**
    $\Upsilon_{best} = \Upsilon_{mutated}$ // replace the best by mutated solution
  end
  $POP'_{size} \leftarrow POP_{size}$*(1-$\beta$) // Calculate the new population size
  if $POP'_{size} < POP_{size}$ **then**
    $POP_{Main} \leftarrow$ CopyBestOptimalSolution$(POP_{Main}, POP'_{size})$
  else
    $POP_{Main} \leftarrow POP_{Main} \cup$ CopyBestOptimalSolution$(POP_{Main}, POP'_{size} - POP_{size})$ // Use Elitism
  end
  $POP_{size} \leftarrow POP'_{size}$; // Update the POP_{size}
end
return Υ_{best}

Table 6
NASA metrics data program public dataset.

Dataset	No. of features	Total number of instance	Number of defective instance	Number of non-defective instance
PC5	39	17 186	516	16 670
JM1	21	10 885	8779	2106
KC1	21	2109	326	1783

4.3. Performance evaluation criteria

The proposed feature selection method is evaluated using standard evaluation criteria: accuracy(success rate), macro averaged precision(P), macro averaged recall(R), and macro averaged F-Score(F_1). They are determined by the four well-known factors listed below.

1. True Positive(TP): classification model accurately predicts the positive class.
2. True Negative(TN): classification model accurately predicts the negative class.
3. False Positive(FP): The classifier misclassified the negative class as positive.
4. False Negative(FN): The classifier misclassified the positive class as negative.

The success rate(accuracy) of the classifier is defined as,

$$Accuracy(success\ rate) = \frac{TP + TN}{P + N} \tag{11}$$

The Precision(P) refers to the percentage of samples labelled as positive that are truly positive.

$$Precision(P) = \frac{TP}{TP + FP} \tag{12}$$

Table 7
Description of software metrics.

S.No.	Attribute	Type	Description
1	loc	Numeric	Line count of code
2	v(g)	Numeric	Cyclomatic complexity
3	ev(g)	Numeric	Essential complexity
4	iv(g)	Numeric	Design complexity
5	n	Numeric	Total number of operators and operands
6	v	Numeric	Volume
7	l	Numeric	Program length
8	d	Numeric	Measure difficulty
9	i	Numeric	Measure intelligence
10	e	Numeric	Measure effort
11	b	Numeric	Effort estimate
12	t	Numeric	Time estimator
13	lOCode	Numeric	Line count
14	lOComment	Numeric	Count of lines of comments
15	lOBlank	Numeric	Count of blank lines
16	lOCodeAndComment	Numeric	Number of codes and comments
17	uniq_Op	Numeric	Unique operators
18	uniq_Opnd	Numeric	Unique operands
19	total_Op	Numeric	Total operators
20	total_Opnd	Numeric	Total operands
21	branchCount	Numeric	Flow graph branch count
22	defects	Boolean	Class Labeltrue, false

The Recall(R) refers to the percentage of true positives that were identified during the classification.

$$Recall(R) = \frac{TP}{TP + FN} \tag{13}$$

The F_1-Score(F) is a performance measure which combines the precision and recall of the classification model.

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{14}$$

The sensitivity is defined as the true positive rate.

$$TPR = \frac{TP}{P} \tag{15}$$

The specificity is defined as the true negative rate.

$$TNR = \frac{TN}{N} \tag{16}$$

False Positive Rate is calculated by the fraction of false positive with all actual non-defective software modules.

$$FPR = \frac{FP}{FP + TN} \tag{17}$$

False Negative Rate is calculated by the fraction of false negative with all actual defective software modules.

$$FNR = \frac{FN}{FN + TP} \tag{18}$$

5. Results and discussion

This section presents a comprehensive experimental results analysis to demonstrate the performance of the proposed TOPSIS-HRO based feature subset selection scheme.

5.1. Algorithms for comparison

The proposed TOPSIS-HRO feature selection technique was compared with the following evolutionary based feature selection techniques,

- Genetic Algorithm(GA) [63]
- Particle Swarm Optimization Algorithm (PSO) [64]
- Firefly Algorithm(FA) [65]
- Boosted Whale Optimization Algorithm(BWO) [66]
- Hybrid Grey Wolf optimization Algorithm(HGWO) [67]
- Binary Jaya optimization(BinJaya) [68]
- Cat and Mouse optimization(CAMO) [69]
- Group Mean-Based Optimizer(GMBO) [70]

Table 8
Algorithm specific parameter values.

Parameter	Value
GA	
C-Crossover method	one-point
P_C-Crossover probability	0.5
M-Mutation method	Swap
P_M-Mutation probability	0.3
PSO	
W - inertia weight	0.65
Wdamp - inertia weight damping ratio	0.6
C1 - personal learning coefficient	1.25
C2 - global learning coefficient	1.25
FA	
α - randomization parameter	0.1
B_0 - Base attraction	1.0
γ - Absorption coefficient	1.0
BWO	
a - linearly decreasing parameter	$a \in [2, 0]$
b -constant(logarithmic spiral shape)	1
l - randomization parameter	$[-1,1]$
HGWO	
a - controlling parameter	$a \in [2, 0]$

Table 8 lists the parameters of the various algorithms employed in this study.

5.2. Comparison results analysis on the PC5 dataset

In Table 9, the performance of various feature selection techniques have compared using NB classifier and SVM classifier on PC5 dataset. By using NB classifier, the proposed THRO based feature selection method has achieved 94.99% of classification accuracy. According to the data presented in Table 9, the proposed feature selection method has precision(P):0.9794, recall(R):0.9499, and F_1-score(F_1):0.9603. It shows the superiority of the proposed feature selection scheme. By using SVM classifier, the proposed THRO method has obtained the classification accuracy of 95.94% and the error rate of 4.06%. For the SVM classifier, the macro averaged precision, recall, and F_1 score of the proposed feature selection model are 0.9813, 0.9594, and 0.9669, respectively,

Table 9
Accuracy comparison on PC5 Dataset.

Feature selection methods	NB Classifier accuracy in %					SVM Classifier accuracy in %				
	ACC	Err	P	R	F_1	ACC	Err	P	R	F_1
GA	85.79%	14.21%	0.9697	0.8579	0.9017	86.79%	13.21%	0.9704	0.8679	0.9079
PSO	87.71%	12.29%	0.9712	0.8771	0.9136	88.71%	11.29%	0.972	0.8871	0.9199
FA	88.14%	11.86%	0.9715	0.8814	0.9163	89.14%	10.86%	0.9723	0.8914	0.9225
BWO	90.44%	9.56%	0.9736	0.9044	0.9307	91.44%	8.56%	0.9746	0.9144	0.9371
HGWO	90.94%	9.06%	0.974	0.9094	0.9339	91.94%	8.06%	0.9751	0.9194	0.9402
BinJaya	92.39%	7.61%	0.9757	0.9239	0.9431	93.39%	6.61%	0.9769	0.9339	0.9496
CAMO	92.49%	7.51%	0.9757	0.9249	0.9437	93.49%	6.51%	0.977	0.9349	0.9502
GMBO	93.12%	6.88%	0.9766	0.9312	0.9478	94.09%	5.91%	0.978	0.9409	0.9542
THRO	94.99%	5.01%	0.9794	0.9499	0.9603	95.94%	4.06%	0.9813	0.9594	0.9669

Table 10
Classifier performance comparison on PC5 Dataset.

Feature selection methods	NB Classifier performance				SVM Classifier performance			
	TPR	TNR	FPR	FNR	TPR	TNR	FPR	FNR
GA	0.8579	0.8566	0.1434	0.1421	0.8679	0.8663	0.1337	0.1321
PSO	0.8771	0.876	0.124	0.1229	0.8871	0.8857	0.1143	0.1129
FA	0.8814	0.8798	0.1202	0.1186	0.8914	0.8895	0.1105	0.1086
BWO	0.9044	0.9031	0.0969	0.0956	0.9145	0.9128	0.0872	0.0855
HGWO	0.9095	0.907	0.093	0.0905	0.9195	0.9167	0.0833	0.0805
BinJaya	0.9239	0.9225	0.0775	0.0761	0.934	0.9322	0.0678	0.066
CAMO	0.925	0.9225	0.0775	0.075	0.935	0.9322	0.0678	0.065
GMBO	0.9312	0.9302	0.0698	0.0688	0.9409	0.9399	0.0601	0.0591
THRO	0.9494	0.9477	0.0523	0.0506	0.95	0.9477	0.0523	0.05

Table 11
Results of the repeatability test on PC5 Dataset.

Feature selection methods	NB Classifier accuracy				SVM Classifier accuracy			
	Best	Worst	Avg.	Stdev.	Best	Worst	Avg.	Stdev.
GA	0.8579	0.7994	0.8378	0.0208	0.8679	0.7995	0.8366	0.0248
PSO	0.8771	0.8195	0.8472	0.0224	0.8871	0.8198	0.8552	0.026
FA	0.8814	0.8294	0.8598	0.0197	0.8914	0.8296	0.8609	0.0231
BWO	0.9044	0.8395	0.8781	0.0243	0.9144	0.8396	0.8802	0.028
HGWO	0.9094	0.8495	0.8804	0.0232	0.9194	0.8596	0.8914	0.0233
BinJaya	0.9239	0.8896	0.9077	0.0133	0.9339	0.8796	0.9127	0.0199
CAMO	0.9249	0.8994	0.9125	0.0088	0.9349	0.8894	0.9155	0.0176
GMBO	0.9312	0.8994	0.9147	0.0119	0.9409	0.8994	0.9222	0.0154
THRO	0.9499	0.9244	0.9381	0.0085	0.9594	0.9246	0.9423	0.0129

which are far superior to the other algorithms. In Table 10, the performance of various feature selection scheme based software defect prediction techniques have compared using TPR, TNR, FPR, and FNR. According to the data presented in the Table 10, it exhibits the superiority of the proposed THRO based feature selection scheme. The boxplots in Fig. 4 show the distribution of the classification accuracy for NB classifier and SVM classifier. These distributions show that the highest classification accuracy (success rate) was obtained by the proposed THRO feature selection method. The data in Fig. 4 clearly shows that the proposed method has less variation in the best, median, and worst results for both NB classifier and SVM classifier. Table 11 shows the multi-trail(20 runs) performance of the proposed THRO feature subset selection method on PC5 dataset. From Table 11, it is noted that the average classification accuracy attained by the proposed feature selection method is 93.81% with 0.0085 of standard deviation (NB classifier) and 94.23% with 0.129 of standard deviation(SVM classifier). It is better compared to the alternative techniques. The feature selection ratio of the proposed feature selection method THRO and other comparative methods over PC5 dataset using NB classifier and SVM classifier are presented in Fig. 5. It can be noted that, the proposed feature selection method obtained the less feature selection ratio. It shows the superiority of the proposed feature selection approach.

5.3. Comparison results analysis on the JM1 dataset

The comparison results presented in the Table 12 are defect prediction rate(Accuracy), error rate, precision, recall, and F_1-score of NB classifier and SVM classifier using various feature selection scheme on JM1 dataset. The macro averaged F-score of proposed feature selection scheme THRO is 0.9466 (NB classifier) and 0.9545 (SVM classifier). By using NB classifier, the proposed THRO based software defect prediction method has achieved 93.93% of classification accuracy. According to the data presented in Table 12, By using SVM classifier, the proposed THRO based defect prediction method obtained the classification accuracy of 94.93% and the error rate of 5.07%. Table 12 demonstrates the superiority of the proposed feature selection approach on the JM1 public dataset.

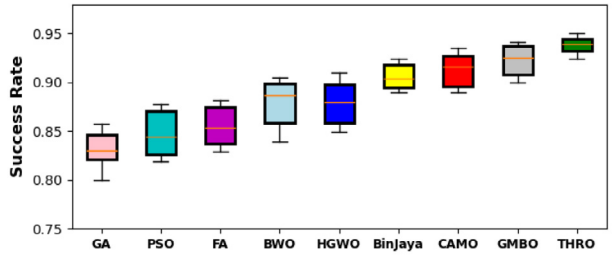
Table 13 shows the performance (TPR, TNR, FPR, and FNR) of the proposed THRO feature subset selection method on JM1 dataset. From Table 13, it is noted that the TPR attained by the proposed feature selection method is 0.9394 (NB classifier) and 0.9493 (SVM classifier). It is better compared to the alternative techniques. Fig. 6 depicts the classification accuracy distribution of the proposed THRO based software defect prediction scheme and other comparative methods over 20 separate independent runs on the JM1 dataset. It is evident from the boxplots that the gaps between outer and inner quartiles whiskers are tiny, which indicates that the proposed THRO feature selection technique's

Table 12
Accuracy comparison on JM1 Dataset.

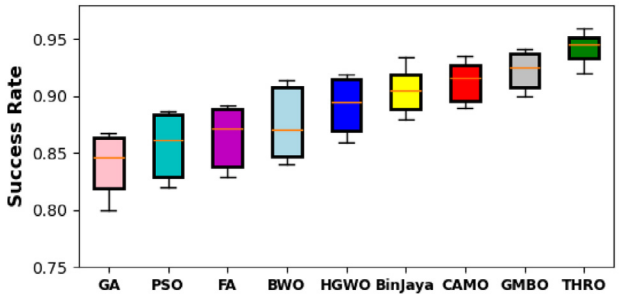
Feature selection methods	NB Classifier accuracy in %					SVM Classifier accuracy in %				
	ACC	Err	P	R	F_1	ACC	Err	P	R	F_1
GA	85.78%	14.22%	0.89	0.8578	0.8668	87.78%	12.22%	0.9028	0.8778	0.8847
PSO	86.7%	13.3%	0.8958	0.867	0.875	88.7%	11.3%	0.9089	0.887	0.893
FA	87.13%	12.87%	0.8986	0.8713	0.8789	89.13%	10.87%	0.9119	0.8913	0.897
BWO	89.44%	10.56%	0.9139	0.8944	0.8997	90.43%	9.57%	0.9208	0.9043	0.9088
HGWO	90.93%	9.07%	0.9243	0.9093	0.9134	91.93%	8.07%	0.9316	0.9193	0.9226
BinJaya	91.38%	8.62%	0.9275	0.9138	0.9175	92.38%	7.62%	0.9348	0.9238	0.9268
CAMO	91.58%	8.42%	0.929	0.9158	0.9194	92.49%	7.51%	0.9356	0.9249	0.9277
GMBO	92.49%	7.51%	0.9356	0.9249	0.9277	92.99%	7.01%	0.9394	0.9299	0.9324
THRO	93.93%	6.07%	0.9466	0.9393	0.9412	94.93%	5.07%	0.9545	0.9493	0.9507

Table 13
Classifier performance comparison on JM1 dataset.

Feature selection methods	NB Classifier performance				SVM Classifier performance			
	TPR	TNR	FPR	FNR	TPR	TNR	FPR	FNR
GA	0.8578	0.8575	0.1425	0.1422	0.8779	0.8775	0.1225	0.1221
PSO	0.8671	0.8666	0.1334	0.1329	0.8871	0.8865	0.1135	0.1129
FA	0.8714	0.8708	0.1292	0.1286	0.8913	0.8913	0.1087	0.1087
BWO	0.8944	0.8941	0.1059	0.1056	0.9043	0.9041	0.0959	0.0957
HGWO	0.9094	0.9088	0.0912	0.0906	0.9194	0.9193	0.0807	0.0806
BinJaya	0.9139	0.9136	0.0864	0.0861	0.9239	0.9236	0.0764	0.0761
CAMO	0.9159	0.9155	0.0845	0.0841	0.9249	0.9245	0.0755	0.0751
GMBO	0.9249	0.9245	0.0755	0.0751	0.9299	0.9297	0.0703	0.0701
THRO	0.9394	0.9387	0.0613	0.0606	0.9493	0.9492	0.0508	0.0507



(a) NB Classifier



(b) SVM Classifier

Fig. 4. Classification accuracy comparison using boxplot on PC5 Dataset.

performance is robust. Table 14 summarizes the mean(AVG), worst, best and the standard deviation (STDEV) of the success rate achieved by the proposed THRO feature selection scheme and other comparative feature selection methods. As per the experimental results presented in Table 14, the proposed feature selection scheme outperformed the other feature selection scheme.

The feature selection ratio of the proposed feature selection method THRO and other comparative methods over JM1 dataset using NB classifier is presented in Fig. 7. It can be noted that, the proposed feature selection method obtained the less feature selection ratio for JM1 dataset. It shows the superiority of the proposed feature selection approach.

5.4. Comparison results analysis on the KC1 dataset

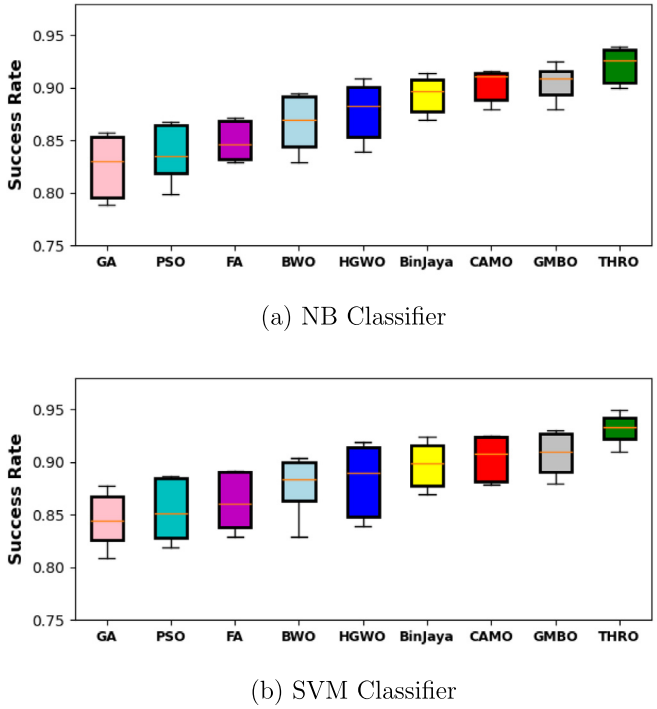
The values of evaluation measures (classification accuracy, error rate, precision, recall, and F_1-score obtained from the proposed THRO and other feature selection algorithms using NB classifier and SVM classifier are presented in Table 15. We can see that proposed algorithm achieved the good classification accuracy for both classifier (NB and SVM) than other algorithms. According to the data presented in Table 15, by using NB classifier, the proposed THRO based software defect prediction model obtained 96.87% of success rate, which is higher than the existing feature selection techniques. By using SVM classifier, the proposed THRO defect prediction method obtained the classification accuracy of 94.93% and the error rate of 5.07%. Also, it has been noted that the proposed algorithm obtained better performance in terms of macro averaged precision, recall and F_1-score. In accordance with the data reported in Table 15, the proposed THRO based software defect prediction algorithm demonstrates superiority over the existing algorithms. Table 16 summarizes the TPR, TNR, FPR, and FNR values achieved by the proposed THRO feature selection scheme and other comparative feature selection methods. As per the experimental results presented in Table 16, the proposed feature selection scheme outperformed the other feature selection scheme. Fig. 8 depicts the classification accuracy score distribution of the proposed THRO based software defect prediction model and other state-of-art methods over 20 separate independent runs on the KC1 dataset. It is evident from the boxplots that the gaps between outer and inner quartiles whiskers are tiny. It is clear that the proposed THRO based defect prediction approach is more reliable than the alternative methods. The comparison results based on the classification accuracy is presented in Table 17. It contains the average, standard deviation, best and worst accuracy score of various feature selection scheme by using NB and SVM classifier. The average classification accuracy of the proposed THRO based feature selection method

(a) NB Classifier (b) SVM Classifier

Fig. 5. Feature selection rate comparison on PC5 Dataset.

Table 14
Results of the repeatability test on JM1 dataset.

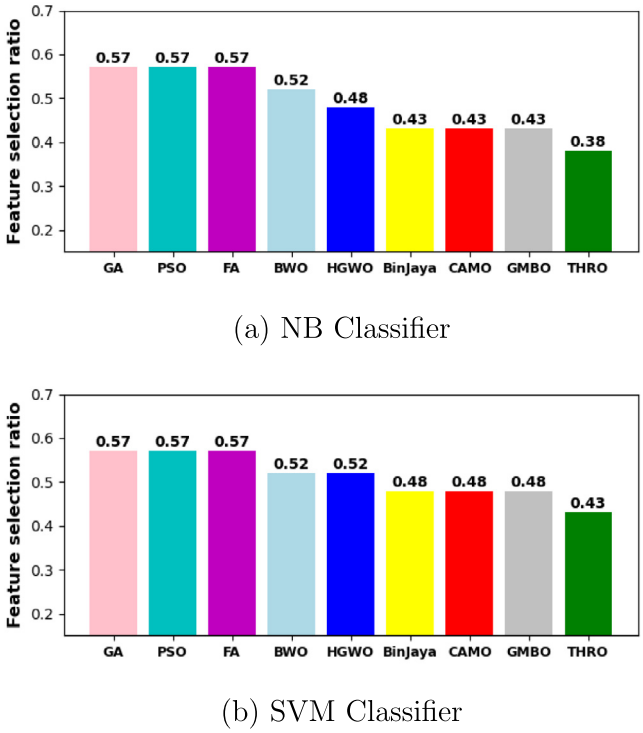
Feature selection methods	NB Classifier accuracy				SVM Classifier accuracy			
	Best	Worst	Avg.	Stdev.	Best	Worst	Avg.	Stdev.
GA	0.8578	0.7891	0.8335	0.0235	0.8778	0.8094	0.8443	0.027
PSO	0.867	0.7996	0.8284	0.028	0.887	0.8193	0.8543	0.0254
FA	0.8713	0.8295	0.8553	0.015	0.8913	0.8292	0.8677	0.0224
BWO	0.8944	0.8291	0.8696	0.0242	0.9043	0.8291	0.8764	0.025
HGWO	0.9093	0.8392	0.8771	0.025	0.9193	0.8394	0.8864	0.0293
BinJaya	0.9138	0.8692	0.8962	0.0175	0.9238	0.8691	0.9012	0.0203
CAMO	0.9158	0.8792	0.9026	0.0141	0.9249	0.8791	0.9039	0.0179
GMBO	0.9249	0.8792	0.907	0.0155	0.9299	0.8792	0.907	0.0186
THRO	0.9393	0.8991	0.9225	0.015	0.9493	0.9092	0.9323	0.0142



(a) NB Classifier

(b) SVM Classifier

Fig. 6. Classification accuracy comparison using boxplot on JM1 Dataset.



(a) NB Classifier

(b) SVM Classifier

Fig. 7. Feature selection rate comparison on JM1 Dataset.

using NB classifier and SVM classifier are 0.9485 and 0.9356 respectively. According to the findings provided in this Table 17, the THRO based feature selection method has obtained the better performance. The feature selection ratio of the proposed feature selection method THRO and other comparative methods over PC5 dataset using NB classifier and SVM classifier are presented in Fig. 9. It can be noted that, the proposed feature selection method obtained the less feature selection ratio. It shows the superiority of the proposed feature selection approach.

5.5. Analysis of variance (ANOVA) statistical test

Analysis of variance (ANOVA) for the average classification accuracy obtained by the proposed feature selection method and other comparative feature selection algorithm on three benchmark dataset using NB classifier is listed in Table 18. The ANOVA results presented in Table 18 showed significant differences in the classification accuracy of feature selection methods. The Table 19 shows the ANOVA statistical test results for the average classification accuracy obtained by the proposed feature selection method and other comparative feature selection algorithm

Table 15
Accuracy comparison on KC1 dataset.

Feature selection methods	NB Classifier accuracy in %					SVM Classifier accuracy in %				
	ACC	Err	P	R	F_1	ACC	Err	P	R	F_1
GA	86.72%	13.28%	0.9062	0.8672	0.8786	86.72%	13.28%	0.9062	0.8672	0.8786
PSO	86.63%	13.37%	0.9058	0.8663	0.8778	87.62%	12.38%	0.9111	0.8762	0.8863
FA	89.05%	10.95%	0.9191	0.8905	0.8987	90.04%	9.96%	0.9248	0.9004	0.9073
BWO	90.37%	9.63%	0.9267	0.9037	0.9103	91.37%	8.63%	0.9327	0.9137	0.919
HGWO	93.84%	6.16%	0.9488	0.9384	0.9412	92.84%	7.16%	0.9422	0.9284	0.9322
BinJaya	93.88%	6.12%	0.9491	0.9388	0.9417	93.31%	6.69%	0.9455	0.9331	0.9365
CAMO	94.4%	5.6%	0.9529	0.944	0.9465	93.5%	6.5%	0.9466	0.935	0.9382
GMBO	95.21%	4.79%	0.9588	0.9521	0.9539	94.12%	5.88%	0.9508	0.9412	0.9439
THRO	96.87%	3.13%	0.9717	0.9687	0.9695	94.93%	5.07%	0.9567	0.9493	0.9513

Table 16
Classifier performance comparison on KC1 dataset.

Feature selection methods	NB Classifier performance				SVM Classifier performance			
	TPR	TNR	FPR	FNR	TPR	TNR	FPR	FNR
GA	0.8676	0.865	0.135	0.1324	0.8676	0.865	0.135	0.1324
PSO	0.8665	0.865	0.135	0.1335	0.8766	0.8742	0.1258	0.1234
FA	0.8906	0.8896	0.1104	0.1094	0.9007	0.8988	0.1012	0.0993
BWO	0.9041	0.9018	0.0982	0.0959	0.9142	0.911	0.089	0.0858
HGWO	0.9389	0.9356	0.0644	0.0611	0.9288	0.9264	0.0736	0.0712
BinJaya	0.9394	0.9356	0.0644	0.0606	0.9333	0.9325	0.0675	0.0667
CAMO	0.9445	0.9417	0.0583	0.0555	0.9355	0.9325	0.0675	0.0645
GMBO	0.9523	0.9509	0.0491	0.0477	0.9417	0.9387	0.0613	0.0583
THRO	0.9692	0.9663	0.0337	0.0308	0.9495	0.9479	0.0521	0.0505

Table 17
Results of the repeatability test on KC1 dataset.

Feature selection methods	NB Classifier accuracy				SVM Classifier accuracy			
	Best	Worst	Avg.	Stdev.	Best	Worst	Avg.	Stdev.
GA	0.8672	0.7862	0.8371	0.0276	0.8672	0.7956	0.8377	0.0246
PSO	0.8663	0.7971	0.8383	0.0233	0.8762	0.8051	0.8449	0.0262
FA	0.8905	0.8288	0.8649	0.0214	0.9004	0.8279	0.8662	0.0253
BWO	0.9037	0.826	0.8687	0.0268	0.9137	0.8255	0.8776	0.0305
HGWO	0.9384	0.8554	0.8999	0.0297	0.9284	0.8559	0.8902	0.0257
BinJaya	0.9388	0.8876	0.916	0.0181	0.9331	0.8853	0.9124	0.016
CAMO	0.944	0.8962	0.9193	0.0177	0.935	0.8853	0.9113	0.0177
GMBO	0.9521	0.8952	0.9225	0.0201	0.9412	0.8952	0.9228	0.0152
THRO	0.9687	0.9222	0.9485	0.0154	0.9493	0.9218	0.9356	0.0083

Table 18
ANOVA for average success rate using NB classifier.

Sources	DF	SS	MS	F	P
Algorithm	8	0.030242	0.00378025	117.6731518	9.629785e−13
Dataset	2	0.000878	0.000439	13.66536965	3.456715e−04
Residual	16	0.000514	0.000032125		
Total	26	0.031634			

Table 19
ANOVA for average success rate using SVM classifier.

Sources	DF	SS	MS	F	P
Algorithm	8	0.025025	0.003128125	128.0051151	4.924706e−13
Dataset	2	0.000088	0.000044	1.800511509	1.965186e−01
Residual	16	0.000391	2.44375e−05		
Total	26	0.025504			

on three benchmark dataset using SVM classifier. It is clearly shows that the superiority of the proposed THRO based software defect prediction model. Finally, the Fisher's least significant difference(LSD) test between various pairwise feature selection method is presented in Table 20. The p-value of each pair of feature selection method in terms of average classification accuracy is presented in Table 20. As per the statistical test, the lower p-value (less than 5%) of the test indicates that the improvement has not occurred by chance. It can be observed from the p-values that, the proposed feature selection scheme THRO shows significant differences in classification accuracy compared with other feature selection methods.

The p-value of each pair of feature selection method in terms of average classification accuracy is presented in Table 21. It can be observed from the p-values that, the proposed feature selection scheme THRO shows significant differences in classification accuracy compared with other feature selection methods.

5.6. Significant software metrics for defect prediction

An in-depth results analysis clearly shows that LOC (m_1), cyclomatic complexity (m_2), essential complexity (m_3), design complexity (m_4), difficulty (m_8), and effort measures (m_{10}) are significant metrics for software defect prediction.

6. Validity threats

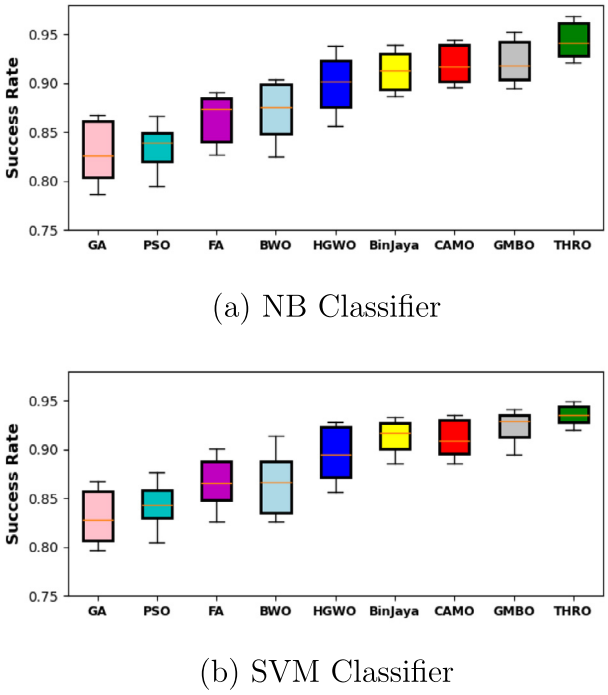
The validity challenges to our proposed feature selection scheme are discussed in this section. Even if the performance of the proposed THRO based feature subset selection method is superior to that of the alternative scheme, the proposed method requires more computing time. Because the proposed work uses five filter-based approaches at the first stage, the TOPSIS method is employed to select the best features, and finally, the wrapper based approach is applied to select the optimal feature subset, which incurs some additional computational cost.

Table 20
p-value of average success rate using NB classifier.

Methods	GA	PSO	FA	BWO	HGWO	BinJaya	CAMO	GMBO	THRO
GA	1	0.75941	0.00149	0.00039	0.00236	0.00028	0.00012	0.00007	0.0002
PSO	0.75941	1	0.02248	0.00529	0.00591	0.00097	0.00054	0.0004	0.00045
FA	0.00149	0.02248	1	0.04103	0.02781	0.00186	0.00077	0.00048	0.00069
BWO	0.00039	0.00529	0.04103	1	0.15133	0.00597	0.00231	0.00138	0.00139
HGWO	0.00236	0.00591	0.02781	0.15133	1	0.08488	0.04069	0.02623	0.0082
BinJaya	0.00028	0.00097	0.00186	0.00597	0.08488	1	0.55509	0.32813	0.03507
CAMO	0.00012	0.00054	0.00077	0.00231	0.04069	0.55509	1	0.64648	0.05014
GMBO	0.00007	0.0004	0.00048	0.00138	0.02623	0.32813	0.64648	1	0.06942
THRO	0.0002	0.00045	0.00069	0.00139	0.0082	0.03507	0.05014	0.06942	1

Table 21
p-value of average success rate using SVM classifier.

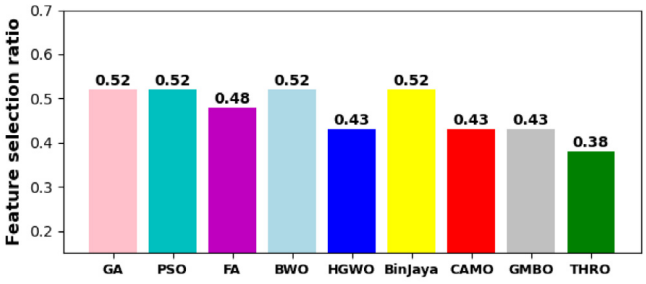
Methods	GA	PSO	FA	BWO	HGWO	BinJaya	CAMO	GMBO	THRO
GA	1	0.04297	0.00131	0.00013	0.00006	0.0001	0.00007	0.00017	0.00001
PSO	0.04297	1	0.02569	0.00157	0.00047	0.00034	0.00024	0.00043	0.00002
FA	0.00131	0.02569	1	0.00501	0.00067	0.00053	0.00034	0.00071	0.00001
BWO	0.00013	0.00157	0.00501	1	0.00389	0.00147	0.00084	0.00176	0.00001
HGWO	0.00006	0.00047	0.00067	0.00389	1	0.00883	0.00489	0.00652	0.00003
BinJaya	0.0001	0.00034	0.00053	0.00147	0.00883	1	0.78719	0.25215	0.00303
CAMO	0.00007	0.00024	0.00034	0.00084	0.00489	0.78719	1	0.3148	0.00268
GMBO	0.00017	0.00043	0.00071	0.00176	0.00652	0.25215	0.3148	1	0.02961
THRO	0.00001	0.00002	0.00001	0.00001	0.00003	0.00303	0.00268	0.02961	1



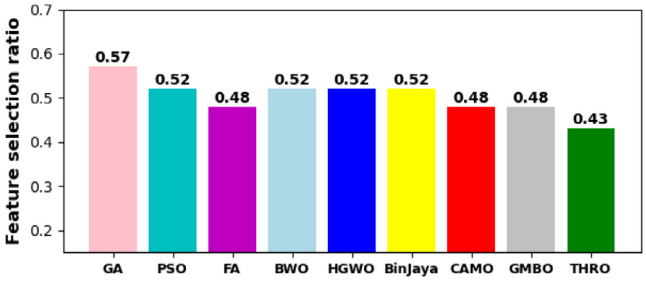
(a) NB Classifier

(b) SVM Classifier

Fig. 8. Classification accuracy comparison using boxplot on KC1 Dataset.



(a) NB Classifier



(b) SVM Classifier

Fig. 9. Feature selection rate comparison on KC1 Dataset.

7. Conclusion

A hybrid software defect prediction approach based on TOPSIS and hybrid Rao optimization algorithm has been proposed in this work. The proposed method improves the search ability towards the global optimal solution by incorporating crossover and mutation into the Rao optimization technique. The proposed work has been compared with six meta-heuristic based feature selection techniques. For performance comparison, three standard benchmark datasets were used. An in-depth results analysis clearly shows that the proposed THRO based feature selection algorithm outperforms the other strategies. The proposed method can be further improved by using the built-in parallelism techniques. The future direction will focus more on improving the proposed method using deep learning techniques.

CRediT authorship contribution statement

Karpagalingam Thirumoorthy: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing – original draft. **Jerold John Britto J.:** Resources, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Dataset available in public repository. We have provided the link in the manuscript dataset section.

Acknowledgements

The authors would like to thank the Management and Principal of Mepco Schlenk Engineering College (Autonomous), Sivakasi for providing us the state of art facilities to carry out this proposed research work in the Mepco Research Centre in collaboration with Anna University Chennai, Tamil Nadu, India. All authors approved the final version of the manuscript.

References

[1] Z. Sun, J. Zhang, H. Sun, X. Zhu, Collaborative filtering based recommendation of sampling methods for software defect prediction, Appl. Soft Comput. 90 (2020) 106163, http://dx.doi.org/10.1016/j.asoc.2020.106163.

[2] Ö.F. Arar, K. Ayan, A feature dependent naive Bayes approach and its application to the software defect prediction problem, Appl. Soft Comput. 59 (2017) 197–209, http://dx.doi.org/10.1016/j.asoc.2017.05.043.

[3] M. Rawat, S. Dubey, Software defect prediction models for quality improvement: A literature study, Int. J. Comput. Sci. Issues 9 (2012) 288–296.

[4] Z. Wan, X. Xia, A.E. Hassan, D. Lo, J. Yin, X. Yang, Perceptions, expectations, and challenges in defect prediction, IEEE Trans. Softw. Eng. 46 (11) (2020) 1241–1266, http://dx.doi.org/10.1109/TSE.2018.2877678.

[5] M.A. Kabir, J. Keung, B. Turhan, K.E. Bennin, Inter-release defect prediction with feature selection using temporal chunk-based learning: An empirical study, Appl. Soft Comput. 113 (2021) 107870, http://dx.doi.org/10.1016/j.asoc.2021.107870.

[6] J.R. Vergara, P.A. Estévez, A review of feature selection methods based on mutual information, Neural Comput. Appl. 24 (1) (2014) 175–186, http://dx.doi.org/10.1007/s00521-013-1368-0.

[7] G. Chen, J. Chen, A novel wrapper method for feature selection and its applications, Neurocomputing 159 (2015) 219–226, http://dx.doi.org/10.1016/j.neucom.2015.01.070, URL https://www.sciencedirect.com/science/article/pii/S0925231215001459.

[8] J. Han, M. Kamber, J. Pei, Data Mining Concepts and Techniques, third ed., Morgan Kaufmann Publishers, 2012.

[9] A. Chug, S. Dhall, Software defect prediction using supervised learning algorithm and unsupervised learning algorithm, in: Confluence 2013: The Next Generation Information Technology Summit (4th International Conference), 2013, pp. 173–179, http://dx.doi.org/10.1049/cp.2013.2313.

[10] A. Rahim, Z. Hayat, M. Abbas, A. Rahim, M.A. Rahim, Software defect prediction with Naïve Bayes classifier, in: 2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST), 2021, pp. 293–297, http://dx.doi.org/10.1109/IBCAST51254.2021.9393250.

[11] S. Pandey, R. Mishra, A. Tripathi, Software bug prediction prototype using Bayesian network classifier: A comprehensive model, Procedia Comput. Sci. 132 (2018) 1412–1421, http://dx.doi.org/10.1016/j.procs.2018.05.071.

[12] H. Aljamaan, A. Alazba, Software Defect Prediction Using Tree-Based Ensembles, in: PROMISE 2020, Association for Computing Machinery, New York, NY, USA, 2020, pp. 1–10, http://dx.doi.org/10.1145/3416508.3417114.

[13] S. Rathore, S. Kumar, A decision tree logic based recommendation system to select software fault prediction techniques, Computing 99 (2017) 255–285, http://dx.doi.org/10.1007/s00607-016-0489-6.

[14] M. Hammad, A. Alqaddoumi, H. Alobaidy, K. Almseidein, Predicting software faults based on K-nearest neighbors classification, Int. J. Comput. Digit. Syst. 8 (2019) 461–467, http://dx.doi.org/10.12785/ijcds/080503.

[15] M. Thangavel, G. Nasira, Support vector machine for software defect prediction, Int. J. Appl. Eng. Res. 9 (2014) 25633–25644.

[16] X. Rong, F. Li, Z. Cui, A model for software defect prediction using support vector machine based on CBA, Int. J. Intell. Syst. Technol. Appl. 15 (2016) 19, http://dx.doi.org/10.1504/IJISTA.2016.076102.

[17] P. Bishnu, V. Bhattacharjee, Software fault prediction using quad tree-based K-means clustering algorithm, IEEE Trans. Knowl. Data Eng. 24 (2012) 1146–1150, http://dx.doi.org/10.1109/TKDE.2011.163.

[18] M. Park, E. Hong, Software fault prediction model using clustering algorithms determining the number of clusters automatically, Int. J. Softw. Eng. Appl. 8 (2014) 199–204, http://dx.doi.org/10.14257/ijseia.2014.8.7,16.

[19] G. Abaei, A. Selamat, Increasing the accuracy of software fault prediction using majority ranking fuzzy clustering, Stud. Comput. Intell. 569 (2015) 179–193, http://dx.doi.org/10.1007/978-3-319-10389-113.

[20] M. Park, E. Hong, Software fault prediction model using clustering algorithms determining the number of clusters automatically, Int. J. Softw. Eng. Appl. 8 (2014) 199–204, http://dx.doi.org/10.14257/ijseia.2014.8.7,16.

[21] S.K. Pandey, R.B. Mishra, A.K. Tripathi, Machine learning based methods for software fault prediction: A survey, Expert Syst. Appl. 172 (2021) 114595, http://dx.doi.org/10.1016/j.eswa.2021.114595.

[22] A. Alsaeedi, M. Khan, Software defect prediction using supervised machine learning and ensemble techniques: A comparative study, J. Softw. Eng. Appl. 12 (2019) 85–100, http://dx.doi.org/10.4236/jsea.2019.125007.

[23] S. Pandey, R. Mishra, A. Tripathi, BPDET: An effective software bug prediction model using deep representation and ensemble learning techniques, Expert Syst. Appl. 144 (2019) 113085, http://dx.doi.org/10.1016/j.eswa.2019.113085.

[24] M. Assim, Q. Obeidat, M. Hammad, Software defects prediction using machine learning algorithms, in: 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI), 2020, pp. 1–6, http://dx.doi.org/10.1109/ICDABI51230.2020.9325677.

[25] S. Nur, K. Wei, S. Kew, Machine learning techniques for software bug prediction: A systematic review, J. Comput. Sci. 16 (2020) 1558–1569, http://dx.doi.org/10.3844/jcssp.2020.1558.1569.

[26] M. Vikas, K. Prabhpreet, Lung cancer detection using chi-square feature selection and support vector machine algorithm, Int. J. Adv. Trends Comput. Sci. Eng. 10 (3) (2021) 2050–2061, http://dx.doi.org/10.30534/ijatcse/2021/801032021.

[27] L. Li, H. Liu, Z. Ma, Y. Mo, Z. Duan, J. Zhou, J. Zhao, Multi-label feature selection via information gain, in: X. Luo, J.X. Yu, Z. Li (Eds.), Advanced Data Mining and Applications, Springer International Publishing, Cham, 2014, pp. 345–355.

[28] S. Cang, H. Yu, Mutual information based input feature selection for classification problems, Decis. Support Syst. 54 (1) (2012) 691–698, http://dx.doi.org/10.1016/j.dss.2012.08.014, URL https://www.sciencedirect.com/science/article/pii/S0167923612002291.

[29] A.M. Kowshalya, R. Madhumathi, N. Gopika, Correlation based feature selection algorithms for varying datasets of different dimensionality, Wirel. Pers. Commun. 108 (3) (2019) 1977–1993, http://dx.doi.org/10.1007/s11277-019-06504-w.

[30] Y. Khourdifi, M. Bahaj, Feature selection with fast correlation-based filter for breast cancer prediction and classification using machine learning algorithms, in: 2018 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), 2018, pp. 1–6, http://dx.doi.org/10.1109/ISAECT.2018.8618688.

[31] E. Hancer, B. Xue, M. Zhang, Differential evolution for filter feature selection based on information theory and feature ranking, Knowl.-Based Syst. 140 (2018) 103–119, http://dx.doi.org/10.1016/j.knosys.2017.10.028, URL https://www.sciencedirect.com/science/article/pii/S0950705117304987.

[32] R.J. Urbanowicz, M. Meeker, W. La Cava, R.S. Olson, J.H. Moore, Relief-based feature selection: Introduction and review, J. Biomed. Inform. 85 (2018) 189–203, http://dx.doi.org/10.1016/j.jbi.2018.07.014, URL https://www.sciencedirect.com/science/article/pii/S1532046418301400.

[33] M. Bennasar, Y. Hicks, R. Setchi, Feature selection using joint mutual information maximisation, Expert Syst. Appl. 42 (22) (2015) 8520–8532, http://dx.doi.org/10.1016/j.eswa.2015.07.007, URL https://www.sciencedirect.com/science/article/pii/S0957417415004674.

[34] J. Liang, L. Hou, Z. Luan, W. Huang, Feature selection with conditional mutual information considering feature interaction, Symmetry 11 (7) (2019) http://dx.doi.org/10.3390/sym11070858, URL https://www.mdpi.com/2073-8994/11/7/858.

[35] K. Kaur, N. Patil, A fast and novel approach based on grouping and weighted mRMR for feature selection and classification of protein sequence data, Int. J. Data Min. Bioinf. 23 (1) (2020) 47–61, http://dx.doi.org/10.1504/IJDMB.2020.105435, arXiv:https://www.inderscienceonline.com/doi/pdf/10.1504/IJDMB.2020.105435.

[36] K. Thirumoorthy, K. Muneeswaran, Optimal feature subset selection using hybrid binary Jaya optimization algorithm for text classification, Sādhanā 45 (1) (2020) 1–13, http://dx.doi.org/10.1007/s12046-020-01443-w.

[37] C. Gunavathi, K. Premalatha, Performance analysis of genetic algorithm with kNN and SVM for feature selection in tumor classification, Int. J. Comput. Inf. Eng. 8 (8) (2014) 1490–1497.

[38] R. Najeeb, B.N. Dhannoon, A feature selection approach using binary firefly algorithm for network intrusion detection system, ARPN J. Eng. Appl. Sci. 13 (2018) 2347–2352.

[39] R. Malhotra, A. Shakya, R. Ranjan, R. Banshi, Software defect prediction using binary particle swarm optimization with binary cross entropy as the fitness function, J. Phys. Conf. Ser. 1767 (1) (2021) 1–10, http://dx.doi.org/10.1088/1742-6596/1767/1/012003.
[40] T. Khuat, L. Thi My Hanh, Binary teaching-learning based optimization algorithm with a new update mechanism for sample subset optimization in software defect prediction, Soft Comput. (2019) 1–17, http://dx.doi.org/10.1007/s00500-018-3546-6.
[41] C. Manjula, L. Florence, Hybrid approach for software defect prediction using machine learning with optimization technique, Int. J. Comput. Inf. Eng. 12 (1) (2018) 28–32.
[42] R.A. Khurma, H. Alsawalqah, I. Aljarah, M.A. Elaziz, R. Damasevicius, An enhanced evolutionary software defect prediction method using island moth flame optimization, Mathematics 9 (15) (2021) http://dx.doi.org/10.3390/math9151722.
[43] R. Malhotra, N. Nishant, S. Gurha, V. Rathi, Application of particle swarm optimization for software defect prediction using object oriented metrics, in: 2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence), 2021, pp. 88–93, http://dx.doi.org/10.1109/Confluence51648.2021.9377116.
[44] M. Panda, A. Azar, Hybrid multi-objective Grey Wolf search optimizer and machine learning approach for software bug prediction: Hybrid multi-objective Grey Wolf search optimizer for software bug prediction, 2020, pp. 1–24, http://dx.doi.org/10.4018/978-1-7998-5788-4.
[45] M. Anbu, G.S. Anandha Mala, Feature selection using firefly algorithm in software defect prediction, Cluster Comput. 22 (2019) 10925–10934, http://dx.doi.org/10.1007/s10586-017-1235-3.
[46] B. Kiran Kumar, G. Jayadev Gyani, G. Narsimha, Software defect prediction using ant colony optimization, Int. J. Appl. Eng. Res. 13 (2018) 14291–14297.
[47] I. Emovon, O.S. Oghenenyerovwho, Application of MCDM method in material selection for optimal design: A review, Results Mater. 7 (2020) 100115, http://dx.doi.org/10.1016/j.rinma.2020.100115, URL https://www.sciencedirect.com/science/article/pii/S2590048X20300571.
[48] A.S.K. Kannan, S.A.A. Balamurugan, S. Sasikala, A customized metaheuristic approaches for improving supplier selection in intelligent decision making, IEEE Access 9 (2021) 56228–56239, http://dx.doi.org/10.1109/ACCESS.2021.3071454.
[49] M. Abdel-Basset, M. Saleh, A. Gamal, F. Smarandache, An approach of TOPSIS technique for developing supplier selection with group decision making under type-2 neutrosophic number, Appl. Soft Comput. 77 (2019) 438–452, http://dx.doi.org/10.1016/j.asoc.2019.01.035, URL https://www.sciencedirect.com/science/article/pii/S1568946619300419.
[50] A.S.K. Kannan, S.A.a. Balamurugan, S. Sasikala, A novel software package selection method using teaching–learning based optimization and multiple criteria decision making, IEEE Trans. Eng. Manage. 68 (4) (2021) 941–954, http://dx.doi.org/10.1109/TEM.2019.2918050.
[51] S.C. Nayak, C. Tripathy, Deadline based task scheduling using multi-criteria decision-making in cloud environment, Ain Shams Eng. J. 9 (4) (2018) 3315–3324, http://dx.doi.org/10.1016/j.asej.2017.10.007, URL https://www.sciencedirect.com/science/article/pii/S2090447917301417.
[52] M.S. Kumar, A. Tomar, P.K. Jana, Multi-objective workflow scheduling scheme: a multi-criteria decision making approach, J. Ambient Intell. Humaniz. Comput. 12 (12) (2021) 10789–10808, http://dx.doi.org/10.1007/s12652-020-02833-y.
[53] I. Grgurevic, G. Kordic, Multi-criteria decision-making in cloud service selection and adoption, 2017, pp. 8–12, http://dx.doi.org/10.18638/rcitd.2017.5.1.104.
[54] G. Kou, Y. LU, Y. Peng, Y. Shi, Evaluation of classification algorithms using MCDM and rank correlation, Int. J. Inf. Technol. Decis. Mak. 11 (2012) 197–225, http://dx.doi.org/10.1142/S0219622012500095.
[55] R. Singh, H. Kumar, R. Singla, TOPSIS based multi-criteria decision making of feature selection techniques for network traffic dataset, Int. J. Eng. Technol. 5 (2013) 4598–4604.
[56] G. Kou, P. Yang, Y. Peng, F. Xiao, Y. Chen, F. Alsaadi, Evaluation of feature selection methods for text classification with small datasets using multiple criteria decision-making methods, Appl. Soft Comput. 86 (2019) 105836, http://dx.doi.org/10.1016/j.asoc.2019.105836.
[57] A. Hashemi, M.B. Dowlatshahi, H. Nezamabadi-pour, MFS-MCDM: Multi-label feature selection using multi-criteria decision making, Knowl.-Based Syst. 206 (2020) 106365, http://dx.doi.org/10.1016/j.knosys.2020.106365, URL https://www.sciencedirect.com/science/article/pii/S0950705120305116.
[58] A. Hashemi, M.B. Dowlatshahi, H. Nezamabadi-pour, Ensemble of feature selection algorithms: a multi-criteria decision-making approach, Int. J. Mach. Learn. Cybern. (2021) http://dx.doi.org/10.1007/s13042-021-01347-z.
[59] H. Ching-Lai, Y. Kwangsun, Methods for multiple attribute decision making, 1981, pp. 58–191, http://dx.doi.org/10.1007/978-3-642-48318-9_3.
[60] R. Venkata Rao, Rao algorithms: Three metaphor-less simple algorithms for solving optimization problems, Int. J. Ind. Eng. Comput. (2020) 107–130, http://dx.doi.org/10.5267/j.ijiec.2019.6.002.
[61] I.M.D. Maysanjaya, I.M.A. Pradnyana, I.M. Putrama, Classification of breast cancer using Wrapper and Naïve Bayes algorithms, J. Phys. Conf. Ser. 1040 (2018) 012017, http://dx.doi.org/10.1088/1742-6596/1040/1/012017.
[62] H. Kamel, D. Abdulah, J.M. Al-Tuwaijari, Cancer classification using Gaussian naive Bayes algorithm, in: 2019 International Engineering Conference (IEC), 2019, pp. 165–170, http://dx.doi.org/10.1109/IEC47844.2019.8950650.
[63] R.B. Bahaweres, A. Imam Suroso, A. Wahyu Hutomo, I. Permana Solihin, I. Hermadi, Y. Arkeman, Tackling feature selection problems with genetic algorithms in software defect prediction for optimization, in: 2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS), 2020, pp. 64–69, http://dx.doi.org/10.1109/ICIMCIS51567.2020.9354282.
[64] R. Malhotra, N. Nishant, S. Gurha, V. Rathi, Application of particle swarm optimization for software defect prediction using object oriented metrics, in: 2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence), 2021, pp. 88–93, http://dx.doi.org/10.1109/Confluence51648.2021.9377116.
[65] I. Arora, A. Saha, Software fault prediction using firefly algorithm, Int. J. Intell. Eng. Inf. 6 (2018) 356–377, http://dx.doi.org/10.1504/IJIEI.2018.10013012.
[66] Y. Hassouneh, H. Turabieh, T. Thaher, I. Tumar, H. Chantar, J. Too, Boosted whale optimization algorithm with natural selection operators for software fault prediction, IEEE Access 9 (2021) 14239–14258, http://dx.doi.org/10.1109/ACCESS.2021.3052149.
[67] M. Panda, A. Azar, Hybrid multi-objective Grey Wolf search optimizer and machine learning approach for software bug prediction: Hybrid multi-objective Grey Wolf search optimizer for software bug prediction, 2020, http://dx.doi.org/10.4018/978-1-7998-5788-4.
[68] M.A. Awadallah, M.A. Al-Betar, A.I. Hammouri, O.A. Alomari, Binary JAYA algorithm with adaptive mutation for feature selection, Arab. J. Sci. Eng. 45 (12) (2020) 10875–10890, http://dx.doi.org/10.1007/s13369-020-04871-2.
[69] M. Dehghani, S. Hubalovsky, P. Trojovsky, Cat and mouse based optimizer: A new nature-inspired optimization algorithm, Sensors 21 (15) (2021) http://dx.doi.org/10.3390/s21155214, URL https://www.mdpi.com/1424-8220/21/15/5214.
[70] M. Dehghani, Z. Montazeri, S. Hubalovsky, GMBO: Group mean-based optimizer for solving various optimization problems, Mathematics 9 (11) (2021) http://dx.doi.org/10.3390/math9111190, URL https://www.mdpi.com/2227-7390/9/11/1190.