Knowledge-Based Systems 301 (2024) 112345

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Feature selection using a classification error impurity algorithm and an adaptive genetic algorithm improved with an external repository

Hossein Nematzadeh [a,b,*], José García-Nieto [a,b,c], Ismael Navas-Delgado [a,b,c], José F. Aldana-Montes [a,b,c]

[a] ITIS Software, Universidad de Málaga, Arquitecto Francisco Peñalosa 18, Málaga, 29071, Spain
[b] Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Málaga, Spain
[c] Biomedical Research Institute of Málaga (IBIMA), Universidad de Málaga, Málaga, Spain

ARTICLE INFO

Keywords:
Feature selection
High-dimensional
Filter
Wrapper
Genetic Algorithm

ABSTRACT

Feature selection in small-sample high-dimensional datasets enhances classification accuracy and reduces computational time for model training. This paper introduces the filter Classification Error Impurity (CEI) as a frequency-based ranker that improves upon existing methods by better identifying non-linear patterns. According to this, the top features identified by the ensemble of CEI, along with Mutual Information (MI) and Fisher Ratio (FR), form the feature space utilized by the Adaptive Genetic Algorithm with External Repository (AGAwER) to identify the optimal feature combination in a wrapper approach. AGAwER leverages an external repository, incorporating the best solutions to enrich the Genetic Algorithm's (GA) population, thus promoting diversity and enhancing exploration. As a result, a hybrid method called CMF-AGAwER is proposed, which surpasses existing modern feature selection methods. The implementation and data are accessible on GitHub at https://github.com/KhaosResearch/CMF-AGAwER.

1. Introduction

Feature selection plays a critical role in scenarios where there is limited data with high dimensionality. In such cases, domain experts aim to identify the most informative features [1–3]. This process not only reduces computational time during model training but also enhances prediction accuracy when utilizing the selected features [4–6]. There are three fundamental feature selection methods: filter [7], wrapper [8,9], and embedded [6,10]. Subsequently, two additional categories have been introduced: ensemble [11,12] and hybrid [13,14]. Hybrid feature selection combines different techniques within a single framework, while ensemble feature selection aggregates the outputs of multiple feature selection models or algorithms to arrive at a consensus. Both approaches aim to improve the quality and effectiveness of feature selection in machine learning tasks.

A frequency-based feature selection approach has recently emerged for classification problems, encompassing various methods that can be employed as filters to rank dataset features. Frequency-based feature selection methods evaluate the importance of features based on the frequency of classes in the response variable. These methods involve sorting feature values and adjusting the order of classes accordingly in the response variable. This reordered response variable is then used to calculate a measure that indicates the importance of each feature in the dataset. Frequency-based feature selection methods are specifically designed for classification datasets. [11,13–17]. However, existing frequency-based rankers often struggle to accurately identify non-linear patterns. Linear patterns involve straight-line relationships between data, while non-linear patterns encompass more complex and curved relationships that cannot be simplified to a straight line. Therefore, the primary objective of this research is to introduce a new frequency-based ranking algorithm, named Classification Error Impurity (CEI), which is capable of effectively recognizing both linear and non-linear patterns simultaneously.

Meta-heuristic algorithms, including Genetic Algorithms (GAs) [18,19], have been widely employed for selecting optimal features, especially in the development of wrapper feature selection methods. Unlike systematic search methods that exhaustively explore the entire search space, these algorithms employ intelligent local search strategies to approximate the optimal set of features through optimization. In

* Corresponding author at: ITIS Software, Universidad de Málaga, Arquitecto Francisco Peñalosa 18, Málaga, 29071, Spain.
E-mail addresses: hnematzadeh@uma.es, hn_61@yahoo.com (H. Nematzadeh), jnieto@uma.es (J. García-Nieto), ismael@uma.es (I. Navas-Delgado), jfaldana@uma.es (J.F. Aldana-Montes).

https://doi.org/10.1016/j.knosys.2024.112345
Received 5 June 2024; Received in revised form 23 July 2024; Accepted 3 August 2024
Available online 8 August 2024
0950-7051/© 2024 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

GAs, several techniques are employed to enhance exploration, which is crucial for discovering diverse regions of the search space and avoiding premature convergence to suboptimal solutions including adjusting crossover and mutation rates, as well as increasing population size. However, they have their own drawbacks and side effects as well. For example, increasing the population size in GAs does not guarantee that the new solutions will be sufficiently diverse compared to existing solutions. While a larger population size can generally lead to greater diversity, it is not a definitive solution to ensuring diversity among solutions. As a result, increasing the population might add some identical solutions to the initial population. In this paper, we introduce the concept of an external repository and integrate it into our proposed GA. The external repository is responsible for inserting sufficiently diverse solutions with acceptable quality into the GA. The external repository contains a set of features that do not appear in any of the solutions in GA population. Diverse solutions are generated in the external repository using these unique features. Additionally, the external repository follows a strategy to ensure that each diverse solution is sufficiently different from the existing solutions in GA population. This strategy involves numerically calculating the distance of each diverse solution and guaranteeing that this distance exceeds a predefined threshold. The external repository iteratively generates a set of diverse solutions and ultimately selects the best one using a tournament selection process, then incorporates it into the GA population. Therefore, the external repository also ensures the quality of the diverse solution. This approach can enhance the chance of exploration within the GA.

In summary, this paper addresses the identified gaps through the following contributions:

- Development of a Classification Error Impurity (CEI) algorithm as a frequency-based ranker capable of effectively recognizing both linear and non-linear patterns.
- Development of an Adaptive Genetic Algorithm with External Repository (AGAwER) to enhance the exploration of GA by integrating sufficiently diverse solutions into the GA population.
- Creation of a hybrid feature selection method (CMF[1]-AGAwER) by combining the ensemble of three filters namely, CEI, Mutual Information (MI), and Fisher Ratio (FR) with the wrapper AGAwER. We have also explored the application of SHapley Additive exPlanations (SHAP) as an explainable AI method in the context of filter feature selection on classification datasets.

The remainder of this paper is structured as follows. Section 2 provides an overview of the history of frequency-based rankers and discusses recent hybrid or ensemble feature selection methods, highlighting their widespread adoption of evolutionary algorithms. Section 3 conducts a numerical investigation to compare existing frequency-based rankers and reveals the absence of a frequency-based ranker capable of effectively recognizing non-linear patterns. Section 4 introduces the proposed method, CMF-AGAwER. Section 5 presents the results and comparisons. Finally, Section 6 concludes the paper by summarizing the findings and contributions, and highlighting future work.

2. Related works

This section begins by tracing the evolution of frequency-based feature selection rankers, starting from their initial attempts as discussed in Section 2.1. Subsequently, a range of contemporary feature selection techniques is presented in Section 2.2. Towards the conclusion of Section 2.2, the deficiencies in current methods are pinpointed, which are addressed in this paper.

[1] CMF is an abbreviation formed from the initial letters of three filters: Classification Error Impurity, Mutual Information, and Fisher Ratio.

2.1. *Evolution of frequency-based rankers*

According to Table 1, the first frequency-based ranker traces back to Mutual Congestion (MC) [16] specifically tailored for classification datasets with binary response variable. Building upon this groundwork, Sorted Label Interference (SLI) [11] was subsequently devised, inspired by the principles underlying MC. Likewise, SLI-γ [15] was introduced as an extension, maintaining applicability to binary response variables. However, it exhibited sensitivity towards one of the classes, distinguishing it from its predecessors. The primary constraint of MC, SLI, and SLI-γ was their assumption of linear data distribution. Consequently, they struggled to identify informative features exhibiting non-linear patterns. Therefore, Extended Mutual Congestion (EMC) and Maximum Pattern Recognition (MPR) were developed to overcome this limitation, resulting in improved capability to identify features with non-linear patterns. The latest frequency-based ranking algorithm proposed was the linear Distance-based Mutual Congestion (DMC) [17]. What sets DMC apart from its predecessors is its unique incorporation of feature values alongside feature frequency in its formulation. The reported time complexity in Table 1 is based on the assumptions that m, n, and l represent the number of columns, rows, and class labels, respectively. In summary, no frequency-based ranking algorithm has been able to effectively address non-linearity, which forms the foundation of our primary contribution in this paper. By “effectively”, we refer to the ability of the frequency-based ranking algorithm to identify features with non-linear patterns. Previous methods did not assign high weights to these types of features, leading to their exclusion by the ranker. However, certain classifiers, such as neural networks, can enhance classification accuracy by utilizing these features.

2.2. *Related feature selection methods*

The literature also indicates that filter feature selection methods are frequently integrated within hybrid or ensemble approaches, as they often do not yield significant improvements in accuracy when employed in isolation [14]. In the following we introduce some recent feature selection methods tailored for small-sample high-dimensional datasets, integrating filter rankers.

Whale Optimization Algorithm-Mutual Congestion (WOA-MC) [16] was a hybrid of two filter methods. Initially, the proposed filter WOA discarded half of the non-informative features, while the remaining features underwent sorting based on the filter frequency-based Mutual Congestion (MC). The final selection of features was executed using an intelligent forward feature selection technique, which involved majority voting among the 10 subsets of features created from the top 10 features achieved through MC. Automatic Thresholding Feature Selection (ATFS) [11] leveraged the filter SLI frequency-based ranker inspired by MC. ATFS initially ranked the features of the small-sample high-dimensional datasets using SLI, MC [16], and ReliefF. Subsequently, the results from the three rankers were ensembled using the concept of non-dominated sorting. ATFS was specifically designed for binary datasets and demonstrated enhanced classifier accuracy on both medical high and non-high-dimensional datasets.

$GA_{rank\&rand}$ [15] integrated the filter frequency-based SLI-γ, a variant of SLI, with the Genetic Algorithm (GA). It was demonstrated that $GA_{rank\&rand}$ attained high accuracy particularly when the GA’s initial population was constructed using the most pertinent features identified by SLI-γ.

Extended Mutual Congestion-Discrete Weighted Evolution Strategy (EMC-DWES) [13] operated as a two-stage hybrid technique. Initially, the filter frequency-based EMC ranker was employed to eliminate 95% of the less informative features. Subsequently, the wrapper DWES utilized the remaining features to automatically select the optimal subset. DWES featured two hyperparameters requiring expert tuning: the type of linkages in hierarchical clustering (single, average, or complete), and

Table 1
Comparison of frequency-based rankers for classification problems.

Frequency-based rankers	Abbreviation	Application	Time complexity	Sensitivity to response variable	Linear assumption of data distribution	Year
Mutual congestion [16]	MC	Binary	$O(mn\log n)$	No	Yes	2019
Sorted label interference [11]	SLI	Binary	$O(mn\log n)$	No	Yes	2021
Sorted label interference-γ [15]	SLI-γ	Binary	$O(mn\log n)$	Yes	Yes	2022
Extended mutual congestion [13]	EMC	Multiclass	$O(mn\log n + mln)$	No	No	2022
Maximum pattern recognition [14]	MPR	Multiclass	$O(mn\log n)$	No	No	2024
Distance-based mutual congestion [17]	DMC	Binary	$O(mn\log n)$	No	Yes	2024

the number of clusters. EMC-DWES demonstrated enhanced accuracy in small-sample high-dimensional medical datasets.

Maximum Pattern Recognition-Multi-objective Discrete Evolution Strategy (MPR-MDES) [14] represented an advancement over EMC-DWES. It offered multiple optimal subsets of varying sizes to experts. Similarly, the wrapper MDES utilized fewer hyperparameters compared to DWES and demonstrated enhancements in subset length and prediction accuracy.

Distance-based Mutual Congestion Genetic Algorithm with Adaptive Rates (DMC-GAwAR) [17] merged the filter frequency-based DMC with the wrapper GAwAR and was designed for binary datasets. Given DMC's consideration of feature values alongside feature frequency, it outperformed other frequency-based rankers in certain scenarios. Moreover, the adaptive crossover and mutation rates in GA helped alleviate premature convergence. As a result, the final feature subset, with a manually set length of 10 in DMC-GAwAR, achieved commendable accuracy.

Modified Gray Wolf Optimization (MGWO) [20] ingeniously combined filters and wrappers for feature selection. In its methodology, MGWO first ranked features utilizing filters such as the ReliefF algorithm and Copula entropy. This approach aimed to diminish the search space for large-scale feature selection problems by leveraging correlation measures, thereby mitigating the inclusion of low-quality features in the initial population. Additionally, MGWO employed the differential evolution algorithm to broaden the search space of the standard Gray Wolf Optimization (GWO).

Pareto-based Ensemble of Feature Selection (PEFS) [12] introduced a novel ensemble approach incorporating four distinct filters: Fisher score, Local Learning-based Clustering, Correlation-based Feature Selection, and Maximum Information Coefficient. PEFS, akin to ATFS [11], embraced a similar ensemble strategy, with its core mechanism grounded in non-dominated sorting. Results highlighted PEFS's superior performance over alternative ensemble feature selection methods and basic algorithms, particularly in terms of accuracy and Fscore.

Ensemble Information Theory-based binary Butterfly Optimization Algorithm (EIT-bBOA) [21], was a hybrid method combining a filter, a wrapper, and an ensemble approach. EIT-bBOA employed Minimal Redundancy-Maximal New Classification Information (MR-MNCI) as a filter method to initially discard 80% of non-relevant features. Subsequently, the Information Gain binary Butterfly Optimization Algorithm (IG-bBOA) was utilized for wrapper optimal feature subset selection. Finally, EIT-bBOA manually selected the top 30 features using an ensemble of ReliefF and Fisher score. The results underscored the effectiveness of EIT-bBOA across six high-dimensional datasets sourced from the UCI repository.

The two-stage hybrid HyCluster [22] employed the filter Symmetrical Uncertainty (SU) index for initial feature ranking. The top-ranked features identified by SU underwent clustering, and subsequently, in the wrapper stage, an Incremental Wrapper Subset Selection search strategy (IWSS) operated on the best clusters to finalize the feature selection.

Table 2 lists and showcases the discussed hybrid or ensemble methods. However, existing feature selection methods for small-sample high-dimensional datasets may lack accuracy and efficiency. Similarly, there is not a single universally optimal global feature selection method applicable to all datasets. These serve as a driving force for researchers to continually innovate and develop novel hybrid or ensemble methods for feature selection. Table 2 also underscores the extensive use of evolutionary algorithms in feature selection, particularly when the feature selection problem is framed within a wrapper approach. This observation, along with the subsequent two reasons, motivated our adoption of GA in our research.

Table 2
Recent hybrid or ensemble feature selection methods for classification problems.

Method	Type	Application	Inclusion of frequency-based ranker	Year
WOA-MC [16]	Hybrid	Binary	Yes	2019
ATFS [11]	Ensemble	Binary	Yes	2021
EIT-bBOA [21]	Hybrid	Multiclass	No	2021
PEFS [12]	Ensemble	Multiclass	No	2021
$GA_{rank\&rand}$ [15]	Hybrid	Binary	Yes	2022
EMC-DWES [13]	Hybrid	Multiclass	Yes	2022
HyCluster [22]	Hybrid	Multiclass	No	2022
MGWO [20]	Hybrid	Multiclass	No	2023
MPR-MDES [14]	Hybrid	Multiclass	Yes	2024
DMC-GAwAR [17]	Hybrid	Binary	Yes	2024

- In the feature selection problem, the optimal set of features holds significance rather than the path leading to the result. Hence, evolutionary algorithms, which do not systematically track the path (i.e., they do not record the path), provide an appropriate solution for this problem.
- Evolutionary algorithms efficiently approximate the optimal set of features within a large search space using a minimal number of evaluated solutions.

3. Preliminaries

This section elucidates how existing frequency-based rankers assign scores to features differently, illustrated through an example. This highlights the existing gap in frequency-based rankers. Let us consider a binary classification task aimed at distinguishing between healthy and unhealthy individuals based on clinical records. Among these features, two significant features are Low-Density Lipoprotein (LDL) levels and blood pressure as shown in Fig. 1.

The first step is common to all existing frequency-based rankers, so that the values of each feature are sorted ascendingly, and the order of classes (including green for healthy and red for unhealthy as presented in Fig. 1) also changes accordingly in the response variable. Features can be categorized into two groups based on their impact on the separability of classes within the response variable. The first category includes features that exhibit a linear distribution, such as LDL. For example, in the case of LDL levels, typically levels greater than 100 mg/dL indicate membership in the unhealthy class, while levels below 100 mg/dL suggest membership in the healthy class, as illustrated in Fig. 1 (left side). Occasionally, misclassifications occur due to noise in the data, leading some unhealthy observations to be erroneously classified into the healthy class. Conversely, some observations are correctly classified as unhealthy even with LDL levels below 100 mg/dL, because they exhibit other symptoms indicative of an unhealthy condition. These

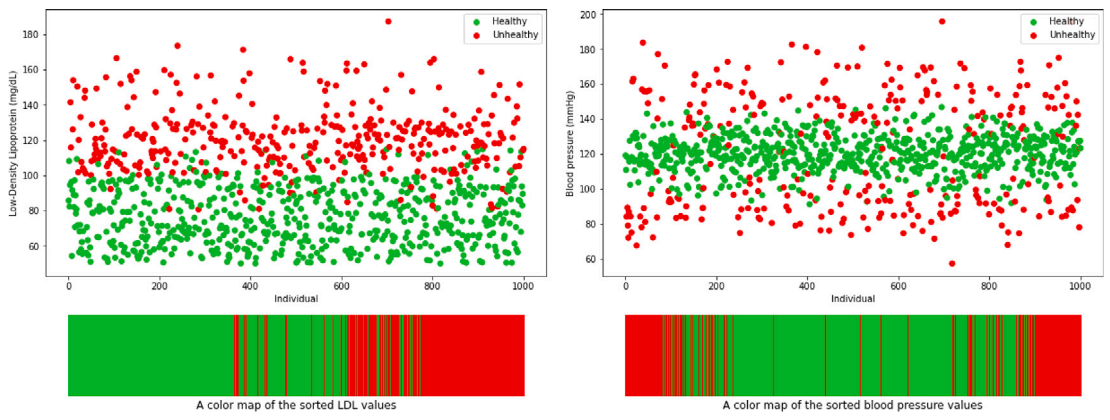

Fig. 1. Features with linear vs. non-linear patterns: LDL exhibits a linear pattern, while blood pressure shows a non-linear pattern.

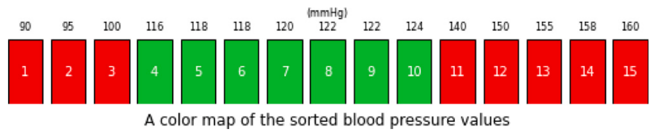

Fig. 2. An illustration demonstrating the perfect separability of blood pressure using 15 observations.

symptoms, such as positive diabetes, contribute to their final classification in the unhealthy category. Fig. 1 (left side) clearly shows that the overall characteristic of LDL distribution follows a linear pattern.

In contrast, blood pressure demonstrates a non-linear distribution of data, with the healthy class positioned between the unhealthy classes in Fig. 1 (right side). While it is true that some observations with seemingly elevated blood pressure levels are still categorized as healthy. This could be attributed to other features such as age, as higher blood pressure readings in older individuals are considered normal. The color maps presented in Fig. 1 distinctly illustrate linear and non-linear patterns.

The concept is that MC [16], SLI [11], SLI-γ [15], and DMC operate under the assumption of linear data distribution, as depicted in Table 1. Consequently, they may struggle to identify features such as blood pressure. Even EMC [13] and MPR [14], which do not rely on linear data distribution assumptions, may still struggle to effectively recognize such features. This limitation is clearly illustrated in Fig. 2, where an example with 15 observations and sorted values of blood pressure is depicted. In that figure, red colors denote the unhealthy class, while green colors represent the healthy class. The observations in both classes are almost balanced, with 8 classified as unhealthy and 7 as healthy. Notably, blood pressure in this dataset demonstrates a feature with perfect separation ability.

According to Eqs. (1)–(3), m_2^G and m_2^R represent the quantities of green and red labels within the mutual congestion region, while m_1^G and m_1^R denote the counts of green and red class labels outside of this region. The mutual congestion region spans from the initial green observation (observation number 4) to the final red observation (observation number 15). In this context, and according to Fig. 2, $m_1^R = 3$, $m_1^G = 0$, $m_2^R = 5$, and $m_2^G = 7$ for the corresponding calculations in Eqs. (1)–(3).

DMC in Eq. (4) [17] considers the feature values relative to their frequency in the response variable, rather than the frequency itself. x^{NMC} and y^{NMC} are the sets of feature values with consecutive red and green labels that are not in the mutual congestion region. Accordingly, x^{MC} and y^{MC} are the sets of feature values with red and green labels that are inside the mutual congestion region. Additionally, X_{max} and Y_{min} are the corresponding feature values of the last appearance of the red label and the first appearance of the green label, respectively. j and l are indices. Since DMC also adheres to the linear distribution of data, the right side of Eq. (4) is not calculable and remains empty according to Fig. 2 because y_l^{NMC} is not available and equals $\emptyset$. Therefore, DMC calculates the separability of the blood pressure feature in Fig. 2 solely with the left side of Eq. (4) as $\frac{|140-116|+|150-116|+|155-116|+|158-116|+|160-116|}{|90-116|+|95-116|+|100-116|} = \frac{183}{63} = 2.91$. SLI is a value in [−1,+1], whereby −1 and +1 denote the feature as completely non-separable and separable, respectively. The perfect case in MC, SLI, and DMC equals 0, indicating that the feature is perfectly separable (important).

$$MC = \frac{m_2^G + m_2^R}{m_1^G + m_2^G + m_2^R + m_1^R} \tag{1}$$

$$SLI = \frac{(m_1^G \times m_1^R) - (m_2^G \times m_2^R)}{\sqrt{(m_1^G + m_2^G)(m_1^G + m_2^R)(m_2^G + m_1^R)(m_1^R + m_2^R)}} \tag{2}$$

$$SLI\text{-}\gamma = \begin{cases} \frac{m_2^R}{m_1^R + m_2^R}, & for\ reds \\ \frac{m_2^G}{m_1^G + m_2^G}, & for\ greens \end{cases} \tag{3}$$

$$DMC = \frac{\sum \left|x_j^{MC} - Y_{min}\right|}{\sum \left|x_j^{NMC} - Y_{min}\right|} + \frac{\sum \left|y_l^{MC} - X_{max}\right|}{\sum \left|y_l^{NMC} - X_{max}\right|} \tag{4}$$

In contrast, EMC offers a more effective approach to handling non-linearity. Unlike MC, SLI, SLI-γ, and DMC, EMC does not default to assuming a linear distribution of data. Instead, it independently investigates the separability of each class within the response variable, as outlined in Eq. (5). Here, k represents the maximum number of classes (in our example in Fig. 2, k equals 2), m_{r_i} denotes the number of non-separable class labels (all labels in mutual congestion) for class r_i, and θ_{r_i} signifies the summation of both non-separable and separable labels for class r_i. The perfect case in EMC equals 0, indicating perfect separability for the feature. The EMC measure for Fig. 2 is calculated as $\frac{7+0}{7+8+0+7} = 0.32$. MPR calculates the maximum patterns, which are the maximum consecutive appearances of classes in the response variable, and then divides this by the total observations belonging to that class, as presented in Eq. (6). In Eq. (6), m_c represents the maximum number of consecutive appearances (pattern) for label c, n_c is the entire number of observations corresponding to label c, and k represents the number of classes. For example, in Fig. 2, the maximum pattern for the healthy class is 7, and for the unhealthy class, it is 5. Therefore, the MPR measure for Fig. 2 is calculated as $\sqrt{\frac{7}{7} \times \frac{5}{8}} = 0.79$. MPR assigns higher scores to superior features and notably, it better recognizes blood pressure as a separable feature in Fig. 2 compared to EMC. It is important to mention that Eqs. (1)–(6) illustrate the calculation of the corresponding frequency-based feature selection method for a specific

Table 3
Comparison of results from frequency-based rankers for blood pressure in Fig. 2. Lower scores indicate a higher degree of separability and, consequently, a more important feature. Notably, the favorable score, denoted by 0, is not attained by any of the methods for blood pressure in Fig. 2.

MC [16]	SLI [11]	SLI-γ [15]	EMC [13]	MPR [14]	DMC [17]
0.8	0.83	1 for greens 0.63 for reds	0.32	0.21	2.91

feature within a dataset. To compute these metrics for all features in the dataset, each feature must be calculated separately.

$$EMC = \frac{\sum_{i=1}^{k} m_{r_i}}{\sum_{i=1}^{k} \theta_{r_i}} \tag{5}$$

$$MPR = \sqrt{\prod_{c=1}^{k} \frac{m_c}{n_c}} \tag{6}$$

Table 3 summarizes and illustrates the scores calculated by each ranker to the blood pressure in Fig. 2. Lower scores in Table 3 are preferable, as they indicate a more important feature for the classification task. SLI yields a result of −0.66, as shown in Eq. (2). However, to align it with the range [0,1], we transformed the value from [−1,1] to [0,1], resulting in 0.17. Subsequently, we subtracted this value from 1 to ensure compatibility with the results of MC, SLI-γ, EMC, and MPR in Table 3. Thus, MC, SLI, SLI-γ, EMC, and MPR all fall within the range [0,1] in Table 3. However, for DMC, while the lower bound is 0, the upper bound is not clearly specifiable. Results in Table 3 reaffirm that MC, SLI, SLI-γ, and DMC were not able to correctly recognize the separability of the blood pressure feature in Fig. 2. In contrast, MPR and EMC assign better ranking scores, with MPR slightly outperforming EMC. However, blood pressure in Fig. 2 exhibits an important feature with perfect separability. Ideally, a perfect frequency-based ranker should be able to calculate a score of 0 for that feature in Table 3. Therefore, our primary objective in this research is to develop the Classification Error Impurity (CEI) algorithm, an effective frequency-based ranker capable of effectively detecting features exhibiting both linear and non-linear patterns.

4. Proposed method

Section 4 begins by introducing the CEI algorithm in Section 4.1. Next, Section 4.2 outlines the general procedure of AGAwER. Subsequently, Section 4.2.1 details the external repository and its role in generating diverse, qualified solutions and inserting them into the GA population. This section also explains how the radius is determined relative to the maximum distance among GA population solutions. In Section 4.2.2, the focus shifts to the GA component of AGAwER, describing the crossover and mutation operators utilized and the procedure for adjusting crossover and mutation rates.

4.1. Classification error impurity

A small-sample high-dimensional dataset D is characterized by a substantially greater number of features (dimensions) relative to the number of samples, expressed as $m \gg n$. The dataset D can be formally shown using Eq. (7) as a set of containing n data points.

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \in (\mathbb{R}^m \times \{c_1, c_2, \dots, c_k\})^n \tag{7}$$

This definition ensures that every sample (data point) in dataset D has a corresponding class label, fulfilling the multiclass classification scenario. Each data point (x_i) is paired with its unique class label (y_i), and all class labels originate from the predefined set of k classes. While the Cartesian product creates all possible pairings, dataset D only selects the relevant ones. It essentially picks the elements from $(\mathbb{R}^m \times \{c_1, c_2, \dots, c_k\})^n$ where the feature vector (x_i) and its corresponding class label (y_i) actually exist in D. This means D, as denoted in the Eq. (8), is a subset of the Cartesian product. It includes only the valid data point-label pairs from all possible combinations.

$$D \subseteq (\mathbb{R}^m \times \{c_1, c_2, \dots, c_k\})^n \tag{8}$$

For each feature of D, the Classification Error Impurity measure (*CEI* variable in Algorithm 1) should be calculated as follows. Like other frequency-based rankers, the feature needs to be sorted, resulting in the reordering of the response variable of D accordingly (Y in Algorithm 1). For each class label in the response variable, a corresponding subarray is created, starting from the first appearance of that class label and ending at the last appearance of that class label (line 4 in Algorithm 1). Algorithm 1 continues by calculating classification error inside the corresponding subarray relative to the class label through lines 5–8. The *CEI* measure for the feature corresponding to the reordered response variable, denoted as Y, is finally calculated as the minimum of the classification errors of the subarrays within the response variable. As a result, the *w_values* in Algorithm 1 for the response variable reordered based on the sorted blood pressure feature in Fig. 2 is $\left[\frac{7}{8+7}, \frac{0}{7+0}\right]$ for red and green class labels, respectively. Thus, *CEI* variable at line 10 equals 0, confirming that the blood pressure feature in Fig. 2 is a perfectly separable feature, which outperforms other frequency-based rankers in Table 3. Finding informative features with *CEI* is a minimization problem. Therefore, after calculating *CEI* for each feature, the list of calculated *CEI*s can be sorted accordingly to obtain a ranking of features.

Algorithm 1 Classification Error Impurity

```
Input: $Y$: the reordered response variable according to the sorted feature
Output: $CEI$
 1: $unique\_classes$ = unique($Y$)
 2: $w\_values$ = []
 3: for $current\_class$ in $unique\_classes$ do
 4:     Create a subarray according to the $current\_class$
 5:     $n\_other$ = Count the occurrences of other classes in the subarray
 6:     $n\_current$ = Count the occurrences of the current class in the subarray
 7:     $w = n\_other/(n\_current + n\_other)$
 8:     $w\_values$.append($w$)
 9: end for
10: $CEI$ = min($w\_values$)
11: Return $CEI$
```

4.2. Adaptive genetic algorithm with external repository

CMF-AGAwER is a hybrid feature selection method that combines the ensemble of top informative features obtained from Classification Error Impurity (CEI), Mutual Information (MI), and Fisher Ratio (FR) as indicated by the purple box, with the wrapper Adaptive Genetic Algorithm with External Repository (AGAwER) represented by green and yellow boxes in Fig. 3.

The ensemble phase concatenates the top features and then extracts the unique features from this concatenation to generate the feature space for AGAwER. We experimentally demonstrate that concatenating the top 50 features of CEI, MI alongside FR effectively generates a pool of sufficiently informative features for AGAwER. Green boxes in Fig. 3 represent tasks typically associated with the GA, while yellow boxes depict tasks related to the external repository. After generating the feature space at lines 1 and 2 and initializing GA parameters at line 3 of Fig. 3, AGAwER generates *nPop* random solutions, each of variable length at line 4. Each solution is a dataset that contains between 1 to 10 features from the feature space. Generally, as soon as a solution is generated, its corresponding fitness is calculated at line 5. The variable *GA_features* at line 6 represents all features that appear

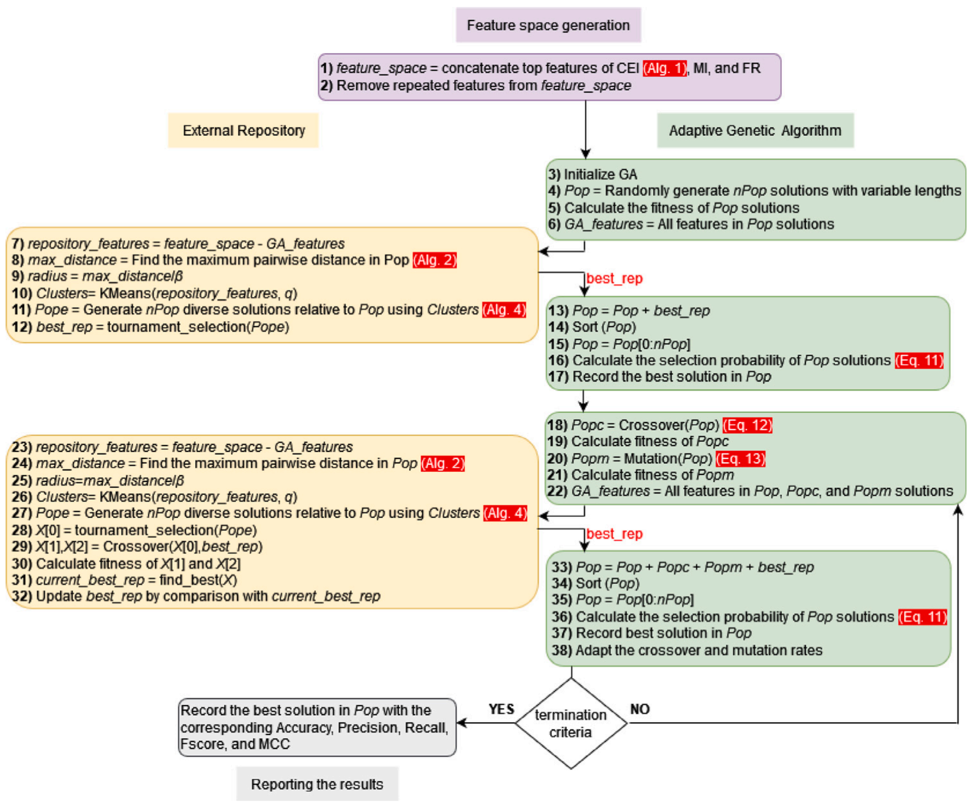
Feature space generation
1) feature_space = concatenate top features of CEI (Alg. 1), MI, and FR
2) Remove repeated features from feature_space
External Repository
Adaptive Genetic Algorithm
3) Initialize GA
4) Pop = Randomly generate nPop solutions with variable lengths
5) Calculate the fitness of Pop solutions
6) GA_features = All features in Pop solutions
7) repository_features = feature_space - GA_features
8) max_distance = Find the maximum pairwise distance in Pop (Alg. 2)
9) radius = max_distance/β
10) Clusters= KMeans(repository_features, q)
11) Pope = Generate nPop diverse solutions relative to Pop using Clusters (Alg. 4)
12) best_rep = tournament_selection(Pope)
best_rep
13) Pop = Pop + best_rep
14) Sort (Pop)
15) Pop = Pop[0:nPop]
16) Calculate the selection probability of Pop solutions (Eq. 11)
17) Record the best solution in Pop
18) Popc = Crossover(Pop) (Eq. 12)
19) Calculate fitness of Popc
20) Popm = Mutation(Pop) (Eq. 13)
21) Calculate fitness of Popm
22) GA_features = All features in Pop, Popc, and Popm solutions
23) repository_features = feature_space - GA_features
24) max_distance = Find the maximum pairwise distance in Pop (Alg. 2)
25) radius=max_distance/β
26) Clusters= KMeans(repository_features, q)
27) Pope = Generate nPop diverse solutions relative to Pop using Clusters (Alg. 4)
28) X[0] = tournament_selection(Pope)
29) X[1],X[2] = Crossover(X[0],best_rep)
30) Calculate fitness of X[1] and X[2]
31) current_best_rep = find_best(X)
32) Update best_rep by comparison with current_best_rep
best_rep
33) Pop = Pop + Popc + Popm + best_rep
34) Sort (Pop)
35) Pop = Pop[0:nPop]
36) Calculate the selection probability of Pop solutions (Eq. 11)
37) Record best solution in Pop
38) Adapt the crossover and mutation rates
termination criteria
YES
NO
Record the best solution in Pop with the corresponding Accuracy, Precision, Recall, Fscore, and MCC
Reporting the results

Fig. 3. The flow diagram of CMF-AGAwER illustrates green boxes representing the Genetic Algorithm (GA) and yellow boxes representing the external repository.

in the GA solutions. Similarly, the variable *repository_features* (line 7) represent the features that do not appear in *Pop* and hence will be used by the external repository. *max_distance* at line 8 represents the maximum distance (calculated using the Euclidean distance) among the solutions in *Pop*. Accordingly, *radius* is a threshold that signifies the degree of diversity, calculated at line 9 relative to *max_distance*. In our experiments, we found that a generally accepted value of $\beta = 2$ leads to good results within acceptable computational time. Therefore, we also adhere to $\beta = 2$ in this research. However, decreasing β (increasing *radius*) allows the external repository to generate more diverse solutions relative to *Pop*, which could potentially increase the computational time.

At lines 10–12, the external repository aims to identify a diverse solution (*best_rep*) to insert into the solutions in *Pop*. To achieve this, *repository_features* is partitioned into q clusters (we set $q = \sqrt{|repository\_features|}$). The aim of clustering *repository_features* by KMeans is avoiding multicollinearity and selecting similar features in *best_rep*. Then, *Pope*, which is a set of *nPop* random solutions with variable lengths, is generated at line 11 using *Clusters*, ensuring that the average distances of these solutions to the population (*Pop*) solutions are greater than a specified *radius*. The best solution is then selected and subsequently recorded in *best_rep* at line 12 via tournament selection and added to *Pop* in the GA at line 13.

Lines 13–15 of GA include merging, sorting, and truncating *Pop* solutions. If *best_rep* is deemed a good solution, it would be retained in *Pop* after truncating; otherwise, it would be discarded. Line 16 calculates the selection probabilities of *Pop* solutions, which will later be utilized to select the corresponding parents during crossover at line 18. Lines 1–17 are implemented once and the best solution in *Pop* is recorded.

The main loop of AGAwER begins at line 18 and continues through line 38. Lines 18–21 are dedicated to creating *Popc* and *Popm*, as well as calculating the fitness of the offspring and mutants. The parents for crossover are selected using Roulette Wheel Selection (RWS) based on the selection probabilities of solutions within the *Pop*. Before merging, sorting, and truncating the *Pop*, the external repository is responsible for identifying the *best_rep* to add it to *Pop* at Line 33. To accomplish this, the external repository generates random diverse solutions relative to *Pop* using lines 23–27. At line 28, the best solution in *Pope* is selected based on tournament selection and recorded in $X[0]$. Subsequently, at line 29, $X[0]$ is crossed over with the existing *best_rep*. Essentially, the external repository utilizes its memory (existing *best_rep*) to create two new offspring by crossing over its memory with a recently found diverse solution, $X[0]$. Finally, at line 31, the external repository selects the best solutions in X (*current_best_rep*) based on their fitness (prediction accuracy) and updates the *best_rep* at line 32, provided that the *current_best_rep* in the current iteration exceeds the best ever *best_rep*. The GA controls the process of AGAwER at line 33. It merges *Pop* with *Popc*, *Popm*, and *best_rep* (line 33), sorts *Pop* (line 34), and truncates *Pop* (line 35). If *best_rep* is a valuable solution, it will be retained in *Pop* after truncating at line 35; otherwise, it will be discarded. The selection probabilities, the selection of the best solution in *Pop*, and the adaptation of crossover and mutation rates are implemented at lines 36–38. The proposed GA checks the possibility of adapting crossover and mutation rates so that it gradually increases the mutation rate (P_m) and decreases the crossover rate (P_c) to improve exploration at line 38. The termination criteria involve reaching the maximum iteration (100) or not observing improvement in the fitness function, which is controlled by adapting crossover and mutation rates, or reaching the maximum fitness (a solution with a fitness of 1 is found).

The general idea is that, by utilizing an external repository and adopting a strategy to adapt the crossover and mutation rates, AGAwER will be able to explore the search space more effectively, generally leading to escaping premature convergence. The remainder of Section 4.2 is dedicated to detailing how external repository and GA operate in 4.2.1 and 4.2.2, respectively.

4.2.1. External repository

After calculating *repository_features* and in order to compute the *radius*, the external repository should calculate the maximum distance (*max_distance*) among pairwise distances of solutions in *Pop*. To do so, the *distance_matrix* should be computed first in Algorithm 2 which uses Algorithm 3 to calculate average minimum pairwise distance. Algorithm 2 takes the population (*Pop*) as input and computes the maximum pairwise distance between solutions, returning this value as *max_distance*. Due to symmetry, half of the *distance_matrix* is calculated at line 5 using average_minimum_distance function, and the other half is symmetrically filled at line 6. To calculate the average minimum distance between two solutions, we first need to identify the most similar features in the two solutions. This can be achieved by computing the Euclidean distance between each pair of features in the two solutions as shown in Algorithm 3.

Algorithm 2 Maximum pairwise distance

```
Input: Pop
Output: max_distance
1:  L = len(Pop)
2:  Initialize a zero distance_matrix with size (L, L)
3:  for i in L do
4:      for j in (i+1,L) do
5:          distance_matrix[i,j] = average_minimum_pairwise_distance (Pop[i],Pop[j])
6:          distance_matrix[j,i] = distance_matrix[i,j]
7:      end for
8:  end for
9:  max_distance = max(distance_matrix)
10: Return max_distance
```

Algorithm 3 Average minimum pairwise distance

```
Input: S1,S2
Output: average_distance
1: distances_from_S1 = Calculate the minimum distance from each feature in S1 to the features of S2
2: distances_from_S2 = Calculate the minimum distance from each feature in S2 to the features of S1
3: average_distance = mean([mean(distances_from_S1), mean(distances_from_S2)]
4: Return average_distance
```

Suppose we have two solutions, $S1$ and $S2$. Solution $S1$ consists of two features: [1, 2] and [3, 4], and solution $S2$ consists of three features: [2, 1], [5, 6], and [7, 8]. Using Euclidean distance, we can compute the distance between each feature in $S1$ and $S2$.

For feature $[1, 2]$ in $S1$:

- Distance to $[2, 1]$ in $S2$: $\sqrt{(1-2)^2+(2-1)^2} = \sqrt{2} \approx 1.41$
- Distance to $[5, 6]$ in $S2$: $\sqrt{(1-5)^2+(2-6)^2} = \sqrt{32} \approx 5.66$
- Distance to $[7, 8]$ in $S2$: $\sqrt{(1-7)^2+(2-8)^2} = \sqrt{72} \approx 8.49$

For feature $[3, 4]$ in $S1$:

- Distance to $[2, 1]$ in $S2$: $\sqrt{(3-2)^2+(4-1)^2} = \sqrt{10} \approx 3.16$
- Distance to $[5, 6]$ in $S2$: $\sqrt{(3-5)^2+(4-6)^2} = \sqrt{8} \approx 2.83$
- Distance to $[7, 8]$ in $S2$: $\sqrt{(3-7)^2+(4-8)^2} = \sqrt{32} \approx 5.66$

Similarly, we compute the distances from features in $S2$ to $S1$:
For feature $[2, 1]$ in $S2$:

- Distance to $[1, 2]$ in $S1$: $\sqrt{(2-1)^2+(1-2)^2} = \sqrt{2} \approx 1.41$
- Distance to $[3, 4]$ in $S1$: $\sqrt{(2-3)^2+(1-4)^2} = \sqrt{10} \approx 3.16$

For feature $[5, 6]$ in $S2$:

- Distance to $[1, 2]$ in $S1$: $\sqrt{(5-1)^2+(6-2)^2} = \sqrt{32} \approx 5.66$
- Distance to $[3, 4]$ in $S1$: $\sqrt{(5-3)^2+(6-4)^2} = \sqrt{8} \approx 2.83$

For feature $[7, 8]$ in $S2$:

- Distance to $[1, 2]$ in $S1$: $\sqrt{(7-1)^2+(8-2)^2} = \sqrt{72} \approx 8.49$
- Distance to $[3, 4]$ in $S1$: $\sqrt{(7-3)^2+(8-4)^2} = \sqrt{32} \approx 5.66$

Now, we take the average of the minimum distances from each set: For $S1$ to $S2$: $\frac{1.41+2.83}{2} \approx 2.12$. For $S2$ to $S1$: $\frac{1.41+2.83+5.66}{3} \approx 3.30$. Therefore, the average_distance in line 3 of Algorithm 3 equals $\frac{2.12+3.30}{2} = 2.71$. Accordingly, the *max_distance* is the maximum value in the *distance_matrix*.

The *radius* is calculated relative to the *max_distance* so that $\frac{1}{\beta}$ of *max_distance* is assigned to the radius. Larger values of β result in smaller radii, yielding less diverse solutions, and vice versa. We experimentally set $\beta = 2$ in this research.

The external repository also partitions the *repository_features* into q clusters (we set $q = \sqrt{|repository\_features|}$) by KMeans and generates *nPop* diverse solutions (*Pope*) relative to *Pop* solutions using *radius* by Algorithm 4. A solution is considered diverse if its average distance from the *Pop* solutions is greater than the specified *radius*. Algorithm 4 randomly selects some clusters from the set *Clusters* at line 5. Subsequently, it randomly chooses a feature from the *selected_clusters* to compose solution X in line 6. Following this, at lines 7–15, the algorithm assesses whether X qualifies as a diverse solution. If X meets the diversity criteria, it is added to the *Pope*. The algorithm ensures that the size of *Pope* solutions grows until it reaches *nPop*.

Algorithm 4 Generation of diverse solutions

```
Input: Pop, nPop, radius, Clusters
Output: Pope
1:  distance = []
2:  Pope = []
3:  size = 0
4:  while size < nPop do
5:      selected_clusters = randomly select some clusters from the set Clusters
6:      X = randomly select a feature from each cluster in selected_clusters
7:      for i in nPop do
8:          distance.append(average_minimum_pairwise_distance(X, Pop[i]))
9:      end for
10:     avg_distance = mean(distance)
11:     if avg_distance > radius then
12:         Pope.append(X)
13:         size = size +1
14:     end if
15: end while
16: Return Pope
```

Fig. 4 illustrates how diverse solutions are recognized and distinguished by non-diverse solutions by external repository. D denotes the maximum distance and can be derived from the *distance_matrix* in Algorithm 2. Similarly, r represents the intended radius for diversity. Accordingly, d_1 and d_2 are the average distances of two diverse solutions relative to solutions in *Pop*, computed using *avg_distance* in Algorithm 4, where both $d_1 > r$ and $d_2 > r$. In contrast, d_3 is an example of average distance of a non-diverse solution ($d_3 < r$).

The current best diverse solution (*best_rep* in iteration 0) is the one in *Pope* with the greatest fitness (prediction accuracy) that can be identified with applying tournament selection on *Pope* solutions. However in the main loop of AGAwER the current best diverse solution ($X[0]$ at line 28) is also crossed over with the *best_rep* ever found to

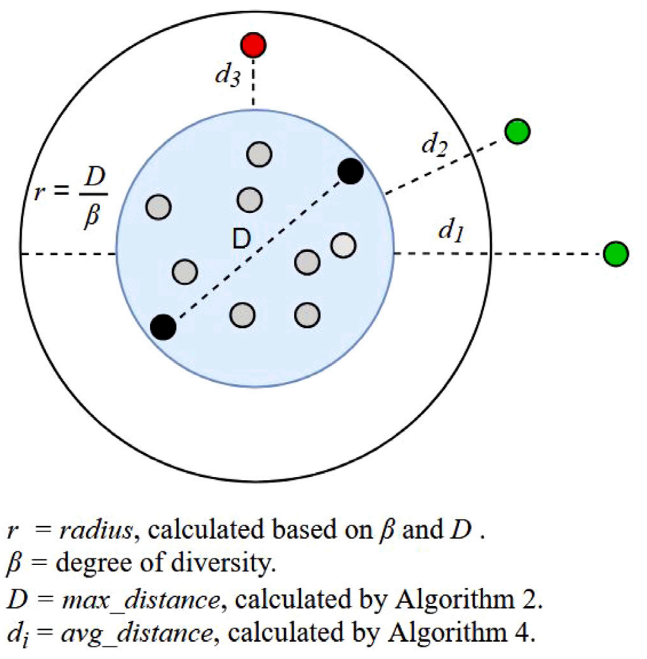
d_3
d_2
d_1
$r = \frac{D}{\beta}$
D
r = radius, calculated based on β and D .
β = degree of diversity.
D = max_distance, calculated by Algorithm 2.
d_i = avg_distance, calculated by Algorithm 4.

Fig. 4. Illustration of two diverse solutions (green circles) and a non-diverse solution (red circle) relative to *r*. The black circles represent two solutions in *Pop* with the maximum distance (*D*), while the gray circles depict other existing solutions in *Pop*.

generate 2 offspring ($X[1]$ and $X[2]$ at line 29). The solution in X with the greatest fitness is the *current_best_rep* for that iteration at line 31. Subsequently, the *best_rep* is updated relative to the *current_best_rep* at line 32.

4.2.2. Genetic algorithm

The GA section of AGAwER includes single-point crossover, which could be used for variable-length solutions. The two parents, S_1 and S_2, are selected for crossover using Roulette Wheel Selection (RWS) as described in Eq. (9). This selection is based on the cumulative probabilities W_i in Eq. (10) and the selection probabilities S_{P_i} in Eq. (11) of the solutions in *Pop*, where k in Eq. (9) denotes the index of the parent solution in *Pop*, and U is a random number in the interval [0, 1]. Additionally, F_j in Eq. (11) is the fitness of jth solution in *Pop*. This ensures that solutions with higher fitness have a higher chance of being selected, as they will have a larger portion of the cumulative probability.

$$k = min\{i : U \leq W_i\} \tag{9}$$

$$W_i = \sum_{j=1}^{i} S_{P_j} \tag{10}$$

$$S_{P_i} = \frac{F_i}{\sum_{j=1}^{nPop} F_j} \tag{11}$$

If both parents S_1 and S_2 have only one feature (i.e., their size is one), then both parents are directly copied to the next generation. If one parent has only one feature and the other parent has more than one feature, a single crossover point is randomly selected on the parent with multiple features. This crossover point is used to generate two offspring from the two parents. Assuming that S_1 and S_2 have been selected using RWS for crossover, and both parents have a size greater than 1, the two offspring, denoted as O_1 and O_2, are generated using the crossover process defined by Eq. (12) in which C_1 and C_2 are crossover points for S_1 and S_2 respectively. GA ensures generating $n_c = 2 \times \lceil \frac{P_c \times \text{nPop}}{2} \rceil$ offspring in *Popc* population. The crossover probability P_c adapts dynamically, leading to corresponding adjustments in the number of offspring n_c throughout the iterations of the GA.

$$\begin{cases} O_1 & = S_1[: C_1] + S_2[C_2 :] \\ O_2 & = S_2[: C_2] + S_1[C_1 :] \end{cases} \tag{12}$$

The mutation operator does not utilize selection probabilities; instead, it randomly selects n_m solutions from the population (*Pop*) to generate the mutation population (*Popm*), where n_m is calculated as $\lceil P_m \times n\text{Pop} \rceil$ and P_m adjusts adaptively. The random replacement mutation operator randomly replaces a feature in the selected solution (S) with another feature from the feature space, ensuring that the replacement feature is not already present in the selected solution. Assuming k represents the index of the feature in S, and $S'[k]$ denotes the kth feature in the mutated solution S'. This equation specifies that the feature at index j in the original solution S is replaced by a random feature ϕ, while all other features remain unchanged.

$$S'[k] = \begin{cases} \phi, & \text{if } k = j \\ S[k], & \text{otherwise} \end{cases} \tag{13}$$

The proposed genetic algorithm (GA) continually monitors fitness improvement during its iterations. If there is no improvement in fitness over several consecutive iterations, the algorithm dynamically adjusts the probabilities of crossover (P_c) and mutation (P_m). The GA terminates under specific conditions: after 20 iterations without fitness improvement, after reaching the maximum iteration limit of 100, or if the fitness (prediction accuracy) reaches 1 in any iteration.

Initially, P_c is initialized to 0.9 and P_m to 0.4. Whenever there is a fitness improvement in an iteration, the probabilities revert to their original values ($P_c = 0.9$ and $P_m = 0.4$). However, if there is no improvement after 5 consecutive iterations, P_c is decreased by 0.3, and P_m is increased by 0.2. Consequently, during the final 20 iterations of the GA, the crossover probabilities (P_c) are adjusted as 0.9, 0.6, 0.3, and 0, and the mutation probabilities (P_m) evolve as 0.4, 0.6, 0.8, and 1. This adaptation ensures that in the last 5 iterations, the GA primarily relies on its mutation operator, making the crossover operator ineffective.

Generally, adjusting the probability of crossover (P_c) and the probability of mutation (P_m) may impact the number of solutions in *Popc* and *Popm* from one iteration to the next, which is part of the adaptive process of GA. This variability can actually be beneficial, as it introduces different levels of exploration and exploitation in different stages of the algorithm's run. The proposed GA also maintains a consistent population size of *nPop* through its truncation strategy, as detailed in lines 15 and 35 of Fig. 3.

Table 4
Description of benchmark datasets under study.

Dataset	Sample size	Feature size	Number of classes	Sample distribution
Colon	62	2,000	2	22-40
CNS	60	7,129	2	21-39
GLI	85	22,283	2	26-59
SMK	187	19,993	2	90-97
Leukemia-Binary	72	7,129	2	47-25
Leukemia-Multiclass	72	7,129	3	25-38-9
Covid-19	234	15,979	3	100-41-93
MLL	72	12,582	3	24-20-28
SRBCT	83	2,308	4	29-11-18-25

5. Results

This section introduces the datasets used in the study, the measurement criteria applied, and evaluates the performance of CMF-AGAwER alongside efficiency analysis of CEI and AGAwER. The research is carried out using Python 3.9.13 on a computer equipped with an Intel(R) Core(TM) i5-4200U CPU running at 1.60 GHz, 12 GB of RAM, a 720 GB HDD, and 64-bit Windows 10 Pro operating system.

Table 5
Overall accuracy before and after application of CMF-AGAwER, along with the average subset length of CMF-AGAwER, on benchmark datasets using a decision tree classifier.

Dataset	Accuracy (before CMF-AGAwER)	Accuracy (after CMF-AGAwER)	Subset length (after CMF-AGAwER)
Colon	0.69	0.94	6
CNS	0.68	0.93	9
GLI	0.81	0.96	9
SMK	0.56	0.77	8
Leukemia-Binary	0.89	0.99	9
Leukemia-Multiclass	0.85	0.97	5
Covid-19	0.6	0.77	8
MLL	0.79	1	8
SRBCT	0.83	0.96	7

5.1. Datasets

The study encompasses 9 benchmark small-sample high-dimensional microarray datasets, as detailed in Table 4. These datasets exhibit a wide range of features, varying from 2000 in Colon to 22,283 in GLI, yet they consist of relatively few samples. While the SMK and MLL datasets demonstrate near-balanced data distributions, the remaining datasets exhibit varying degrees of imbalance.

5.2. Measurement criteria

The overall accuracy is a commonly acknowledged measure used to assess classification models, reflecting the percentage of accurate predictions generated by the model. It is expected that an effective feature selection technique will improve overall accuracy by pinpointing the most relevant features. Mathematically, the overall accuracy is defined in Eq. (14), where TS denotes the test set with a size of $|TS|$. Moreover, $h(x_i)$ and Y_i represent the classifier's prediction and the true class label for the ith element of the test set, respectively. The numerator of Eq. (14) acts as a binary variable, assuming a value of 1 if $(Y_i = h(x_i))$, and 0 otherwise.

Key metrics, such as precision (Eq. (15)), recall (Eq. (16)), and Fscore (Eq. (17)) are important metrics in addition to overall accuracy, especially in scenarios with imbalanced datasets such as those found in medical datasets. Recall represents a model's capability to identify all relevant cases within a dataset. Conversely, precision quantifies the ratio of correct positive identifications among the total positive identifications made. Furthermore, the Fscore is calculated as the harmonic mean of precision and recall.

Eqs. (15)–(17) define precision, recall, and Fscore for binary classification. In this context, False Negatives (FN) refer to positive samples that were incorrectly classified as negative, False Positives (FP) denote negative samples misclassified as positive, and True Positives (TP) signify correct positive predictions. For multiclass classification, the performance metrics (precision, recall, and Fscore) for each class label are independently computed using a one-vs-rest strategy, followed by averaging across all classes.

$$\text{Overall accuracy} = \frac{\sum x_i \in TS \;\; \mathbf{1}(Y_i = h(x_i))}{|TS|} \tag{14}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{15}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{16}$$

$$\text{Fscore} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{17}$$

The Matthews Correlation Coefficient (MCC) is a measure of the quality of classifications, taking into account true and false positives and negatives. It is particularly useful when dealing with imbalanced datasets. The MCC value ranges between −1 and +1, where: +1 indicates a perfect prediction, 0 indicates a random prediction, and −1 indicates total disagreement between prediction and observation. The MCC for binary classification is calculated using Eq. (18). MCC can be extended for multiclass tasks by considering each class against the rest and then averaging the individual MCC values across all classes.

$$\text{MCC} = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \tag{18}$$

The other criterion is the subset length which is the length of the automatically selected features by CMF-AGAwER.

Finally, in this paper we also reflect Number of Function Evaluations (NFE) which refers to the total count of fitness function evaluations performed during the optimization process in AGAwER. Tracking NFE can reflect the efficiency and convergence behavior of the optimization algorithm, as it measures the actual work (fitness evaluations) done by the algorithm. The Overall NFE of AGAwER (Overall NFE in Eq. (19)) is the summation of the NFE of the GA (NFE_{GA} in Eq. (20)) and the NFE of the external repository (NFE_{ER} in Eq. (21)). In our proposed external repository approach, during each iteration, the best solution from the current population (Pope) undergoes crossover with the best solution ever found in the external repository. Consequently, in each iteration, a total of ($Pope$ + 2) solutions are evaluated.

$$\text{Overall NFE} = \text{NFE}_{\text{GA}} + \text{NFE}_{\text{ER}} \tag{19}$$

$$\text{NFE}_{\text{GA}} = Pop + [(Popc + Popm) \times \text{number of iterations}] \tag{20}$$

$$\text{NFE}_{\text{ER}} = Pope + [(Pope + 2) \times \text{number of iterations}] \tag{21}$$

5.3. Classifiers

The classifier that is used for the analysis of CMF-AGAwER in Tables 5, 6, 8, 9, and Figs. 5–8 is a decision tree with the *random_state* parameter fixed at 42. However, to demonstrate that CMF-AGAwER is independent of the classifier, we conducted an experiment in Fig. 9 using Support Vector Machine (SVM) and Naive Bayes (NB) classifiers. The SVM in Fig. 9 uses an RBF kernel, while the NB classifier is a Gaussian NB. Both have the *random_state* parameter set to 42.

Table 6
Average prediction accuracy based on top N features ($N = 10, 20, 30, 40, 50$). The superior values are highlighted in bold. The classifier used is a decision tree.

Dataset	Non-frequency-based rankers						Frequency-based rankers						
	Mutual Information	Fisher Ratio	Correlation Coefficient	Random Forest	ReliefF	SHAP	MC	SLI	SLI-γ	EMC	MPR	DMC	CEI
Colon	0.81	0.78	0.78	0.79	0.78	0.78	0.79	0.79	0.75	0.80	0.79	0.81	**0.82**
CNS	**0.76**	0.70	0.70	0.66	0.64	0.64	0.71	0.74	0.62	0.74	0.69	0.62	0.70
GLI	**0.86**	0.83	0.83	0.85	0.83	**0.86**	0.85	**0.86**	0.77	0.85	0.85	0.84	0.82
SMK	**0.67**	**0.67**	0.66	0.66	0.63	0.64	0.65	0.65	0.64	0.65	0.66	0.62	0.65
Leukemia-Binary	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**	**0.90**
Leukemia-Multiclass	0.86	0.87	0.86	**0.88**	0.85	**0.88**	NA	NA	NA	0.85	0.85	NA	**0.88**
Covid-19	0.69	0.68	0.66	0.70	**0.71**	0.68	NA	NA	NA	0.65	0.65	NA	**0.71**
MLL	0.90	**0.94**	0.91	0.91	0.93	0.88	NA	NA	NA	0.90	0.91	NA	0.91
SRBCT	0.84	**0.87**	0.82	0.83	0.85	0.85	NA	NA	NA	0.85	0.85	NA	**0.87**

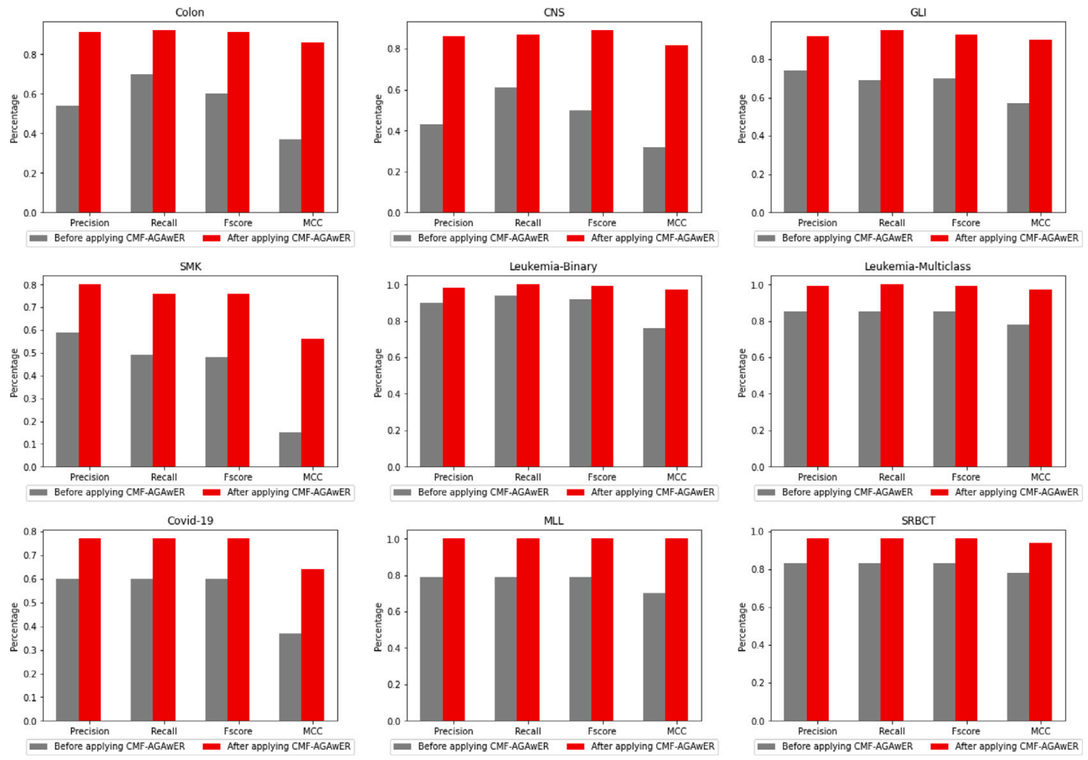

Fig. 5. Precision, recall, Fscore, and MCC achieved by CMF-AGAwER, on benchmark datasets using a decision tree classifier. The red bars indicate improvements after applying CMF-AGAwER, while the gray bars represent results without CMF-AGAwER.

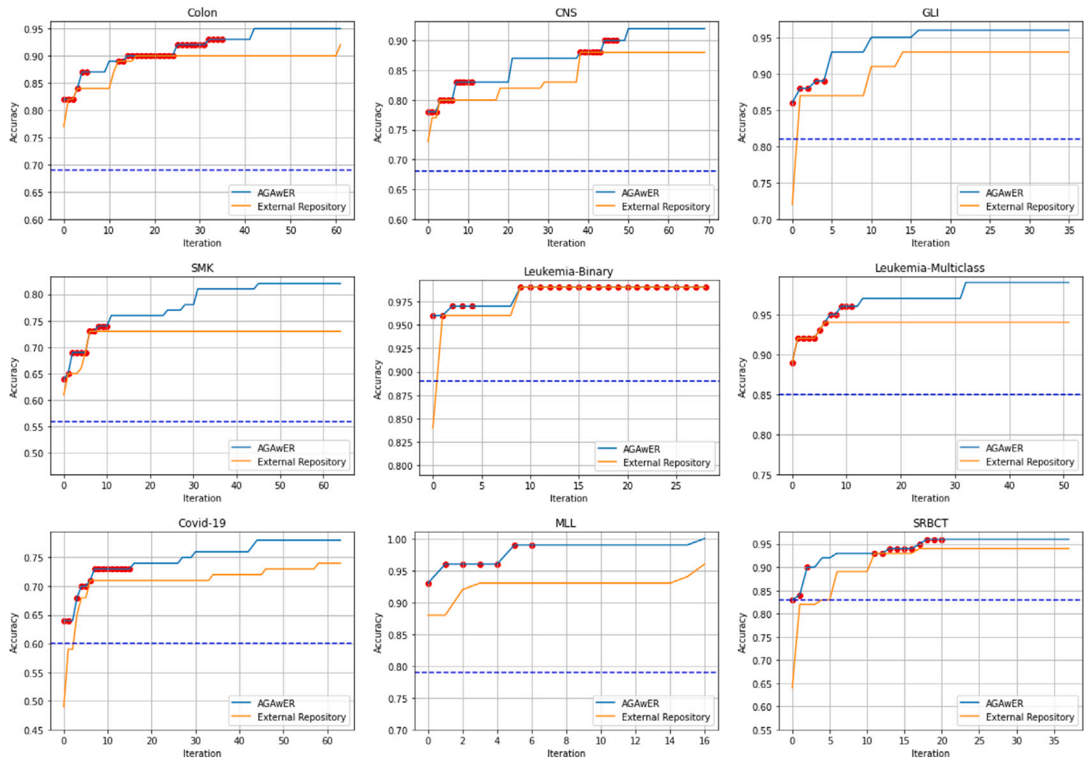

Fig. 6. The impact of the external repository on GA. The red circles represent the best solutions computed by the external repository. The horizontal dotted line indicates the prediction accuracy before applying CMF-AGAwER. The classifier used is a decision tree.

5.4. Performance analysis

The results presented in Table 5 and Fig. 5 are computed based on the average of 10 runs and employing stratified 5-fold cross-validation.

Table 5 displays the average overall accuracy and average subset length achieved after applying CMF-AGAwER, compared to not using CMF-AGAwER. The table confirms a significant improvement in overall accuracy, with subset lengths consistently low across all datasets,

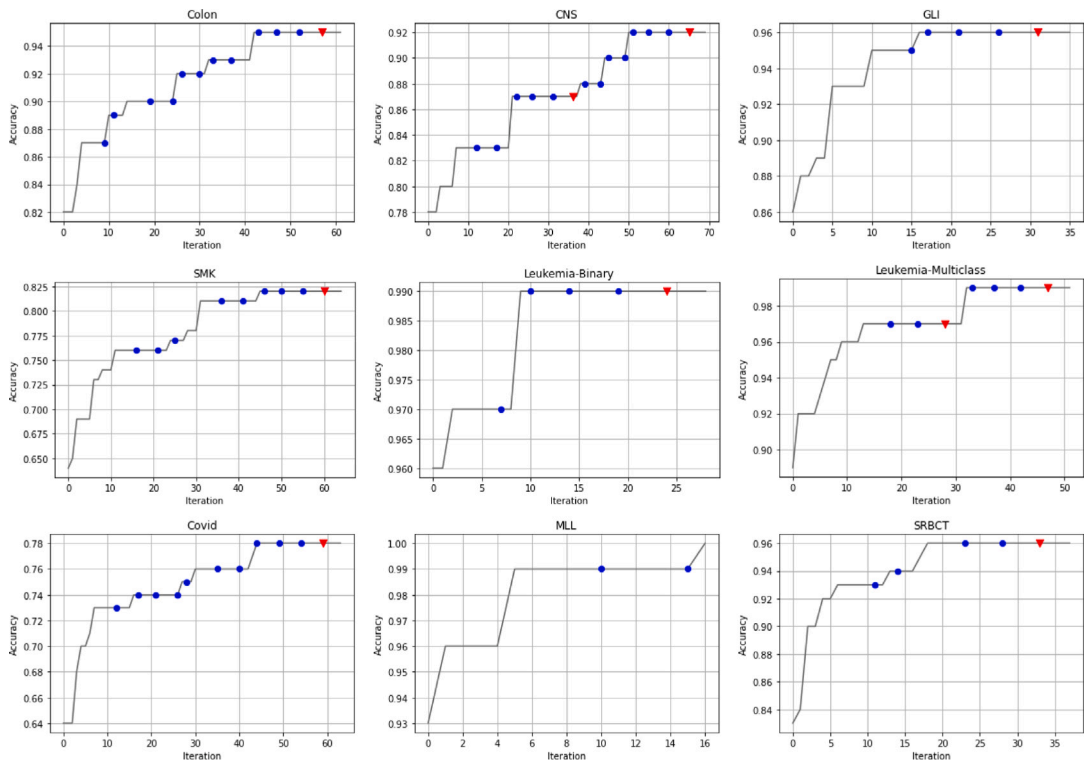

Fig. 7. The impact of adaptive crossover and mutation rates on GA. The red triangle indicates a setting of rates with $P_c = 0$ and $P_m = 1$. The classifier used is a decision tree.

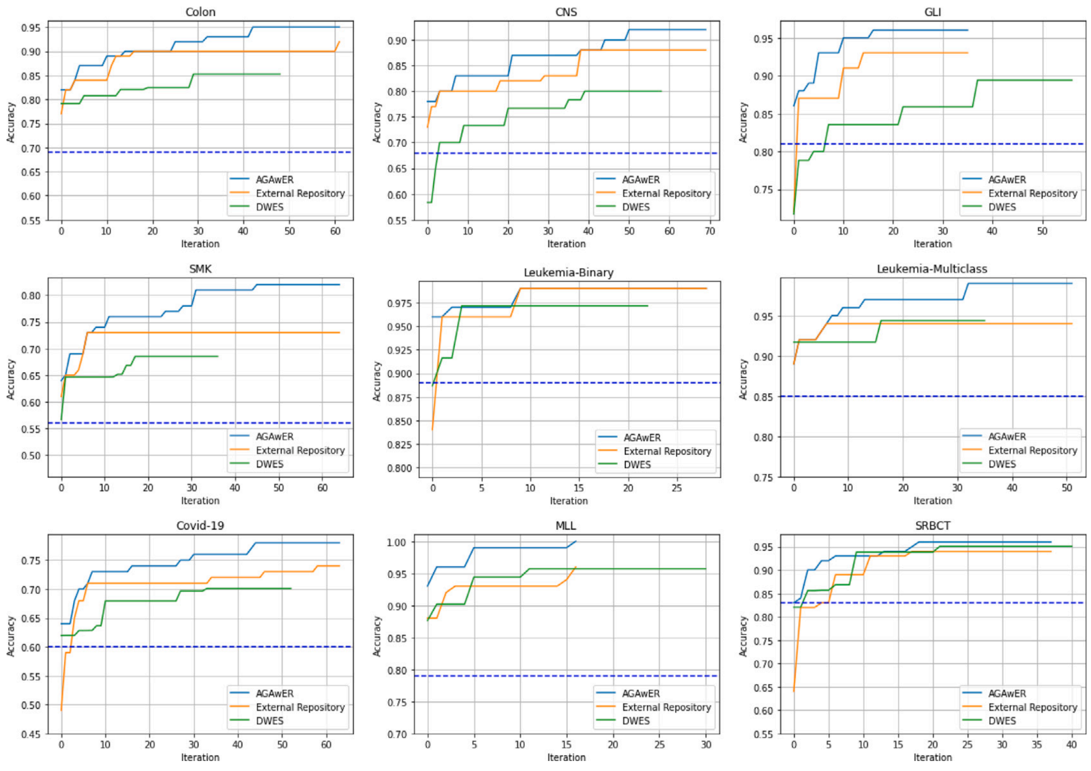

Fig. 8. Comparison of convergence graphs between AGAwER and its external repository with DWES [13]. The horizontal dotted line indicates the prediction accuracy before applying CMF-AGAwER. The classifier used is a decision tree. The classifier used is a decision tree.

ranging from 5 to 9. Additionally, Table 5 shows that the maximum accuracy increase occurred in Colon and CNS, with an increase of 0.25. Following this, SMK and MLL rank second with a 0.21 increase. Covid-19, GLI, Leukemia-Multiclass, and Leukemia-Binary are next in the ranking, showing increases of 0.17, 0.15, 0.12, and 0.1 in accuracy, respectively.

Fig. 5 illustrates how the proposed CMF-AGAwER enhances the precision, recall, Fscore, and Matthews Correlation Coefficient (MCC)

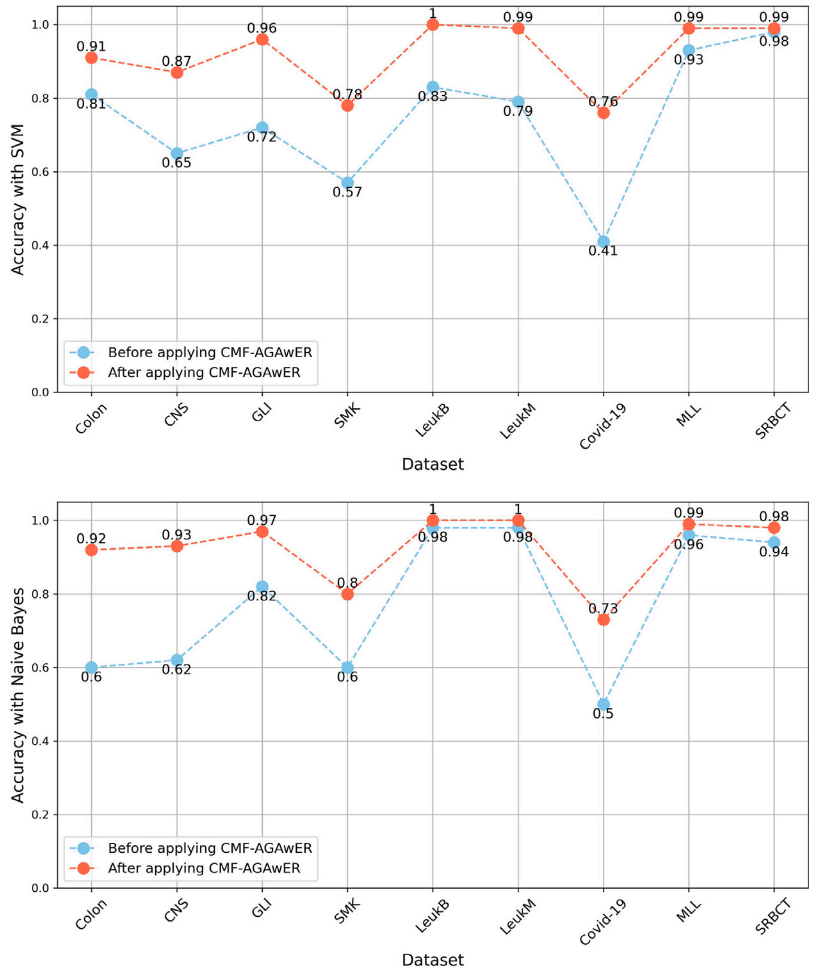

Fig. 9. Assessment of the independence of CMF-AGAwER from the classifier. The dataset LeukB refers to Leukemia-Binary, while LeukM refers to Leukemia-Multiclass.

of the datasets. This improvement is notably substantial; for example, the MCC measure in Colon, CNS, and SMK has significantly increased from 0.37, 0.32, and 0.15 to 0.86, 0.82, and 0.56, respectively. In another example, all performance measures in MLL have achieved a perfect score of 1 when CMF-AGAwER is applied, compared to not using CMF-AGAwER.

Fig. 6 illustrates how the external repository enriches the GA with diverse, high-quality solutions, thereby enhancing its convergence. The red circles on AGAwER indicate whether the inserted *best_rep* computed by the external repository (Fig. 3) remains in the top *nPop* population of GA after merging, sorting, and truncating the solutions. Clearly, if the *best_rep* still appears in the GA population, it signifies that the diverse solution introduced by the external repository is the informative solution. For example, in the case of SRBCT, the *best_rep* introduced by the external repository remains in the GA population during iterations 0–2 and 11–20. Even if the *best_rep* is not observed in other iterations, there is a possibility that the top solutions of GA may inherit variables from the *best_rep* through the crossover and mutation operators. In other words, the influence of the *best_rep* could be implicitly present within the top solutions of GA. Experiments in Fig. 6 demonstrate that the external repository can enhance the performance of the GA, particularly in the initial iterations. Furthermore, integrating GA with the external repository enhances the convergence of the GA, compared to using the external repository alone.

Fig. 7 demonstrates how adaptive adjustment of the probability of crossover (P_c) and probability of mutation (P_m) improves convergence. We utilize the same convergence graph of AGAwER as shown in Fig. 6. The blue circles indicate instances where P_c and P_m are adjusted (either changed or reverted to their initial values). The red triangle represents the last attempt of AGAwER to adjust the rates by generating all solutions through mutation and deactivating the crossover operator (assigning $P_c = 0$ and $P_m = 1$). For example, in the case of CNS, the first adjustment of P_m to 1 and consequently setting P_c to 0 (illustrated by the first red triangle on the convergence graph) increased the exploration factor, resulting in an improvement in fitness (prediction accuracy). This fitness improvement prompted the reversal of P_c and P_m to their initial values (0.9 and 0.4) at iteration 39. The same thing happened with Leukemia-Multiclass, where the first adjustment of P_m to 1 and P_c to 0 improved the fitness. As a result, AGAwER reverted the P_m and P_c values to their initial values afterward. In the case of MLL, from iteration 5–10, no improvement was observed in fitness. Therefore, for the first time at iteration 10, the rates were adjusted ($P_c = 0.6$, $P_m = 0.6$). However, this adjustment did not lead to further improvement in fitness. Consequently, at iteration 15, the rates were adjusted again ($P_c = 0.3$, $P_m = 0.8$). This adjustment resulted in an increase in fitness to a perfect score of 1. Since the stopping criteria of AGAwER include early stopping upon achieving a perfect accuracy of 1, AGAwER stopped at iteration 16. The other datasets depicted in Fig. 7 can be interpreted analogously.

Fig. 8 shows a comparison of convergence graphs between AGAwER, an external repository, and the Discrete Weighted Evolution Strategy (DWES) [13] described in Algorithm 5. The original DWES involved considering the linkage type and number of clusters as hyperparameters. However, to ensure a fair comparison, given that CMF-AGAwER achieved the maximum subset length of 9 (as detailed at Table 5), we set the number of clusters to 10 in line 1 of Algorithm

5. Experimental results demonstrated that DWES achieved superior outcomes with complete linkage. Additionally, KMeans clustering bears a resemblance to the complete linkage hierarchical clustering method employed in the external repository. Therefore, we opted for complete linkage to partition the *feature_space* into 10 clusters at line 1.

In summary, Algorithm 5 utilizes a (1+1) evolution strategy for discrete problems. It starts by acquiring the *feature_space*, formed by concatenating the top 50 features of CEI, MI, and FR. DWES operates on this *feature_space* and partitions it into 10 clusters ($C : c_1, c_2, \ldots, c_{10}$) using hierarchical complete linkage at line 1, and the corresponding weights are initialized ($W : w_1 = 0.5, w_2 = 0.5, \ldots, w_{10} = 0.5$) at line 2. Next, at line 3, certain clusters from C are randomly selected based on a generated random number in [0,1]. If this random number is less than the corresponding weight of the cluster, the cluster is chosen. Line 4 then randomly selects one feature from each selected cluster in S to form a solution, namely *final_solution*, and records its prediction accuracy in *best_acc* (using a decision tree with *random_state* = 42 within a stratified 5-fold cross-validation) at line 5. Subsequently, lines 6–14 of DWES iteratively generate new solutions, updating *final_solution*, *best_acc*, and the weights of the superior clusters. Therefore, the superior clusters (implicitly recognized as containing more informative features) identified with their weights have a greater chance of being selected.

Fig. 8 clearly demonstrates that AGAwER, which leverages the external repository, exhibits better convergence than both the standalone external repository and DWES. Even the external repository performs better than DWES in the majority of cases. DWES only showed slightly better results in Leukemia-Multiclass and SRBCT compared to the external repository. In conclusion, it can be inferred that AGAwER achieves better convergence compared to DWES with the implementation described in Algorithm 5.

Table 6 summarizes the average prediction accuracy of a decision tree model when using different numbers of top features. We selected the top 10, 20, 30, 40, and 50 features identified by ranking algorithms and calculated the model's accuracy for each selection. This process was repeated for each number of features to determine the average accuracy. These results offer insights into how the model's performance varies depending on the number of most relevant features included. Table 6 also presents an intriguing investigation where the SHAP method is utilized as a ranking algorithm. In this approach, the SHAP explainer is initialized with a RandomForestClassifier model, which is then employed to compute SHAP values for the features within the dataset. These SHAP values reflect the impact of each feature on the model's predictions. Subsequently, the top features are selected based on their calculated impact, shedding light on the crucial factors influencing the model's predictive performance. This innovative use of SHAP as a ranking algorithm offers valuable insights into the significance of individual features in the predictive modeling process. The results in Table 6 are derived from the average of five stratified train–test splits with a test size of 0.2. From Table 6, two significant conclusions can be drawn:

- The results suggest that CEI generally demonstrates superior performance compared to other frequency-based rankers. This finding highlights the effectiveness of CEI in identifying relevant features for predictive modeling tasks.
- Despite the notable outcomes observed with SHAP on certain datasets such as GLI, Leukemia-Binary, and Leukemia-Multiclass, Table 6 emphasizes the effectiveness of combining CEI, MI, and FR to create a set of highly informative features. This indicates that the combination of multiple ranking algorithms yields superior results in identifying predictive features across various datasets.

Table 7 displays the number of unique features in the feature space after concatenating the top 50 features from CEI, MI, and FR (CEI | MI | FR). If each ranker identifies entirely distinct sets of top 50 features, the feature space size would be 150 (the last column from the left). However, as illustrated in Table 7, this number ranges from 121 in Leukemia-Binary to 145 in SMK for the column representing the concatenation of CEI | MI | FR. A lower number indicates greater overlap among the three rankers in recognizing common features. Table 7 also reveals an interesting observation: when concatenating MI and FR, more common features are identified. For example, in datasets like Leukemia-Binary, Leukemia-Multiclass, Covid-19, MLL, and SRBCT, both the concatenation of the top 50 features from CEI and FR (CEI | FR) and CEI and MI (CEI | MI) yield 100 unique features. This suggests that no common features have been recognized in these datasets between these concatenated pairs. This implies that CEI's selection criterion is distinct from both MI and FR. CEI prioritizes features differently compared to MI and FR, leading to unique sets of top features when combined with different methods. In summary, the presence of common features when concatenating MI and FR supports the idea of agreement or may imply that the methods are capturing similar underlying patterns or relationships in the data. Accordingly, there appears to be minimal relevance between CEI and FR or CEI and MR.

Fig. 9 illustrates the accuracy of Support Vector Machine (SVM) and Naive Bayes (NB) classifiers before and after applying CMF-AGAwER on the same datasets. This analysis demonstrates that the calculated accuracy is independent of the classifier and the classifiers' functions do not affect the feature selection process. The SVM in Fig. 9 uses an RBF kernel, while the NB classifier is a Gaussian NB. Both have the *random_state* parameter set to 42. The accuracy results shown in Fig. 9 are the averages calculated across three runs. To determine if there are significant differences in the use of different classifiers (DT, NB, and SVM), two statistical significance tests [23] are applied to

Algorithm 5 DWES

```
Input: feature_space
Output: final_solution, best_acc
 1: $C$ = complete_linkage (feature_space, 10)
 2: $W$ = Initialize the weights of $C$ equally to 0.5
 3: $S$ = Randomly select from $C$ using $W$
 4: final_solution = Randomly select a feature from each cluster of $S$
 5: best_acc = prediction_accuracy(final_solution)
 6: while stopping criteria have not been reached do
 7:     $S$ = Randomly select from $C$ using $W$
 8:     temp_solution = Randomly select a feature from each cluster of $S$
 9:     if prediction_accuracy(temp_solution) > best_acc then
10:         best_acc = prediction_accuracy(temp_solution)
11:         final_solution = temp_solution
12:         Update the $w_i$ values for the clusters in $S$    ▷ $w_i = w_i + (0.1 \times (1 - w_i))$
13:     end if
14: end while
15: Return final_solution, best_acc
```

Table 7
Number of unique features in the feature space after concatenating the top 50 features from CEI, MI, and FR (CEI | MI | FR), where | denotes the concatenation symbol.

Dataset	CEI | FR	CEI | MI	MI | FR	CEI | MI | FR
Colon	99	99	80	128
CNS	99	100	95	144
GLI	100	99	82	131
SMK	99	100	96	145
Leukemia-Binary	100	100	71	121
Leukemia-Multiclass	100	100	80	130
Covid-19	100	100	92	142
MLL	100	100	77	127
SRBCT	100	100	73	123

Table 8
Average NFE of AGAwER across 3 runs. The dataset LeukB refers to Leukemia-Binary, while LeukM refers to Leukemia-Multiclass. The classifier used is a decision tree.

Colon	CNS	GLI	SMK	LeukB	LeukM	Covid-19	MLL	SRBCT
788	1159	964	1319	742	764	1368	667	898

the distributions of the results. First, the Friedman test was conducted, showing distributions according to a chi-square value of 1.05 with 2 degrees of freedom, resulting in a p-value of 0.59. Second, the Iman and Davenport test, distributed according to an F-distribution with 2 and 16 degrees of freedom, yielded a statistic of 0.49 and a p-value of 0.61. In both cases, the p-values were higher than 0.05 (at a 95% confidence level), leading to the conclusion that the null hypothesis of no difference in distributions cannot be rejected. Therefore, it can be said that there is no significant difference in the performance of these classifiers. Consequently, a decision tree was chosen from the beginning as it serves as a baseline method with a simple training procedure, avoiding unnecessary complexity in the proposed approach.

5.4.1. *Efficiency analysis*

This section assesses the efficiency of the proposed CEI and AGAwER algorithms by analyzing their time complexity and Number of Functional Evaluations (NFE). We believe that analyzing time complexity and NFE offers a more insightful understanding of the methods' efficiency compared to directly calculating execution time. Execution time can vary significantly due to factors such as implementation specifics and hardware requirements. Therefore, referencing time complexity and NFE provides a global, implementation-independent, and hardware-independent comprehension of efficiency.

Time complexity of CEI. As part of a further study, the time complexity of CEI has also been investigated. As per the definition of CEI as a frequency-based ranker in Section 4.1, each feature undergoes sorting, and CEI is then calculated based on the reordered response variable. Sorting involves a time complexity of $O(n \log n)$ using a merge sort algorithm for each feature. Additionally, Algorithm 1 calculates the classification error at lines 3–9, where each class label l requires consideration of all n rows in the worst-case scenario. Consequently, the overall time complexity can be estimated as $O(m(n \log n + ln)) = O(mn \log n + mln)$, where m, n, and l represent the number of columns, number of rows, and number of class labels, respectively.

NFE of AGAwER. The average Number of Function Evaluations (NFE) for AGAwER across each dataset is calculated based on 10 runs, as shown in Table 8. The table clearly demonstrates that AGAwER evaluates a minimal number of feature subsets relative to the entire search space, which can be estimated from the feature space. In the current AGAwER setup, each solution with variable length can range in size from 1 to 10. Consequently, in the worst-case scenario where two solutions of size 10 undergo crossover, the resulting offspring can have a maximum size of 18. This indicates that the optimal solution can ultimately range in size from 1 to 18. Therefore, to determine the total number of ways to distribute 150 distinct variables into sets of sizes ranging from 1 to 18, where the order of variables within each set does not matter is $\sum_{k=1}^{18} \binom{150}{k}$. This calculation represents a computationally intensive task, but provides an estimation of the search space, allowing the vast number of possible solutions to be explored in the search for the optimal solution. The results shown in Table 8 clearly demonstrate that AGAwER evaluates only a small number of feature subsets across various datasets.

5.4.2. *Comparative analysis of CMF-AGAwER vs. selected related works*

Table 9 presents a comparison of the average accuracy and subset length of CMF-AGAwER with several related works. All the methods in Table 9 have been reimplemented using the same settings, including the type of cross-validation, classifier used, and other relevant parameters. Among the works listed in Table 9, WOA-MC [16], ATFS [11], EMC-DWES [13], MPR-MDES [14], and DMC-GAwER [17] incorporate a frequency-based ranker into their proposals. Additionally, WOA-MC, EIT-bBOA, EMC-DWES, MPR-MDES, and DMC-GAwER explicitly employ meta-heuristic algorithms, while ATFS and PEFS indirectly use non-dominated sorting in their approach. CMF-AGAwER utilizes an enhanced GA as its meta-heuristic and integrates the frequency-based ranker CEI. These related works were selected for comparison because they share many similarities with CMF-AGAwER. Table 9 shows that the proposed CMF-AGAwER generally performs better than other methods, achieving higher average accuracy in 5 out of 9 datasets. The automatic feature selection of CMF-AGAwER generally results in smaller subset lengths. Another advantage of CMF-AGAwER over WOA-MC, ATFS, and DMC-AGAwER is its capability to handle both binary and multiclass classification problems.

6. Conclusions

This paper introduces a hybrid (two-stage) feature selection method called CMF-AGAwER. In the first stage, the top 50 features are individually selected based on Mutual Information (MI), Fisher Ratio (FR), and a newly proposed Classification Error Impurity (CEI) frequency-based ranker. Subsequently, these top 50 features from each ranker are combined to form an ensemble, constituting the feature space containing the most informative features. Then, the Adaptive Genetic Algorithm with External Repository (AGAwER) is employed as a wrapper feature selection method to automatically identify the optimal set of features. AGAwER utilizes an adaptive strategy to adjust crossover and mutation rates, and it leverages an external repository to aid in exploration.

The paper highlights that CEI generally achieves superior results among frequency-based rankers due to its ability to recognize both linear and non-linear patterns. The time complexity of CEI and the number of function evaluations (NFE) of AGAwER demonstrate that the proposed method is efficient for feature selection. Additionally, the results presented in the paper indicate that CMF-AGAwER can be effectively applied to small-sample high-dimensional classification datasets.

Table 9
Comparison of the proposed method CMF-AGAwER (in the rightmost column) with state-of-the-art related methods. The dataset LeukB refers to Leukemia-Binary, while LeukM refers to Leukemia-Multiclass. SL indicates the average subset length, and Acc denotes the mean accuracy across 3 runs. NA indicates that the value is Not Applicable. The classifier used is a decision tree.

Dataset	WOA-MC [16]		ATFS [11]		PEFS [12]		EIT-bBOA [21]		EMC-DWES [13]		MPR-MDES [14]		DMC-GAwAR [17]		CMF-AGAwER	
	SL	Acc	SL	Acc	SL	Acc	SL	Acc	SL	Acc	SL	Acc	SL	Acc	SL	Acc
Colon	10	0.74	14	0.83	40	0.76	30	0.86	6	0.91	6	0.94	10	**0.96**	6	0.94
CNS	10	0.72	8	0.68	20	0.66	30	0.84	26	0.82	6	0.88	10	0.91	9	**0.93**
GLI	10	0.86	6	0.83	100	0.85	30	0.84	29	0.91	5	0.93	10	**0.96**	9	**0.96**
SMK	10	0.66	9	0.70	80	0.71	30	**0.82**	33	0.70	6	0.75	10	0.80	8	0.77
LeukB	10	0.92	4	0.92	10	0.94	30	0.89	17	0.97	3	0.99	10	**1**	9	0.99
LeukM	NA	NA	NA	NA	10	0.85	30	0.85	20	**0.97**	4	0.95	NA	NA	5	**0.97**
Covid-19	NA	NA	NA	NA	10	0.38	30	**0.94**	25	0.75	7	0.72	NA	NA	8	0.77
MLL	NA	NA	NA	NA	80	0.90	30	0.88	6	0.96	5	0.95	NA	NA	8	**1**
SRBCT	NA	NA	NA	NA	60	0.83	30	0.92	12	0.94	7	0.94	NA	NA	7	**0.96**

Looking ahead, the capability of CMF-AGAwER, including a detailed assessment of CEI's applicability, can be evaluated across various domains and datasets, considering diverse real-world applications and scenarios. The external repository introduced in this paper can also be evaluated for solving other problems using Genetic Algorithms (GAs).

Additionally, most feature selection papers address the feature selection problem numerically, particularly when using correlation techniques. However, numeric correlation could be more authentic if plausible causality is also considered. Therefore, one direction for future work is to incorporate causality analysis into feature selection or multicollinearity test.

Finally, much like the ''No Free Lunch'' (NFL) theorem, which asserts that no single optimization algorithm is universally superior for every possible problem, ongoing research and exploration in the field of feature selection methods continue, acknowledging the absence of a universally optimal technique.

CRediT authorship contribution statement

Hossein Nematzadeh: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization. **José García-Nieto:** Writing – review & editing, Validation, Supervision, Conceptualization. **Ismael Navas-Delgado:** Writing – review & editing, Supervision. **José F. Aldana-Montes:** Supervision, Project administration, Funding acquisition.

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

The authors have included a link to the GitHub page containing the implementation and datasets at the end of the abstract.

Acknowledgments

This work has been partially funded by grants (funded by MCIN/AEI/10.13039/501100011033/) PID2020-112540RB-C41, AETHER-UMA (A smart data holistic approach for context-aware data analytics: semantics and context exploitation), and QUAL21 010UMA (Junta de Andalucía). It is also funded for open access charge: Universidad de Málaga/CBUA.

References

[1] N. Sánchez-Maroño, O. Fontenla-Romero, B. Pérez-Sánchez, Classification of microarray data, in: V. Bolón-Canedo, A. Alonso-Betanzos (Eds.), Microarray Bioinformatics, 2019, pp. 185–205, http://dx.doi.org/10.1007/978-1-4939-9442-7_8.

[2] C. Yan, J. Ma, H. Luo, A. Patel, Hybrid binary coral reefs optimization algorithm with simulated annealing for feature selection in high-dimensional biomedical datasets, Chemometr. Intell. Lab. Syst. 184 (2019) 102–111, http://dx.doi.org/10.1016/j.chemolab.2018.11.010.

[3] P. Agarwalla, S. Mukhopadhyay, GENEmops: Supervised feature selection from high dimensional biomedical dataset, Appl. Soft Comput. 123 (2022) 108963, http://dx.doi.org/10.1016/j.asoc.2022.108963.

[4] T. Thaher, H. Chantar, J. Too, M. Mafarja, H. Turabieh, E.H. Houssein, Boolean particle swarm optimization with various evolutionary population dynamics approaches for feature selection problems, Expert Syst. Appl. 195 (2022) 1–30, http://dx.doi.org/10.1016/j.eswa.2022.116550.

[5] S. Osama, H. Shaban, A.A. Ali, Gene reduction and machine learning algorithms for cancer classification based on microarray gene expression data: A comprehensive review, Expert Syst. Appl. 213 (2023) 118946, http://dx.doi.org/10.1016/j.eswa.2022.118946.

[6] X. Gong, J. Wang, Q. Ren, K. Zhang, E.-S.M. El-Alfy, J. Mańdziuk, Embedded feature selection approach based on TSK fuzzy system with sparse rule base for high-dimensional classification problems, Knowl.-Based Syst. 295 (2024) 111809, http://dx.doi.org/10.1016/j.knosys.2024.111809.

[7] K. Kanti Ghosh, S. Begum, A. Sardar, S. Adhikary, M. Ghosh, M. Kumar, R. Sarkar, Theoretical and empirical analysis of filter ranking methods: Experimental study on benchmark DNA microarray data, Expert Syst. Appl. 169 (2021) 114485, http://dx.doi.org/10.1016/j.eswa.2020.114485.

[8] G. Sahebi, P. Movahedi, M. Ebrahimi, T. Pahikkala, J. Plosila, H. Tenhunen, GeFeS: A generalized wrapper feature selection approach for optimizing classification performance, Comput. Biol. Med. 125 (2020) 103974, http://dx.doi.org/10.1016/j.compbiomed.2020.103974.

[9] H. Saadatmand, M.-R. Akbarzadeh-T, Set-based integer-coded fuzzy granular evolutionary algorithms for high-dimensional feature selection, Appl. Soft Comput. (2023) 110240, http://dx.doi.org/10.1016/j.asoc.2023.110240.

[10] A. Jiménez-Cordero, J.M. Morales, S. Pineda, A novel embedded min-max approach for feature selection in nonlinear Support Vector Machine classification, European J. Oper. Res. 293 (1) (2021) 24–35, http://dx.doi.org/10.1016/j.ejor.2020.12.009.

[11] S. Abasabadi, H. Nematzadeh, H. Motameni, E. Akbari, Automatic ensemble feature selection using fast non-dominated sorting, Inf. Syst. 100 (2021) 101760, http://dx.doi.org/10.1016/j.is.2021.101760.

[12] A. Hashemi, M. Bagher Dowlatshahi, H. Nezamabadi-pour, A pareto-based ensemble of feature selection algorithms, Expert Syst. Appl. 180 (2021) 115130, http://dx.doi.org/10.1016/j.eswa.2021.115130.

[13] H. Nematzadeh, J. García-Nieto, I. Navas-Delgado, J.F. Aldana-Montes, Automatic frequency-based feature selection using discrete weighted evolution strategy, Appl. Soft Comput. 130 (2022) 109699, http://dx.doi.org/10.1016/j.asoc.2022.109699.

[14] H. Nematzadeh, J. García-Nieto, J.F. Aldana-Montes, I. Navas-Delgado, Pattern recognition frequency-based feature selection with multi-objective discrete evolution strategy for high-dimensional medical datasets, Expert Syst. Appl. 249 (2024) 123521, http://dx.doi.org/10.1016/j.eswa.2024.123521.

[15] S. Abasabadi, H. Nematzadeh, H. Motameni, E. Akbari, Hybrid feature selection based on SLI and genetic algorithm for microarray datasets, J. Supercomput. 78 (2022) 19725–19753, http://dx.doi.org/10.1007/s11227-022-04650-w.

[16] H. Nematzadeh, R. Enayatifar, M. Mahmud, E. Akbari, Frequency based feature selection method using whale algorithm, Genomics 111 (6) (2019) 1946–1955, http://dx.doi.org/10.1016/j.ygeno.2019.01.006.

[17] H. Nematzadeh, J. Mani, Z. Nematzadeh, E. Akbari, R. Mohamad, Distance-based mutual congestion feature selection with genetic algorithm for high-dimensional medical datasets, 2024, arXiv:2407.15611, URL https://arxiv.org/abs/2407.15611.

[18] M. Negnevitsky, Artificial Intelligence: A Guide to Intelligent Systems, Pearson Education Limited, 2011, URL https://books.google.es/books?id=8mmpBwAAQBAJ.

[19] F. Amini, G. Hu, A two-layer feature selection method using genetic algorithm and elastic net, Expert Syst. Appl. 166 (2021) 114072, http://dx.doi.org/10.1016/j.eswa.2020.114072.

[20] H. Pan, S. Chen, H. Xiong, A high-dimensional feature selection method based on modified Gray Wolf Optimization, Appl. Soft Comput. 135 (2023) 110031, http://dx.doi.org/10.1016/j.asoc.2023.110031.

[21] Z. Sadeghian, E. Akbari, H. Nematzadeh, A hybrid feature selection method based on information theory and binary butterfly optimization algorithm, Eng. Appl. Artif. Intell. 97 (2021) 104079, http://dx.doi.org/10.1016/j.engappai.2020.104079.

[22] M.A. Ganjei, R. Boostani, A hybrid feature selection scheme for high-dimensional data, Eng. Appl. Artif. Intell. 113 (2022) 104894, http://dx.doi.org/10.1016/j.engappai.2022.104894.

[23] D. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, Fifth Edition, CRC Press, 2020, http://dx.doi.org/10.1201/9780429186196.