Information Sciences 698 (2025) 121785

Contents lists available at ScienceDirect

Information Sciences

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

Mining interpretable rules with MCRM: A novel rule mining algorithm with inherent feature selection and discretization

Mohammadreza Khosravi, Alireza Basiri *

Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran

ARTICLE INFO

Keywords:
Data mining
Rule-based classification
Artificial bee colony
Multivariate discretization
Feature selection

ABSTRACT

This paper presents MCRM, a new meta-heuristic rule learning algorithm based on the Pittsburgh approach. MCRM extracts all rules simultaneously with a single execution of a meta-heuristic algorithm. The proposed algorithm uses the artificial bee colony (ABC) as a meta-heuristic optimization algorithm. In the MCRM algorithm, classification rules are extracted as a result of a multivariate discretization process and the task of the ABC optimization algorithm is to guide the discretization process. The introduced structure for rule learning guarantees that the extracted rules are mutually exclusive. It also allows the algorithm to ignore the less important features while learning the rules and perform the feature selection operation implicitly. Also, the algorithm uses a special post-pruning step for pruning the final rules, which helps to increase the interpretability of the final rules by reducing the number and length of the rules. The proposed algorithm is compared with 15 other classification algorithms on 32 datasets from the OpenML website and the UCI machine learning repository. The experimental results show the promising performance of the proposed algorithm compared to other classifiers applied in the experiments.

1. Introduction

One of the common data mining techniques is data classification. Classification is a process in which the relationship between input features and predetermined objectives is discovered. The structure with which the mentioned relationship is expressed is called a model. Finally, such model, called classifier, predicts labels for new instances. So far, many classification methods have been proposed, such as methods based on neural networks, decision tree induction, rule extraction, Bayes theory, and support vector machines, each of which has its advantages and disadvantages compared to other methods. This has caused each of them to receive more attention in a number of specific fields. Among them, rule-based classifiers and decision trees are generally preferred due to the interpretable nature of the classification structures they create from training data [1]. This interpretability allows humans to understand the reasoning behind the model's predictions, which can be crucial in fields such as medicine or finance.

In data mining literature, the effectiveness of rule-based classification algorithms has been proven in various applications. For example, in predicting customer churn [2], these algorithms enable organizations to retain old customers and attract new ones. In the financial field [3], they can help banks and financial institutions validate customers to make loan decisions. Their effectiveness in the medical field has been shown in applications such as predicting the probability of death due to heart diseases [4] and detection of epilepsy [5]. They are also used in fields such as bioinformatics [6], text classification [7], and detection of phishing websites [8].

* Corresponding author.
E-mail address: basiri@iut.ac.ir (A. Basiri).

https://doi.org/10.1016/j.ins.2024.121785
Received 21 October 2023; Received in revised form 31 October 2024; Accepted 18 December 2024
Available online 19 December 2024

Rules in rule-based classification algorithms typically follow a specific format, as shown in equation (1). These rules consist of two main parts: the antecedent and the consequent. The antecedent part contains the conditions that apply to the features of each instance. And the consequent part specifies the class of instances covered by the rule. If an instance satisfies all the conditions of the antecedent part of a rule, the class of the consequent part of the rule is assigned to it.

$$\text{IF} \ < \text{conditions} > \ \text{THEN} \ < \text{class} > \tag{1}$$

Various methods for learning rules have been proposed, some of which use meta-heuristic optimization algorithms. These algorithms can be classified into two categories based on how they represent individuals. The first group is algorithms based on the Michigan approach, where each individual is a rule and evolves to cover a part of the classification model. The second group is algorithms based on the Pittsburgh approach, in which each individual is a combination of several rules and is a potential classifier [9]. Therefore, the evaluation of each individual in algorithms based on the Pittsburgh approach is equivalent to the evaluation of a candidate classifier. In contrast, the evaluation of each individual in the algorithms based on the Michigan approach is equivalent to the evaluation of only one of the candidate rules in the classifier, and the effect of the rules is not evaluated on the classifier as a whole. On the other hand, the algorithms that have been presented so far based on the Pittsburgh approach have two major flaws compared to the Michigan approach; First, it is simply not possible to determine the number of rules for each individual [9]. And secondly, the need to evolve multiple sets of rules simultaneously imposes high computational costs on the algorithm [10].

In this article, a meta-heuristic rule learning algorithm called MCRM (Multivariate discretization based Classification Rule Miner) is presented. MCRM utilizes the artificial bee colony (ABC) meta-heuristic optimization algorithm [11] to learn classification rules. The algorithm is based on the Pittsburgh approach, but its structure alleviates the limitations associated with earlier approaches. In essence, MCRM combines aspects of both Pittsburgh and Michigan approaches. Each individual in MCRM represents a potential classifier, aligning with the Pittsburgh approach. However, the conciseness of each individual, limited to the number of features, offers advantages similar to the Michigan approach. This conciseness allows the algorithm to more effectively search for optimal solutions within the problem space. Additionally, in MCRM, the rules of a potential classifier are created implicitly. This eliminates the need to predetermine the initial number of rules. Despite the fixed length of individuals, the number of rules for each potential classifier can vary during the algorithm's execution.

The general idea of this method involves selecting a cut point for each feature while considering the location of cut points for other features. This ensures that the selected cut point, along with other cut points, divides the problem space in a way that separates the training instances into distinct regions with greater purity. This method for zoning the problem space causes the rules to have common boundaries, eliminating the issue of rule collision, a common problem in many rule-based classification algorithms. Furthermore, due to the introduced structure, it is easy to ignore some less important features during the learning of the rules. In this way, the effect of the removed features on the whole classification rules is ignored and it leads to the feature selection process.

Since MCRM extracts classification rules through a multivariate discretization process, the algorithm's effectiveness relies on its ability to separate data within the multidimensional space. For proper separation, the algorithm considers the relationships between features in the problem space. Another goal of the algorithm is to extract simple and interpretable rules. Three key factors contribute to the simplicity and interpretability of the generated rules. The first factor is the feature selection process, which reduces the number of features and consequently limits the variety of features appearing in the antecedent (IF) part of each rule. In essence, the number of selected features defines the maximum length of the rules. The second factor is the rule pruning stage, which is introduced as the integration of areas. This method not only reduces the overall length of the rules but also decreases their total number. The third factor is inherent to the algorithm's structure and ensures that the variety of conditional expressions for a single feature remains low across all the rules, ultimately resulting in two distinct cases for each feature.

In summary, our research makes the following main contributions:

- This paper introduces a novel classification rule learning algorithm based on the ABC meta-heuristic optimization algorithm. This algorithm incorporates discretization and feature selection as inherent aspects of the rule learning process, eliminating the need for separate steps. Moreover, the problem space representation within the algorithm leads to the generation of highly interpretable rules. The reason that leads to the high interpretability of the algorithm is that in the process of learning the rules, the values of each feature are divided into two intervals based on only one cut point. The conditions of the antecedent part of the rules are formed binary, based on these cut points. In such a way that either the values of that attribute must be less than the cut point or more than it. Since the cut point for a feature is fixed among all the rules; Only two conditions can be expressed for each feature, and each rule can have only one of these two conditions for the desired feature. This structure in the definition of the antecedent part of the rules makes the variety of the conditions of the antecedent part of the rules to be very low; Therefore, the analysis of the rules is easily done. In addition to this, the feature selection operator, which is in the essence of the algorithm, reduces the number of features and, as a result, the number of conditions in the antecedent part of the rules, which has a significant effect on increasing the interpretability of the rules.
- We also propose a post-processing step that merges rules for improve interpretability. This step reduces the overall number of rules and the number of conditions within their antecedent parts.
- The performance of the proposed algorithm, measured in terms of accuracy and the number of extracted rules, was compared to established classification algorithms encompassing various approaches. The experiments demonstrate the effectiveness of the embedded feature selection process in reducing the number of features used in the rules' antecedent parts. Furthermore, they highlight the post-processing step's positive impact on reducing both the number and length of the final rules.

In the following, the content of the article is organized as follows: Section 2 provides a brief overview of meta-heuristic optimization algorithms and rule-based classification algorithms. Section 3 introduces the fundamental concepts of the Artificial Bee Colony (ABC) algorithm. Section 4 delves into the details of the proposed algorithm based on the ABC algorithm. This section begins by examining the algorithm's overall structure, followed by an explanation of how the ABC optimization algorithm is modified to support this structure. Finally, the operation of the key components within the proposed algorithm is explored. Section 5 analyzes the performance of the proposed algorithm through experimental results. The paper concludes with a summary of the findings in Section 6.

2. Related works

The main focus of this research is rule-based classification, and specially meta-heuristic rule-based classification. Therefore, the contents of this section are organized in two separate subsections. In the first subsection, a brief history of some famous meta-heuristic optimization algorithms will be presented, and in the second subsection, a brief history of rule-based classification algorithms will be presented.

2.1. Meta-heuristic optimization

In optimization problems, the goal is to find values for a set of parameters that optimize a given objective. Meta-heuristic algorithms are a type of stochastic optimization method used to find solutions to complex problems. The ability of meta-heuristic algorithms to global exploration of the problem space in a reasonable time, as well as their lack of dependence on the nature of the problem, has caused these algorithms to receive more attention [12]. The Ant Colony Optimization (ACO) algorithm [13] is one of the meta-heuristic optimization algorithms inspired by the foraging behavior of ants. ACO probabilistically builds a solution by iteratively adding solution components to partial solutions. Typically, good quality solutions emerge as a result of collective interaction among ants, which is obtained via indirect communication mediated by the information of pheromone trail values. The Artificial Bee Colony optimization (ABC) algorithm [11] is inspired by the foraging behavior of honey bees. In this algorithm, the behavior of three groups of bees is simulated. Employee and onlooker bees are responsible for improving current solutions and scout bees are responsible for finding new solutions and exploring the problem space. The Imperialist Competitive Optimization Algorithm (ICA) [14] is proposed and inspired by Imperialistic competition. Population individuals in this algorithm are called countries. Countries fall into two types: colonies and imperialists, and all together form some empires. Imperialistic competition among these empires forms the basis of the algorithm. In the Particle Swarm Optimization (PSO) algorithm [15], the main idea is to simulate the collective behavior of social animals; In particular, bird flocking and fish schooling behaviors. In this algorithm, each population individual is called a particle. Each particle changes its current position on the problem space based on a combination of two components; One of them is attraction towards the best solution that it individually has found, and the other is attraction towards the best solution found by the particles in its neighborhood. The Biogeography-Based Optimization (BBO) algorithm [16] is inspired by how species migrate from one habitat to another. The idea of migration among different habitats is modeled by sharing some features of high-quality solutions with low-quality solutions. Sharing features from high-quality solutions to low-quality solutions may improve their quality. The Spider Monkey Optimization (SMO) algorithm [17] is presented inspired by the foraging behavior of spider monkeys. The foraging behavior of spider monkeys places them in the category of fission–fusion social structure (FFSS) based animals. Accordingly, when food resources are scarce, they divide into smaller subgroups to find food but remain part of a larger social unit. This algorithm has modeled the behavior of spider monkeys in six main stages to achieve the optimal solution in the problem space. The Differential Evolution (DE) algorithm [18] is one of the evolutionary optimization algorithms. The basic idea of this algorithm is the self-organization of the step length during the evolution process, which is possible with differential vectors. The Equilibrium Optimization (EO) algorithm [12] is inspired by the control volume mass balance phenomenon in physics. To achieve the global optimum, this algorithm uses the concept of the equilibrium pool, which consists of the four best individuals in the population and their arithmetic mean.

In addition to data mining, which is the main topic of the article, meta-heuristic optimization algorithms are used in other real-world problems. For example, [19] has shown the application of meta-heuristic optimization algorithms in engineering design problems. In [20], the efficiency of meta-heuristic algorithms in solving the wind farm layout optimization problem is shown as a comparative study. In the field of manufacturing industries, the article [21] has presented an optimization approach based on the artificial bee colony algorithm for the optimal selection of cutting parameters in multi-pass turning operations. [22] in a comparative study shows the application of meta-heuristic optimization algorithms in reliability-based design optimization. In the automobile industry, meta-heuristic algorithms can play a significant role in obtaining superior optimized designs for different vehicle components. In [23], the hunger game search (HGS) algorithm is applied to optimize the automobile suspension arm.

2.2. Rule-based classification

So far, many rule-based classification methods have been proposed, which can be classified in different ways. Some of these algorithms first create a decision tree from training instances and then extract rules from it. As an example of this category, we can refer to the C4.5 algorithm [24]. Another group of rule-based classification algorithms follows the sequential covering approach to learn rules. In the sequential covering approach, initially, the list of rules is empty and the rules are learned sequentially (one at a time). Each time, the learned rule is added to the list of classifier rules, and all instances covered by it are removed from the training set. IREP [25] is an example of sequential covering algorithms that was presented in 1994. In this algorithm, first, the rules are defined in general and by adding conditions to the antecedent part, they become more specific and accurate rules. After that, the RIPPER algorithm [26] was

presented by applying three basic changes to IREP to improve it. In 1998, a combination of RIPPER and C4.5 algorithms called PART [27] was introduced. This algorithm extracts rules by iteratively constructing partial decision trees. The RACER algorithm [28] has a different approach compared to the sequential covering algorithms. It starts with N initial rules (where N is the number of training instances). This algorithm uses a specific rule representation that enables it to consider each instance in the training data as an initial rule. To retrieve an applicable rule set, RACER tries to combine the initial rules.

Another group of rule-based classification algorithms uses meta-heuristic optimization algorithms to learn rules. These algorithms usually first provide a representation of the problem space, which is a mapping from the problem space to the solution space. Then, they use a meta-heuristic optimization algorithm to search for the best solution in the solution space. Meta-heuristic rule learning algorithms can be divided into two categories. The first category is algorithms based on the Pittsburgh approach, where each individual of the population is a potential solution. In other words, each individual of the population is considered as a classifier. Therefore, evaluating each population individual in these algorithms is equivalent to evaluating a candidate classifier. This is one of the advantages of the Pittsburgh approach. On the other hand, the proposed algorithms based on the Pittsburgh approach suffer from two basic problems. First, it is not easy to determine the number of final classification rules at the beginning because there is uncertainty about the number of rules. A certain number of rules encoded in each individual of the population may be insufficient to create a suitable classification model. Meanwhile, some algorithms support a variable number of rules for population individuals, but they must use special search mechanisms to manage the different lengths of population individuals [9]. Second, the need to evolve multiple sets of rules simultaneously brings heavy computational costs for these algorithms [10]. As examples of algorithms based on the Pittsburgh approach, the following can be mentioned: In [29], for the classification of medical data, the first rule-based classification algorithm based on the differential evolution (DE) optimization algorithm is presented. In this algorithm, the maximum number of rules for each individual (potential classifier) is determined by the user. The cAnt-Miner$_{PB}$ algorithm [30] is introduced based on a combination of the Pittsburgh approach and the sequential covering strategy and uses the Ant Colony Optimization (ACO) algorithm to extract rules. The goal of this algorithm is to converge to the best list of rules, instead of converging to a list of best rules like sequential covering algorithms. To achieve this goal, each individual in the population is considered a potential classifier, and its rules are extracted by a sequential covering approach. Therefore, in this algorithm, there is no need to initially determine the number of individual rules. In [31], the first application of the Biogeography Optimization (BBO) algorithm for rule-based classification is presented. In this algorithm, the length (number of rules) of each individual of the population is sampled from a predetermined interval; Therefore, the length of individuals can be different. Due to the different lengths of the individuals, a special mechanism is considered for searching the solution space.

The second category of meta-heuristic rule learning algorithms is based on the Michigan approach. In these algorithms, each individual in the population represents a rule and evolves to cover a part of the whole classification model. One of the advantages of this approach is that the length of each individual is small, so it has lower computational costs than the Pittsburgh approach. Along with its advantages, this structure for displaying individuals also has disadvantages. One of these disadvantages is that each rule is evaluated independently of other rules and the effect of the rules on the final classifier is not evaluated. Some of the algorithms in the Michigan category learn the entire classifier rules by running the meta-heuristic algorithm once; Such as the CoABCMiner algorithm [9], which is based on the Artificial Bee Colony (ABC) algorithm. Such algorithms should not allow the population to converge; Because all the individuals of the population are needed to create the classifier. Some other algorithms in the Michigan category, which are known as sequential covering algorithms, learn the rules of a classifier with sequential running of the meta-heuristic algorithm. As an example of these algorithms, in 2002, the first rule-learning algorithm that used the Ant Colony Optimization (ACO) algorithm to learn rules was introduced with the name Ant-Miner [32]. This algorithm is only able to work with discrete features. The cAnt-Miner algorithm [33] is presented as an extension of the Ant-Miner algorithm. This algorithm incorporates an entropy-based discretization method in order to cope with continuous features during the rule-learning process. In [34] a hybrid rule-based classifier namely, ant colony optimization/genetic algorithm ACO/GA is introduced to improve the classification accuracy of the Ant-Miner classifier by using GA. In [35], a privacy-preserving rule-based classifier based on the Artificial Bee Colony (ABC) optimization algorithm is introduced. BeeMiner [36] is another algorithm based on ABC optimization algorithm. In [37], a rule-based classification algorithm based on the Spider Monkey Optimization (SMO) algorithm called SM-RuleMiner is proposed for diabetes diagnosis. BeeMiner and SM-RuleMiner are developed for continuous features. In [38], a rule-based classification algorithm based on the Particle Swarm Optimization (PSO) algorithm for heart disease diagnosis is presented. In addition to these, we can mention other cases: For example, the CORER algorithm [39] based on the Imperialist Competitive Optimization (ICA) algorithm and the DEOA-CRM algorithm [40] based on the Equilibrium Optimization (EO) algorithm are presented, which are only able to work with discrete features. Also, the LGBBO-RuleMiner algorithm [3] based on the biogeography-based optimization (BBO) algorithm, and the RIM-GP algorithm [41] based on the genetic programming (GP) evolutionary algorithm are presented.

3. Background

Meta-heuristic rule learning algorithms need a meta-heuristic optimization algorithm to search the solution space and find the optimal solution. In the history of meta-heuristic rule learning algorithms, the use of meta-heuristic optimization algorithms is very diverse. This issue can depend on factors such as the compatibility of the optimization algorithm with the rule learning method and the representation of the problem space. Therefore, it cannot be said that a specific meta-heuristic optimization algorithm is preferable to other ones for the rule learning problem.

In this research, we chose the artificial bee colony meta-heuristic optimization algorithm (ABC) to search the solution space. The reason for this choice is the compatibility of the ABC algorithm with the proposed rule learning method and also the proven efficiency

of this algorithm in the history of rule learning as evidenced in studies like [9,35,36]. The ABC equations were found to be particularly well-suited to our problem representation, requiring only minor adjustments for integration into our algorithm. This adaptability, combined with the algorithm's demonstrated effectiveness in similar rule learning contexts, made ABC a strong candidate for our research.

Artificial Bee Colony (ABC) is a meta-heuristic optimization algorithm presented in 2005 by Karaboga [11]. In this algorithm, agents include three groups of bees; Employee, onlooker, and scout bees. Employee bees make up half of the bee population. Each employee bee acquires a food source and evaluates its neighborhood by local search. It then returns to the hive and shares the information it has gained from its assessments with the other bees. Each onlooker bee selects a food source based on the information provided by the employee bees and helps the corresponding employee bee find a better food source. If a food source does not improve after a certain number of attempts, the corresponding employee bee temporarily abandons its responsibility as an employee bee and becomes a scout bee to find a new food source. Here, each food source corresponds to a potential solution in the problem space. The amount of nectar of each food source is equivalent to the fitness of the desired solution.

Based on the explanations given, the steps of the ABC algorithm can be described as follows [11,42].

Step 1) Initialization: First, each employee bee selects a solution (food source) randomly from the problem space according to (2).

$$X_i^j = X_{\min}^j + rand(0,\ 1) \times (X_{\max}^j - X_{\min}^j) \tag{2}$$

Here $i \in \{1, 2, \cdots, ne\}$ where ne is the number of employee bees. X_{min}^j and X_{max}^j, while $j \in \{1,2,\cdots,D\}$, show the minimum and maximum values of the j^{th} parameter of the D-dimensional problem space, respectively. Also, $rand(0,1)$ is a random number between 0 and 1.

Step 2) Evaluating the fitness value of food sources: In this step, the fitness value of the solution of each of the employee bees is calculated based on the fitness function related to the problem we are looking to optimize.

Step 3) Improving the solution of employee bees: In each iteration of the algorithm, each employee bee first tries to improve its solution according to (3) by local search.

$$X_i^{*j_{rand}} = X_i^{j_{rand}} + rand(-1,\ 1) \times (X_i^{j_{rand}} - X_k^{j_{rand}}) \tag{3}$$

where $j_{rand} \in \{1, 2, \cdots, D\}$ is the index of one of the parameters of the problem space that is chosen randomly. $X_i^{j_{rand}}$ represents the current value and $X_i^{*j_{rand}}$ represents the new value of the selected parameter for solution i. $X_k^{j_{rand}}$ is the value of the selected parameter for solution k so that $i \neq k$. And $rand(-1,1)$ is a random number between -1 and 1.

According to (4) if the fitness value of the current solution is higher than the new solution, the current solution is kept. Otherwise, the employee bee abandons the current solution and adopts the new solution.

$$X_i = \begin{cases} X_i, & \text{if } F(X_i) > F(X_i^*) \\ X_i^*, & \text{else} \end{cases} \tag{4}$$

Here X_i is the current solution and X_i^* is the new solution. and F determines the fitness value of each solution.

Step 4) Recruiting onlooker bees: After each employee bee has updated its position, it's time to recruit onlooker bees. Each onlooker bee chooses an employee bee to help it with a probability of (5).

$$P_i = \frac{F(X_i)}{\sum_k^{ne} F(X_k)} \tag{5}$$

where, P_i is the probability of selecting the employee bee corresponding to solution i by the onlooker bees.

After an onlooker bee has made its choice and is recruited by an employee bee; With the same mechanism as the employee bees, that is, by using the equations (3) and (4), it tries to improve the desired solution.

Step 5) Memorizing the best solution: In this step, the best solution achieved so far, which has the highest fitness, is memorized.

Step 6) Scout bees: If the solution of an employee bee does not improve after a certain number of attempts, the employee bee abandons its solution and takes a new random position in the problem space. This specific number is determined by the "limit" parameter, which balances exploration and exploitation.

Steps 3 to 6 are repeated until the termination condition is satisfied. The best solution is the (sub)optimum solution for the problem.

4. Proposed rule learning method: MCRM

4.1. The general structure of the algorithm

In this section, we will present a new rule-based classifier, which takes advantage of the Pittsburgh approach and is less affected by the problems of this approach. As mentioned in the previous sections, in the Pittsburgh approach, each member of the population consists of several rules and is considered a potential classifier. Accordingly, the rule learning algorithms based on the Pittsburgh approach mainly suffer from two basic problems: The large length of the population members and uncertainty in determining the appropriate number of rules for each classifier. In addition to the problems of the Pittsburgh approach, one of the challenges of meta-heuristic rule learning algorithms is the collision problem of the final rules extracted by them. This problem, in addition to reducing the

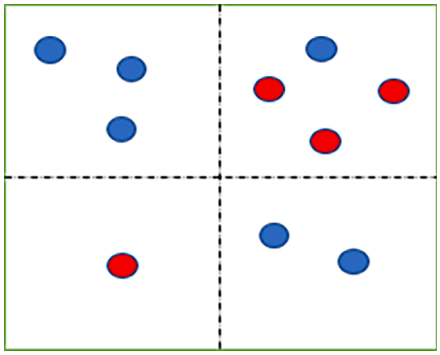

Fig. 1. An example for grid structure in a two-dimensional space.

interpretability of the rules, leads to uncertainty in determining the label of instances that are in the common range of rules of different classes. In the following, we introduce a structure and a method for learning the rules that do not have the stated problems.

We want to classify some labeled data expressed by features F_1 to F_m in different areas; In such a way that the purity of each area is the highest value in terms of the sameness of the labels of the instances in that area. To do this, we consider a fixed number of areas; In such a way that both adjacent areas have a common boundary. For better understanding, consider the problem space as two-dimensional. In this case, we put a grid structure on the problem space so that each cell of this grid is an area. For the grid structure to be able to separate the instances well, the boundaries between its areas should be in a suitable position on the problem space. To construct such a structure, we consider a cut point for the values of each feature of the problem. The values of each feature are divided into two categories based on the cut point corresponding to that feature; Values that are less than the cut point and values that are greater than it. The set of these cut points defines our grid structure in the problem space. This grid structure will contain 2^m areas, where m is the number of numerical features. Fig. 1 shows a grid structure in a two-dimensional space. In this figure, the dashed lines show the cut points of each dimension, which together created a grid structure and led to the production of four areas.

The location of each cut point can be any point in the allowed range of the desired feature. Our goal is to use optimization algorithms to change the location of these cut points in such a way that the areas in the grid have the highest purity. The value of the purity of the whole grid is obtained from the equation (6).

$$\text{purity} = \sum_{i}^{\text{all areas}} \frac{\text{in area}_i; \text{number of instances with label c that have max frequency}}{\text{number of all instances}} \tag{6}$$

Here, the number of all instances refers to the total number of training instances across all areas. Label *c* represents the class label to which the majority of instances in area *i* belong.

In performing the calculations that depend on the information of the areas, including the cost function and the penalty term that will be presented in this section, only the information of the areas that have covered at least one training instance is used. From now on, for better expression, we call them active areas.

The algorithm begins with a preprocessing stage. The purpose of this stage is to reduce the search space and create suitable candidates for cut points. Subsequently, initial solutions are generated. In the next step, the employee bee and onlooker bee phases of the ABC algorithm are employed to modify the solutions and potentially improve them. If a solution is improved compared to its previous state, the change is retained; otherwise, the solution is restored to its previous state. In each iteration of the algorithm, the best solution found so far is updated. The process of modifying and improving solutions continues until the termination condition is met.

Once the search for the best solution concludes, the rule extraction phase begins. Initially, active areas are extracted from the best solution. Subsequently, in the post-processing stage, these areas are combined to form more comprehensive areas. Finally, the structure of the rules is derived from these areas.

In the remainder of this section, we will delve deeper into the various components of the proposed algorithm.

4.2. Pre-processing step

MCRM uses a meta-heuristic optimization algorithm to find the appropriate location of feature cut point. cut points can take any position within the allowed range of corresponding attribute values. On the other hand, for a continuous feature, there are infinite points between two adjacent instances that can be candidates for the cut point, while all these points classify the samples in the same way based on the desired feature and do not differ from each other for the optimization algorithm. Therefore, between any two adjacent instances, we select their average, representing all points, as a candidate cut point. In addition to the extracted candidate cut points, for each feature, we add two boundary cut points to the beginning and end of the cut points so that all the training samples are between these two points. Boundary cut points are used in the process of removing less important features. The extracted cut points for each feature are stored in an independent array. So if we have m features, we will have m arrays with different lengths. Doing the pre-processing stage has two advantages; First, it guarantees that the selected boundaries between areas (cut points) have the greatest

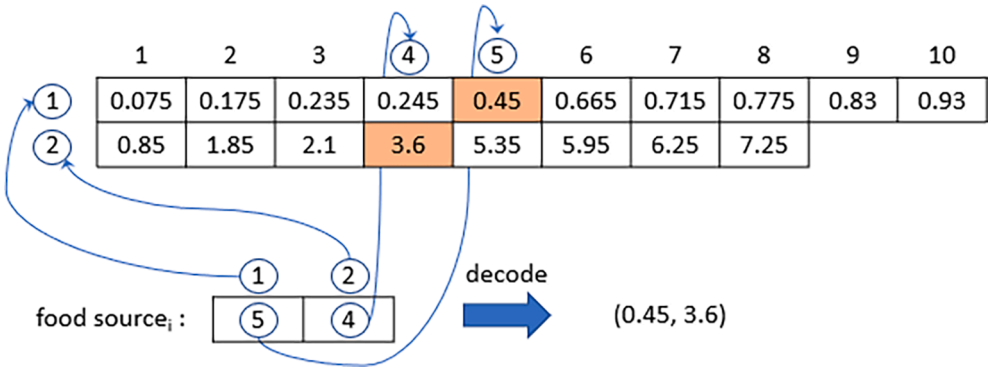

Fig. 2. Extracting the cut point of each feature from a food source array and the cut points table.

distance from the closest examples of each area. Secondly, the search space for the optimization algorithm is reduced.

4.3. Modified ABC algorithm

4.3.1. Problem representation

Each individual in the ABC algorithm is an array of positive integers of length m. Each element of the array corresponds to one of the features of the problem space and stores the index of one of the candidate cut points corresponding to the desired feature. For example, suppose that for a two-dimensional problem, the candidate cut points are extracted in the preprocessing step in a table as shown in Fig. 2. Each row of the table stores the candidate cut points of a feature. The candidate cut points of each feature have their unique indexes that determine their order. Therefore, to access the value of a specific candidate cut point, two components are needed; The row number of the table, which corresponds to the feature number, and the number of one of the elements of the desired row, which corresponds to the index of a candidate cut point. In the introduced structure, we need m cut points to create the classifier. Therefore, we consider each individual (food source) in the ABC algorithm, which represents a classifier, as an array of length m; Each of its elements corresponds to one of the features (one of the rows of the table) and it stores the index of one of the candidate cut points of the desired feature. When evaluating each individual in the population of solutions, which is equivalent to a classifier, it is necessary to form the structure of the classifier implicitly. Therefore, from the information of the array related to the desired individual, the cut points needed to create the classifier can be easily extracted from the table of cut points as shown in Fig. 2.

4.3.2. Initialization

Each element of the food source array is initialized according to (7).

$$\text{foodSource}_i[j] = \text{randInt}(1, T_j) \tag{7}$$

Here, assuming that the index numbering of the elements in the array starts from 1, the location of the cut point of feature j in the table of cut points, for food source *i*, is determined by a random integer between the number 1 and *Tj*. Where *Tj* is the number of candidate cut points in the j^{th} row of the cut points table.

4.3.3. Improve solutions

The local search equation is modified according to (8) to improve each solution.

$$X_i^{*j_{rand}} = X_i^{j_{rand}} + z \times randInt\,(0, \left|X_i^{j_{rand}} - X_k^{j_{rand}}\right|), \qquad z = -1 \text{ or } 1 \tag{8}$$

where $j_{rand} \in \{1, 2, \cdots, D\}$ is the index of one of the parameters of the problem space that is chosen randomly. $X_i^{j_{rand}}$ represents the current value and $X_i^{*j_{rand}}$ represents the new value of the selected parameter for solution *i*. $X_k^{j_{rand}}$ is the value of the selected parameter for the solution *k* so that $i \neq k$. And *z* takes one of the values -1 or 1 with equal probability. The *randInt* function generates a random integer between the difference of the parameter values of solutions *i* and *k*, which is the amount of movement of solution *i* on feature *j*.

4.3.4. Evaluating food sources (solutions)

For each food source, based on its cut points, a cost value is calculated that determines its fitness value. When comparing food sources, lower cost means higher quality. In the ABC algorithm, each food source is a potential solution and the task of the algorithm is to find a solution with minimum cost. In the rest of this section, we describe how to calculate the cost of each food source.

I. Create identifiers for areas

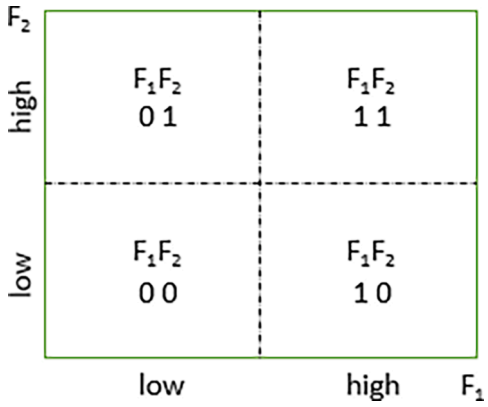

Fig. 3. Create identifiers for areas.

Instances	Cut points = (0.45, 3.6)	(Area identifier, Instance class)
(0.12, 6.1, A)	0.12 < 0.45 & 6.1 > 3.6	(01, A)
(0.24, 5.8, A)	0.24 < 0.45 & 5.8 > 3.6	(01, A)
(0.23, 4.9, A)	0.23 < 0.45 & 4.9 > 3.6	(01, A)
(0.25, 1.9, B)	0.25 < 0.45 & 1.9 < 3.6	(00, B)
(0.65, 2.3, A)	0.65 > 0.45 & 2.3 < 3.6	(10, A)
(0.80, 1.8, A)	0.80 > 0.45 & 1.8 < 3.6	(10, A)
(0.75, 6.4, A)	0.75 > 0.45 & 6.4 > 3.6	(11, A)
(0.68, 5.8, B)	0.68 > 0.45 & 5.8 > 3.6	(11, B)
(0.75, 4.9, B)	0.75 > 0.45 & 4.9 > 3.6	(11, B)
(0.86, 5.8, B)	0.86 > 0.45 & 5.8 > 3.6	(11, B)

Fig. 4. Determining areas that cover training instances.

To evaluate a food source, we first extract the cut points related to it according to Fig. 2 from the table of cut points. Each feature is divided into two intervals based on its cut point. We assign code 0 to the range of values smaller than the cut point and code 1 to the range of values greater than the cut point. By dividing the problem space based on the cut points of the desired food source, 2^m areas are formed on the problem space. Each area in the problem space can be represented by a binary code string of length m. So that one bit is considered for each feature; If an area is located before the cut point of a feature, the bit corresponding to that feature will be 0, otherwise, if the area is located after the cut point, the bit corresponding to the feature will be 1. Fig. 3 shows a two-dimensional space with the cut point of each feature marked with a dashed line. The cut points have resulted in the generation of 4 areas, which are represented by a binary code string of length 2, based on the provided description.

II. Labeling areas and classifying data

Based on the extracted cut points, for each training instance, we extract the area code that covers it. To do this, we compare the value of each feature of the training instance with the cut point corresponding to that feature; If it is less than the value of the cut point, we set the bit corresponding to the desired feature with 0, otherwise its value will be 1.

After assigning the codes of the areas to the instances, it is clear how to distribute the instances in each area. Now, based on the distribution of instances, the label of the active areas is determined. The label of an area corresponds to the label of the class whose instances are the most frequent in the desired area. For example, suppose that in a two-dimensional space, the cut points extracted from a food source (solution) are in ordered pairs of (0.45, 3.6); So that 0.45 is the cut point of the first feature and 3.6 is the cut point of the second feature. Also, suppose we have ten training instances with the feature values given in the first column of Fig. 4. In this case, the identifiers of the areas that cover each of the instances are obtained in the form of Fig. 4, according to the explanations provided. Here, each instance is specified by an ordered triple whose elements, from left to right, are the value of

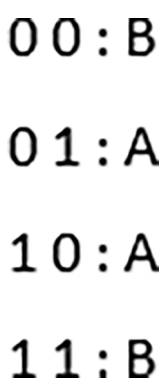

Fig. 5. Determining the labels of areas based on the distribution of training instances in areas.

attribute1, the value of attribute2, and the class label of the instance. In this figure, the ordered pair (01, A), i.e., an instance belonging to class A, is placed in an area with identifier 01.
Since the number of features is 2, there will be a maximum of 4 active areas. According to Fig. 4, one instance of class A and three instances of class B are placed in area 11, so the label of this area is determined based on the class with the highest frequency, i.e. class B. Similarly, the label of other areas is determined according to Fig. 5. In this example, each of the areas has covered at least one instance, so all four areas are active and the information of all of them will be used in the cost calculation.

III. Cost calculations

After determining the label of the active areas, the purity of each active area is calculated based on (9).

$$\text{purity}_i = \frac{\text{NC}_i}{\text{N}_i} \tag{9}$$

where NC_i is the number of instances in area i whose label is the same as the label of area i. Also, N_i is the total number of instances in area i.
Purity value of all areas according to (10) is obtained from the sum of weighted purity of active areas.

$$purity = \sum_{i}^{Areas} \frac{N_i}{N} \times purity_i = \frac{1}{N} \sum_{i}^{Areas} NC_i \tag{10}$$

Here, N is the total number of training instances.
The purity value of all areas always has a value between 0 and 1, so the cost of each food source can be obtained from equation (11).

$$\text{cost} = 1 - \text{purity} \tag{11}$$

IV. Modification of the cost function to solve the overfitting problem

In datasets with a low number of instances and a high number of features, the algorithm leads the grid structure to overfit. So that by placing each instance in a separate area, the purity of each active area will reach 1 and then the cost of the food source will be 0. To prevent overfitting of the classifier, we define a penalty term according to (12) to minimize the number of active areas.

$$\text{penalty} = \frac{\text{A}}{\text{MA}}, \ \text{MA} = \begin{cases} 2^{\text{m}}, \text{ if } 2^{\text{m}} < \text{N} \\ \text{N}, \text{ otherwise} \end{cases} \tag{12}$$

where, A is the number of active areas and MA is the maximum possible number of active areas, which can have two values. If the number of all areas, including active and inactive areas, obtained from the equation 2^m is less than the number of training instances specified by N, we can imagine a situation where at least one instance is placed in each area. In this case, all available areas will be active areas. Otherwise, if the number of training instances is less than the number of areas, we can imagine a situation where there is at most one training instance in each area. In this case, the number of active areas is equal to the number of training instances. By adding the penalty term to the cost function, equation (11) is modified as (13).

$$cost = (1 - purity) + \alpha \times penalty \tag{13}$$

Here, the coefficient α is a real number and determines the influence of the penalty term on the cost function.

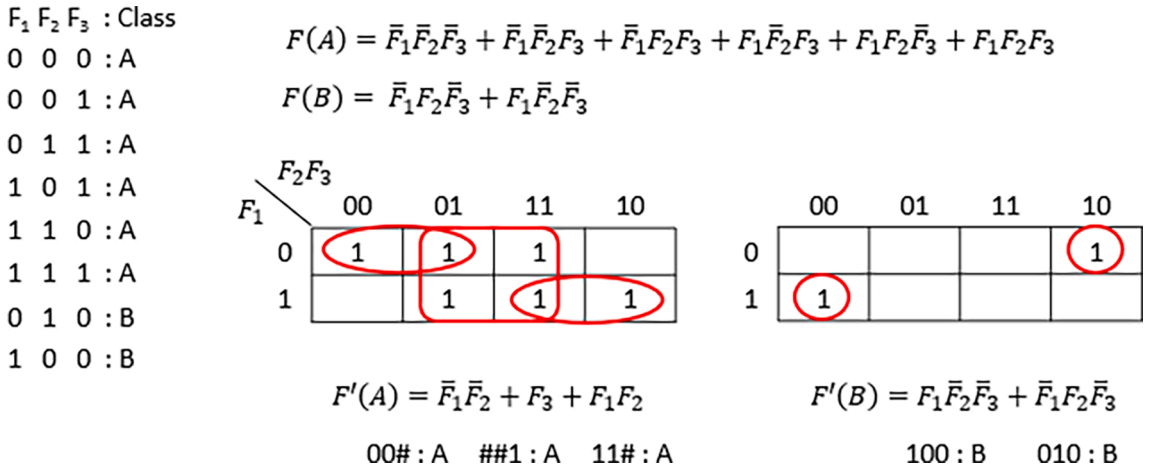

Fig. 6. Pruning rules with boolean function simplification technique.

4.3.5. Recruiting onlooker bees

To recruit onlooker bees, first, the fitness value of food sources is calculated as (14).

$$\text{fitness}_i = \text{maxCost} - \text{cost}_i + \text{eps} \tag{14}$$

where, *maxCost* is the highest cost among the food sources and $cost_i$ is the cost of the i^{th} food source. *eps* is a small value that prevents the probability of choosing the worst food source from going to zero.

Now the probability of choosing each food source by the onlooker bees is calculated according to (15).

$$p_i = \frac{\text{fitness}_i}{\sum_j^J \text{fitness}_j} \tag{15}$$

where, *J* is the number of food sources. According to this equation, food sources with higher fitness are more likely to be chosen.

4.4. Extracting the rules

MCRM uses (1) to display its rules. Each rule can be equivalent to an active area or a set of active areas. If we consider each active area as a rule, the antecedent part of the rule will contain m′ conditions, where m′ is the number of selected features. Each condition corresponds to a selected feature and is expressed based on the cut point of that feature along with one of the comparison operators $<$, $>$, $\leq$ or $\geq$. If an area is placed before the cut point based on the feature F_i, the condition related to that feature is expressed as (16).

$$\text{condition}_i = F_i < CP_i \tag{16}$$

where, CP_i is the cut point of feature *i*. According to (16), if the value of feature *i* for a new instance is lower than the cut point of that feature, then the i^{th} condition of the rule is satisfied. The set of these conditions, which represent the space covered by an active area, form the antecedent part of the rule corresponding to that area. Since each instance must satisfy all the conditions of the antecedent part of a rule to be covered by it; The antecedent part of the rule is expressed as a conjunction of conditions. The consequent part of the rule takes the area label from which conditions were derived.

To enhance rule interpretability and reduce redundancy, a rule simplification process can be applied. For this purpose, each rule can be considered as a set of areas. Let us consider both adjacent areas that are the same based on all features and differ only in one feature as neighboring areas, and call the cut point of the feature that differentiates them as the neighborhood boundary. In this case, we can merge the neighboring areas that have the same class label and ignore the effect of the feature related to the neighborhood boundary that differentiates them. If there is a neighborhood condition between the more general areas, they are merged. This scenario continues until there is no neighborhood condition between any two areas or the neighboring areas do not have the same class label. When we label features as don't care in area merging, it means that no conditions for don't care features are considered when creating rules for that area. Therefore, according to this rule, all values are acceptable for don't care features.

The order of integration of areas is important; If two areas are randomly selected for merging, the number of final areas is not necessarily minimized. To integrate the areas optimally, for each class, we create a boolean function in the form of CSP (Canonical Sum Of Products); That is, a function written in the form of the sum of several minterms (if all m boolean variables are combined with the AND operator, 2^m states are created, each of these states is called a minterm). As we said before, the identifier of each area is expressed by a binary code string with the length of the number of selected features, where one bit is considered for each feature. If we consider each feature as a boolean variable in the identifier of the areas, then the identifier of each area is a sequence of boolean literals and is equivalent to a minterm. Now we can simplify each of the boolean functions related to a specific class with one of the boolean function

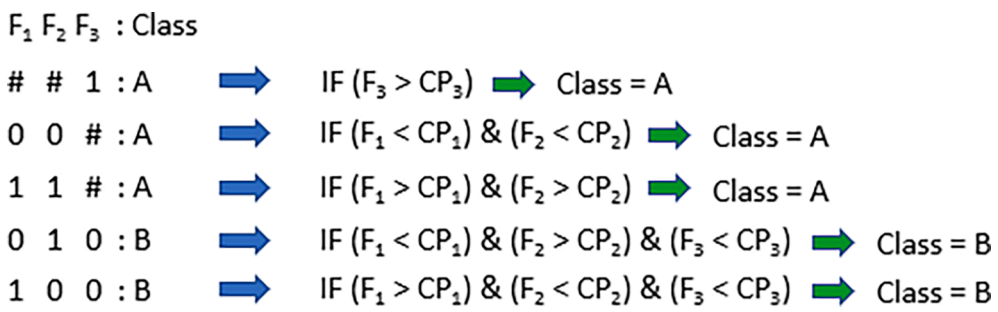

Fig. 7. Extracting the rules from final cut points and identifier of active areas.

simplification methods. By doing this, the minterms (initial areas) are merged and the simplified boolean functions are obtained in the form of SOP (Sum Of Products). Finally, each product term in simplified boolean functions will be equivalent to an area. If there is no literal for a boolean variable in a product term, we consider that variable as don't care and we use the # sign instead of its literal in the binary form of the product term. For example, consider Fig. 6. In this example, the training instances are in two classes A and B. Therefore, two boolean functions F(A) and F(B) are created from the areas extracted by the algorithm. Here, for simpler representation, the Karnaugh map technique is used to simplify these functions. Finally, the functions F′(A) and F′(B) are the simplified form of the functions F(A) and F(B), and each product term in them is equivalent to an area.

According to Fig. 7, the rules are extracted based on the identifier of the final active areas. In this figure, CP_i defines the cut point of the i^{th} feature. As stated before, the value of bit 1 for each attribute in the identifier of an area means that the value of the desired attribute in all instances covered by this area is greater than the cut point corresponding to that attribute. Also, the value of bit 0 for an attribute means that the value of this attribute is lower than its corresponding cut point in all instances covered by the desired area. Based on the stated definitions, it is easy to derive the rules by having the cut points of the features and the identifier of the areas. In this figure, don't care features are marked with #.

4.5. Classify new data

For classifying new instances, each active area can be considered as a rule. Therefore, the feature values of the new instance are compared to the cut points to find the area that covers it. Then the area that covers the instance determines its label. If an instance is not covered by any of the active areas, it will be labeled with the area that most closely matches it. If the rules are extracted from the areas; Among the classification rules, a rule determines the label of the new instance if the new instance satisfies all the conditions of the antecedent part of it. Otherwise, if the new instance is not covered by any rule, it will be labeled by a rule that the new instance satisfies more conditions of the antecedent part of it. The described method can be extended to the case where the rules are obtained from the integrated areas.

4.6. Efficient management and storage of areas

In MCRM, each solution is a potential classifier and according to the structure stated at the beginning of this section for classifiers, it includes some areas. To evaluate each solution, we need the information of its areas, and this information is unique in each iteration of the algorithm. As mentioned earlier, the number of areas for each solution is a function of the number of selected features at the current moment. For example, at the beginning of the algorithm, when there are m selected features, the number of areas is 2^m. The problem is that with the increase in the number of features, the number of areas grows exponentially until it becomes impossible to store and process the information of the areas. But the interesting thing to note is that most of the time the space of the areas is very sparse; This means that in many of these areas, no training instances are placed, so storing and processing their information is not beneficial for us. In this section, with the help of the hash data structure, we provide a method for storing the information of the areas so that the storage and retrieval of the information of the areas can be done more efficiently.

If the training instances are in C different classes, we need an array of positive integers of length C to store the information of each area. Each element of this array corresponds to a class and stores the number of instances covered by the area in the desired class. In the evaluation of each solution, the algorithm examines a training instance in each step and finds the identifier of the area covering this instance based on the information of its features and the cut points of the desired solution. Then, in the array corresponding to the area, it adds one unit to the current value of the element corresponding to the instance class.

To access the array of each area through its identifier, we need a dictionary. We use the hash table to implement the dictionary. In this method, we consider the hash table as implementable size t and a hash function receives the key of the values as input and maps the values to one of the t elements of the hash table based on their keys. More than one key may be mapped to a particular element, which is called a collision problem. One of the collision avoidance methods used in this research is the chaining method. In the chaining method, each element of the hash table points to a linked list instead of holding a value. Based on this, each key mapped to element i of the hash table by the hash function, if it is not present in the linked list of the i^{th} element, it is added to the beginning of the linked list along with its value. In this case, the insertion operation is performed in O(1) time. However, the search time is proportional to the

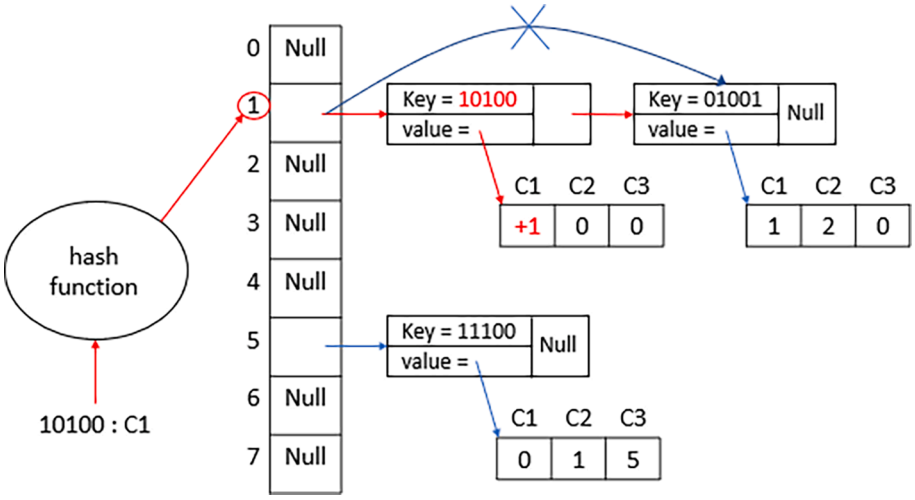

Fig. 8. Data storage of an active area.

length of the list. It can be shown that if the hash function is simple uniform, it means that every key is mapped to every element of the table with equal probability, and assuming that the hash function performs its calculations in O(1), the average search time will be θ(1 + α) [43]. Here, α is the load factor for the hash table obtained from the equation $\alpha = n / t$, where t is the size of the table and n is the number of values stored in the table. The load factor is the average length of the hash table lists.

As mentioned, in the MCRM algorithm, the hash table is used to efficiently store the information of the areas. Each node added to the hash table lists stores the identifier of an area and a pointer to the information array for that area. Initially, all hash table elements point to an empty list (they have null value). Then the following steps are performed to store the information of the areas.

1. Repeat steps 2 to 5 for each training instance.
2. Based on the cut points and the instance feature values, identify the area that covers the instance.
3. Using the hash function, map the identifier of the desired area to an element of the hash table.
4. In the element list, find the node where the identifier of the desired area is stored. If there is no node already created in the list of this element to hold the area identifier, create a new node and store the area identifier in it. Also, create an array of length C to store area information and store a pointer to this array in the node. Then add the new node to the beginning of the list.
5. In the area information array, add one unit to the value of the element corresponding to the instance class.

According to this method, only the information of the active areas is kept. In Fig. 8, the data storage path of a new area is highlighted in red. In this figure, the new instance is covered by the 10100 area, which has not yet created a node for this area.

4.7. Discretization of feature values

MCRM uses discretization process to learn simple yet accurate rules. In fact, it learns the rules by performing the discretization operation. In this algorithm, discretization is of multivariate type. The discretization process is not performed as a separate step before the algorithm execution but is part of the rule learning process. Unlike univariate methods that discretize the values of each feature independently from other features, in MCRM correlation information between features is used to find more appropriate cut points. In this method, only one cut point is considered for each feature, so the values of each feature are divided into two categories based on its cut point. Probably, the instances cannot be separated based on one feature, with only one cut point, but this method categorizes the instances in the m-dimensional space (m is the number of features), so all the cut points together determine the accuracy of the discretization method. As stated in the preprocessing section, many candidate cut points are considered for each feature. The algorithm has the task of choosing only one of the candidate cut points of each feature in such a way that the instances are best classified in the m-dimensional space of the problem.

In [44], three different aspects for evaluating discretization methods are proposed, which include the following:

1. Simplicity
2. Consistency
3. Accuracy

The smaller the number of cut points and then the number of intervals for the continuous values of the feature, the simpler the discretization method and the higher its quality. However, the problem is that with the increase in simplicity, the possibility of inconsistency increases. In the definition of inconsistency, it can be said that two instances are inconsistent if their feature values are

the same but their class labels are different. Obviously, less inconsistency is better; So, the discretization method's ability to minimize inconsistency while producing minimum cut points defines its strength. MCRM uses only one cut point to discretize the values of each feature, so it has a high level of simplicity. Also, to reduce the amount of inconsistency due to the increase in simplicity, it performs the discretization operation in the m-dimensional space. On the other hand, the output of the discretization process is used in the classification process, so the accuracy of the classifier on the discretized data can be considered an accuracy evaluation criterion.

4.8. Feature selection

The proposed grid structure for data classification allows us to remove irrelevant and less important features from the feature set. This work has several advantages; First, it increases the speed of information processing during the training of the classification model. Second, it increases the readability and interpretability of the final extracted rules. Thirdly, it helps to avoid overfitting the classification model.

Due to the introduced structure, the feature selection operation is easily and automatically performed by the optimization algorithm. In the pre-processing section, we added two boundary cut points to the set of cut points of each feature, which is one of their applications in feature selection operations. When the algorithm selects a boundary cut point for a feature, it means that the algorithm deems that feature irrelevant for instance separation and prefers that the instances are all placed in the same category based on this feature. For example, choosing the boundary cut point that is at the end of the cut point array of a feature causes the value of this feature to be smaller than the cut point in all training instances. Therefore, when assigning them to an area, the bit corresponding to the desired feature in the area identifier will be 0 for all instances. It means that the desired feature does not effect on the separation of instances. Therefore, these features can be ignored in some calculations.

The most important factor that makes the algorithm tend to choose boundary cut points for features is the penalty term that was added to the cost function in the evaluation of solutions (food resources in the ABC algorithm) to avoid overfitting. If the coefficient of the penalty term is too small, only features that have a negative effect or a very small positive effect on the separation of instances will be removed. And if the coefficient of the penalty term is high, only the basic and very effective features in the separation of instances will remain, to the point that if the coefficient is too high, the effect of the first term in the equation (13), which has the role of increasing accuracy, becomes very insignificant. Therefore, the algorithm prefers to ignore accuracy and remove all features, having only one active area containing all instances to minimize the cost.

4.9. Time complexity analysis

The time complexity of the MCRM algorithm mainly depends on the cost function structure and the iterations of the ABC algorithm. Let K be the maximum number of iterations of the ABC algorithm, D the dimensions of the problem space (number of features), N the number of training instances, and Ne the number of employee bees. In the initialization stage, the time complexity of generating initial solutions and calculating their cost is $O(Ne \times (D + f(N, D)))$. Here, $f(N, D)$ is the time complexity of the cost function, which depends on the number of training instances and the dimensions of the problem, which we will discuss later.

In the iterative process of the algorithm, in the employee bees phase, the time complexity of updating the food sources is $O(K \times Ne \times f(N, D))$. In the onlooker bee phase, each bee must first choose a food source based on its fitness. The time complexity of doing this is $O(K \times Ne)$. Then the onlooker bees repeat the steps of the employee bees. The time complexity of the scout bee phase is $O(K \times Ne)$. Therefore, it can be said that in the iterative process of the algorithm, the time complexity is $O(K \times Ne \times (2f(N, D) + 2))$.

The cost function, independent of the steps of the ABC algorithm, is defined differently for each problem and has a great impact on the execution time of the algorithm. In the cost function of the MCRM algorithm, first, the information related to the distribution of training instances in the areas is stored in a hash table structure. To do this, each training instance is mapped to an area. The time complexity of this will be $\theta(N \times (D + \alpha))$ assuming that the hash function is simple uniform and its calculation time depends on the number of features of the training instances. Here, α is the load factor of the hash table. After that, the cost of the food source is calculated based on the information of the distribution of training instances in the active areas. This work depends on the number of active areas and the number of classes of training instances. The number of active areas in the worst case is equal to the number of training instances, so the time complexity is $O(N \times C)$, where C is the number of classes of training instances.

The noteworthy point in the time complexity of the cost function is that the MCRM algorithm uses the potential of feature selection in reducing its computation cost. As we move away from the beginning of the algorithm execution, more features are marked as unimportant features and the number of selected features usually decreases significantly. Therefore, in calculations where the number of features of the training instances is effective, such as storing information in a hash table, instead of traversing all the features (D), only the selected features are considered. So the time complexity becomes much less compared to the expression that was stated earlier for the worst case. In addition, when calculating the cost of food sources, the number of active areas is usually much less than the number of training instances.

MCRM algorithm outside the framework of ABC algorithm has two pre-processing and post-processing stages. In the pre-processing stage, the values of each feature are sorted first, and then the average of both adjacent values is calculated. In the worst case, if each feature has a different value for each training instance; This is done with $O(D \times (N \log_2 N + N))$ time complexity. In the post-processing stage, the creation of boolean functions is done by separating the area identifiers into different classes, which depends on the number of final areas (rules) and can be done with linear time complexity. The time complexity of simplifying boolean functions depends on the algorithm used for this purpose.

Table 1
Details of selected datasets.

ID	Dataset name	Number of instances	Number of features	Number of instances with miss values	Number of classes
1	Acute Inflammations (urinary bladder) [47]	120	6	0	2
2	Ada [46]	4147	48	0	2
3	Algerian Forest Fires [47]	244	10	0	2
4	Breast Cancer Coimbra [47]	116	9	0	2
5	Breast Cancer Wisconsin (Diagnostic) [47]	569	30	0	2
6	Breast Cancer Wisconsin (Original) [47]	699	9	16	2
7	BHP flooding attack on OBS Network [47]	1075	21	0	4
8	Caesarian Section Classification [47]	80	5	0	2
9	Cardiotocography [47]	2126	21	0	3
10	Cervical Cancer Behavior Risk [47]	72	19	0	2
11	Contraceptive Method Choice [47]	1473	9	0	3
12	Dermatology [47]	366	34	8	6
13	Diabetes [46]	768	8	0	2
14	Diabetic Mellitus [46]	281	97	2	2
15	Divorce Predictors [47]	170	54	0	2
16	Haberman's Survival [47]	306	3	0	2
17	Heart failure clinical records [47]	299	12	0	2
18	Iris [47]	150	4	0	3
19	Jasmine [46]	2984	144	0	2
20	Madeline [46]	3140	259	0	2
21	Madelon [46]	2600	500	0	2
22	Mammographic Mass [47]	961	5	131	2
23	Parkinsons [47]	197	22	0	2
24	Raisin [47]	900	7	0	2
25	Somerville Happiness Survey [47]	143	6	0	2
26	Statlog (Australian Credit Approval) [47]	690	14	37	2
27	Statlog (Heart) [47]	270	13	0	2
28	Tokyo1 [46]	959	44	0	2
29	Triazines [46]	186	60	0	2
30	Wall-Following Robot Navigation [47]	5456	24	0	4
31	Website Phishing [47]	1353	9	0	3
32	Zoo [47]	101	16	0	7

5. Experimental results

To evaluate MCRM's performance, we compared it with established classification algorithms. We primarily focused on rule-based and decision tree algorithms, but for a more comprehensive assessment, we also included algorithms employing different learning approaches. Most comparison algorithms were implemented in the Weka data mining tool [45]. We compared MCRM with the results of Weka classifiers (version 3.9.2), following Weka's naming convention for the selected classifiers. Additionally, we incorporated CORER [39], a meta-heuristic rule learning algorithm, and RACER [28], a recent rule learning algorithm, for a broader comparison.

To compare MCRM with the aforementioned classifiers, we selected 32 diverse datasets from the OpenML website [46] and the UCI machine learning repository [47]. These datasets span various fields and exhibit a variety of instances, features, and classes. Since MCRM handles numerical, binary, and ordinal features, most datasets contain minimal or no nominal features. If present the limited number of nominal features, MCRM can remove them during rule learning.

Preprocessing was necessary for most datasets to ensure compatibility with both MCRM and Weka classifiers. This involved adding metadata, removing instances with missing values, eliminating irrelevant attributes like dates and identifiers, and converting categorical values denoted by Latin letters to integers. Table 1 details the processed datasets. To simplify result analysis, each dataset was assigned a unique identifier used in place of its name.

MCRM uses the ABC meta-heuristic optimization algorithm to find the cut points that lead to the generation of high-quality rules. This algorithm includes parameters for which the selection of suitable values has a great effect on the result of the optimization process. For this purpose, the termination condition for the ABC algorithm was set based on two parameters. The first parameter is the maximum number of consecutive iterations of the algorithm in which the best solution does not improve. The second parameter is the maximum number of iterations of the algorithm. Based on the complexity of the problem space, which is a function of the number of features and the variety of values of each feature; The maximum value of the first parameter was set to 10,000 and the second parameter to 20000. The number of population individuals (sum of employee bees and onlooker bees) was considered to be 200 (100 potential solutions) at most. Also, the maximum value of the limit parameter was determined to be 600. MCRM has two other adjustable parameters independent of the parameters related to the ABC algorithm. The first parameter helps the process of removing less important features. This parameter specifies the probability of checking the possibility of removing a feature from a solution after the desired solution does not improve in the normal flow of the ABC algorithm. In all experiments, we considered the value of this parameter to be 0.1. The second parameter is the coefficient of the penalty term in the cost function. The appropriate value for this parameter depends on several factors such as the number of instances, the number of features, and how the instances are distributed in the problem space. We set the value for this parameter between 0 and 2.5.

Table 2
The average accuracy obtained for the classifiers in 100 runs on each of the datasets.

	DecisionTable .[45]	JRip.[45]	OneR.[45]	PART.[45]	HoeffdingTree.[45]	J48.[45]	LMT.[45]	RandomForest.[45]	RandomTree.[45]	REPTree.[45]	NaiveBayes.[45]	Logistic.[45]	SMO.[45]	RACER.[28]	CORER.[39]	MCRM
1	99.83	99.25	76.42	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	99.25	99.08	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
2	84.23	83.98	79.52	83.86	77.42	84.85	**85.48**	85.14	79.79	84.37	83.90	84.81	84.06	82.16	80.30	84.84
3	96.30	97.12	97.45	96.33	94.42	96.34	97.58	97.49	97.00	97.12	94.39	96.77	94.38	**98.19**	97.45	97.86
4	62.97	71.16	63.94	74.80	62.44	73.71	72.21	74.34	66.11	65.86	62.78	73.39	64.24	72.05	**79.33**	77.44
5	93.64	93.68	88.86	94.48	93.06	93.74	97.40	96.33	92.82	93.36	93.29	94.92	**97.59**	95.96	91.67	96.38
6	95.34	95.93	92.11	95.65	96.19	95.67	96.49	**97.12**	94.95	94.83	96.28	96.65	96.90	95.91	95.05	96.15
7	99.87	99.81	90.94	99.96	69.75	99.91	88.29	**100.00**	**100.00**	97.32	71.77	84.75	78.34	70.52	76.23	99.97
8	51.63	55.75	43.75	56.88	58.88	62.50	57.63	54.50	53.13	52.38	59.75	54.50	57.88	55.37	54.63	**65.5**
9	90.93	92.85	82.37	93.33	80.44	92.97	93.81	**94.83**	91.59	92.49	82.24	89.18	89.36	90.17	88.78	91.34
10	80.80	81.11	75.50	85.14	91.61	83.57	89.64	88.71	82.52	78.93	90.32	91.70	90.45	**92.89**	85.911	91.52
11	53.10	52.70	47.15	49.08	49.61	51.49	53.39	51.07	46.92	52.42	50.59	50.51	48.72	46.27	47.66	**55.27**
12	86.93	87.18	50.40	94.05	97.43	93.57	97.32	96.93	86.63	90.65	**97.46**	96.85	96.09	92.58	31.974	96.29
13	74.90	74.83	71.54	73.48	75.59	74.74	76.96	76.58	70.20	74.31	75.77	**77.57**	76.85	67.65	72.37	76.84
14	**99.29**	**99.29**	**99.29**	98.38	95.77	99.21	97.99	94.84	79.78	99.07	90.65	83.45	91.08	97.45	67.89	**99.29**
15	96.41	96.18	96.29	96.29	97.82	96.59	97.18	97.65	96.24	95.76	97.65	97.65	**98.18**	97.82	83.29	97.71
16	71.22	72.79	71.38	71.21	73.73	70.59	73.02	67.57	65.91	72.26	74.58	74.35	73.34	73.53	**76.03**	75.66
17	82.30	82.07	**85.31**	78.14	76.28	81.13	82.69	83.96	76.88	82.17	76.17	82.65	82.59	80.78	77.98	83.37
18	92.80	94.13	92.33	94.87	95.40	95.00	96.20	95.13	93.87	94.00	95.40	**97.20**	95.67	69.07	72.33	94.40
19	79.24	80.12	76.27	73.86	74.87	75.97	79.26	**81.46**	72.96	79.21	74.86	78.40	79.01	74.74	47.87	80.97
20	71.82	79.89	58.49	66.54	60.04	73.93	76.30	73.59	60.55	78.04	60.09	56.68	56.84	62.61	48.93	**86.01**
21	71.27	77.40	56.63	61.81	59.34	69.73	73.23	64.68	56.26	75.97	59.31	55.50	55.86	65.44	47.41	**85.66**
22	82.65	83.30	82.59	82.05	83.05	82.34	**84.05**	79.96	77.76	82.54	83.13	83.81	82.51	81.46	82.02	83.44
23	82.07	87.48	87.29	83.97	74.30	84.64	85.82	90.82	86.15	85.57	69.56	85.12	87.35	88.09	87.19	**93.14**
24	84.59	84.89	85.09	85.40	83.61	85.36	85.83	85.89	80.72	85.13	83.59	85.64	**86.61**	86.08	86.28	85.25
25	64.40	63.18	**65.80**	57.40	56.00	60.33	61.82	58.42	54.77	58.63	57.69	57.00	58.96	57.40	58.14	65.59
26	84.67	85.33	85.51	84.70	84.74	85.35	85.26	**86.75**	78.72	84.94	77.26	85.28	84.91	82.67	82.48	85.84
27	77.70	79.41	70.41	76.67	84.37	76.56	82.74	81.81	73.74	75.63	84.63	83.74	83.81	80.63	77.19	**84.96**
28	91.09	91.91	88.43	90.84	89.87	91.46	92.30	**93.20**	90.33	91.99	90.55	92.44	91.83	92.57	89.25	92.71
29	71.80	75.25	58.57	**78.25**	53.27	76.73	77.78	76.27	73.44	72.44	70.06	70.18	73.51	74.75	68.55	78.13
30	90.85	98.84	75.67	**99.72**	53.89	99.61	97.61	99.56	97.91	99.36	52.61	70.36	71.35	95.02	94.97	97.34
31	85.07	90.39	81.75	89.34	85.53	**90.74**	89.93	89.46	86.96	88.79	84.24	85.99	86.18	88.17	86.63	87.62
32	86.01	88.95	72.64	92.14	88.09	91.75	93.93	**96.51**	94.90	88.98	95.23	95.01	92.73	92.85	86.13	94.82
Avg	82.37	84.25	76.55	83.08	78.65	84.06	84.97	84.71	79.96	83.24	79.24	81.63	81.47	81.28	75.68	**86.92**

Table 3
The average number of features selected from each dataset to create classification rules in a hundred times of running the MCRM algorithm.

ID	Actual number of features	Number of selected features	ID	Actual number of features	Number of selected features	ID	Actual number of features	Number of selected features
1	6	3.05	12	34	7.76	23	22	4.04
2	48	4	13	8	2	24	7	1.64
3	10	1.82	14	97	1	25	6	1.01
4	9	5.04	15	54	1.87	26	14	6.66
5	30	3.01	16	3	3	27	13	3.17
6	9	5	17	12	1.06	28	44	3.4
7	21	16.25	18	4	2.85	29	60	5.18
8	5	2.91	19	144	6.94	30	24	8.01
9	21	13.29	20	259	5.07	31	9	6
10	19	4.72	21	500	5	32	16	6.82
11	9	3	22	5	3			

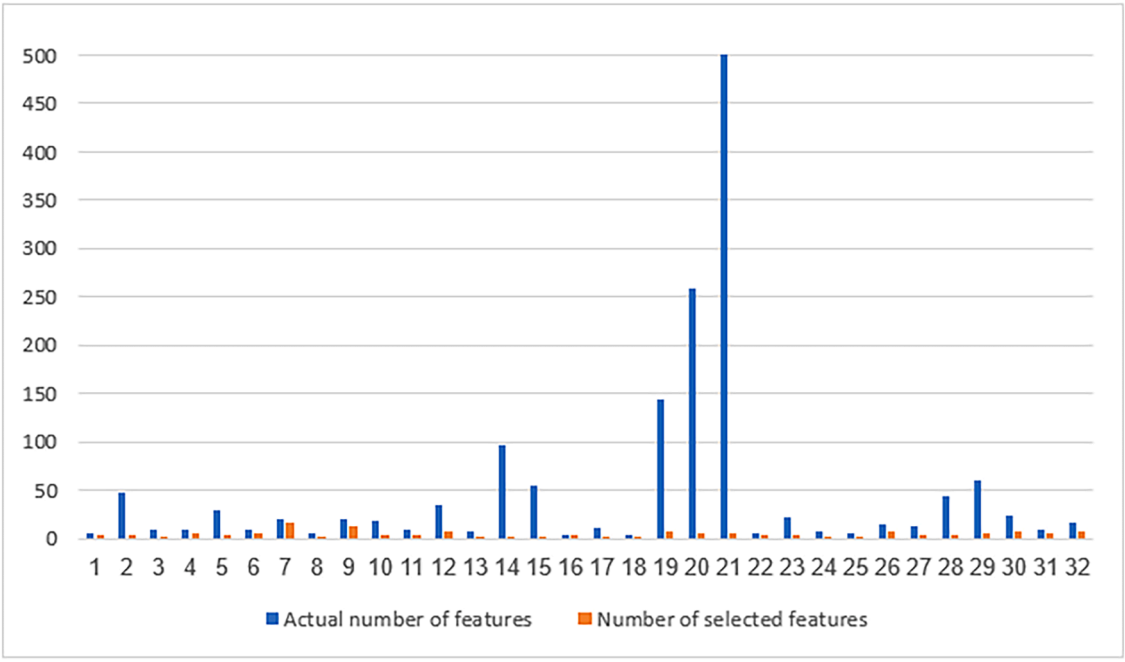

Fig. 9. Comparison of the number of a dataset features and the number of features selected by the proposed algorithm to create classification rules for that dataset.

Since the distribution of data in the training and testing sets affects the performance of classifiers, to make a fair comparison between classifiers, we have shuffled each dataset 10 times and created 10 versions of each dataset. Then, using the tenfold cross-validation method, we evaluated the performance of each classifier on each version of a dataset. In the tenfold cross-validation method, the instances of each dataset are divided into ten separate groups, ensuring that there are no common instances between any two groups. Then, the training process is executed ten times, and in each run, one of the ten groups of instances is separated as the test set, and the remaining nine groups are used as the training set to create the classifier model. Based on the provided explanation, to evaluate the performance of a classifier on a dataset, the classification process is performed one hundred times on the desired dataset.

Classification accuracy is one of the criteria that we use to evaluate the performance of our proposed method. The accuracy of a classifier is the ratio of correct predictions to the total number of predictions made by it, which can be calculated from equation (17).

$$\text{Accuracy} = \frac{CP}{P} \tag{17}$$

Where P is the number of predictions made (number of test instances) by the classifier and CP is the number of correct predictions by it.

To assess MCRM's accuracy performance, we compared its prediction accuracy with selected classifiers on the datasets listed in Table 1. Table 2 presents this comparison. Each column represents a classifier, and each row represents a dataset identified by its unique ID. The last row shows the average accuracy for each classifier across all datasets. Bold values in each row highlight the highest accuracy achieved on that dataset. As evident from Table 2, MCRM outperforms other classifiers on eight datasets and achieves a

Table 4
Comparison of the average number of rules extracted by the MCRM algorithm on each dataset with six other rule learning algorithms in 100 executions of each rule learning algorithm.

	DecisionTable [45]	JRip [45]	PART [45]	J48 [45]	RACER [28]	CORER [39]	MCRM
1	7	**3.01**	4	4	4	3.78	4.61
2	78.56	6.55	128.89	106.21	59.57	831.43	**5**
3	6.15	**2.77**	4.61	4.7	6.18	6.45	2.82
4	**3.43**	3.54	5.95	9.25	15.24	17.41	8
5	26.19	**4.98**	6.74	11.47	15.84	66.38	5.99
6	21.65	**5.48**	9.87	11.05	18.29	23.4	10.41
7	190.66	**14.64**	18.22	33.78	16.57	38.24	75.51
8	6.35	**2**	7.43	2.06	14.88	11.01	3.99
9	206.37	**13.86**	32.94	52.52	69.29	111.43	43.53
10	7.8	**2.78**	3.67	5.24	6.8	13.76	7.99
11	82.17	4.65	167.93	147.1	82.83	182.41	**4**
12	74.58	12.85	**8.3**	24.71	11.38	309.89	13.69
13	34.6	3.74	7.69	21.46	11.24	74.33	**3**
14	2.02	**2**	2.69	2.02	3.29	206.31	**2**
15	3.98	**2.63**	3.75	3.9	4.01	44.55	2.87
16	**1.44**	2.08	3.25	2.9	1.7	4.02	4.91
17	7.02	3.41	18.42	15.2	45.39	42.73	**2.06**
18	4.07	**3.7**	3.82	4.71	5.6	4	3.85
19	121.96	**8.45**	162.53	191.07	159.49	2685.57	17.12
20	138.8	**11.68**	79.58	190.92	163.79	2826	16.07
21	138.49	9.04	65.54	165.69	69.07	2340	16
22	35.02	**3.52**	14.59	14.26	42.24	52.57	4
23	18.06	**3.93**	6.39	10.02	15.66	29.16	5.31
24	28.23	3.96	5.42	7.62	12.23	25.58	**2.64**
25	**2.01**	2.15	9.32	11.32	15.62	17.09	**2.01**
26	33.51	**4.35**	31.89	22.88	61.96	17.97	8.95
27	22.49	**4.22**	18.3	21.76	38.07	46.98	6.17
28	43	4.83	12.64	15.94	27.72	138.93	**4.46**
29	15.26	**4.85**	15.01	14.45	25.91	69.92	6.31
30	698.37	14.93	**11.15**	21.25	56.33	144.2	26.62
31	104.11	**15.11**	44.93	56.18	70.64	96	20.6
32	14.54	**7.21**	7.62	8.31	8.8	9.99	11.03
Avg	68.06	**6.03**	28.85	37.94	36.24	327.86	10.99

Table 5
Comparison of the average number of rules extracted by the MCRM algorithm in each dataset, before and after the post-processing stage.

ID	Number of rules before merging	Number of rules after merging	ID	Number of rules before merging	Number of rules after merging	ID	Number of rules before merging	Number of rules after merging
1	6	4.61	12	24.38	13.69	23	9.22	5.31
2	12.04	5	13	4	3	24	3.28	2.64
3	3.64	2.82	14	2	2	25	2.02	2.01
4	14.27	8	15	3.74	2.87	26	20.08	8.95
5	7.99	5.99	16	7.98	4.91	27	8.17	6.17
6	24.04	10.41	17	2.12	2.06	28	6.77	4.46
7	122.22	75.51	18	4.7	3.85	29	7.15	6.31
8	5.85	3.99	19	48.91	17.12	30	76	26.62
9	90.20	43.53	20	32.16	16.07	31	62.98	20.6
10	11.27	7.99	21	32	16	32	14.26	11.03
11	8	4	22	8	4			

higher average accuracy overall.

As mentioned in the previous sections, MCRM can perform feature selection operations while learning rules. In Table 3, we have given the average number of features selected by MCRM after running the algorithm a hundred times on each dataset. Fig. 9 provides a visual comparison of the data presented in Table 3. Each dataset is represented by two bars: one indicating the original number of features and the other showing the number of features selected by the proposed algorithm. A clear disparity between these bars highlights the algorithm's effectiveness in reducing feature dimensionality for most datasets. In MCRM, the number of selected features is an upper bound for the length of the rules. When merging areas and extracting final rules, some selected features may be identified as don't care and ignored for a rule. Therefore, the number of conditions in the antecedent part of a rule can be less than or equal to the number of selected features.

In another experiment, we compared the average number of rules generated by MCRM with six other algorithms. The results are presented in Table 4. In determining the number of rules, as discussed earlier, each MCRM rule can contain a single area or merge multiple adjacent areas. Table 4 reveals that MCRM achieves the second-lowest average number of rules.

Table 6
Comparison of the average length of rules extracted by the MCRM algorithm in each dataset, before and after the post-processing stage.

ID	Length of rules before merging	Length of rules after merging	ID	Length of rules before merging	Length of rules after merging	ID	Length of rules before merging	Length of rules after merging
1	3.05	2.41	12	7.76	6.78	23	4.04	3.07
2	4	2.59	13	2	1.33	24	1.64	1.22
3	1.82	1.27	14	1	1	25	1.01	1
4	5.04	4.18	15	1.87	1.29	26	6.66	5.41
5	3.01	2.01	16	3	2.13	27	3.17	2.19
6	5	3.21	17	1.06	1.02	28	3.4	2.46
7	16.25	15.47	18	2.85	2.43	29	5.18	4.91
8	2.91	2.18	19	6.94	5.07	30	8.01	6.01
9	13.29	11.93	20	5.07	3.76	31	6	3.73
10	4.72	3.96	21	5	3.69	32	6.82	6.4
11	3	2	22	3	1.76			

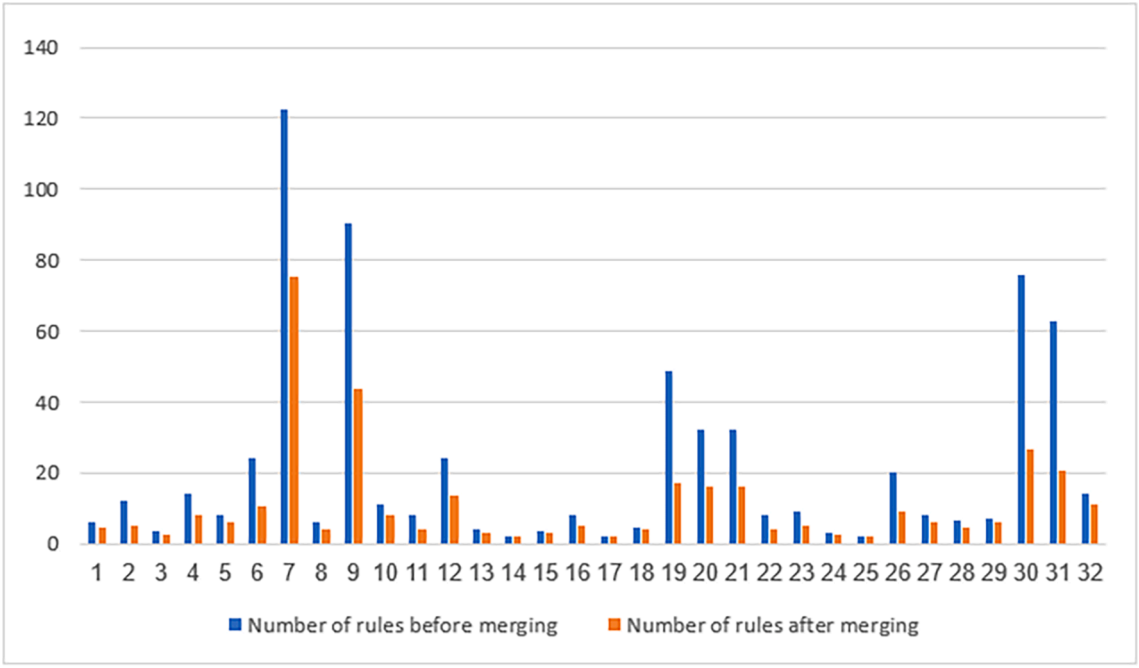

Fig. 10. Comparison of the average number of rules extracted by the MCRM algorithm in each dataset, before and after the post-processing stage.

Tables 5 and 6 demonstrate the effectiveness of the post-processing stage in streamlining the rule set. Table 5 reveals a significant reduction in the average number of rules extracted by the algorithm, highlighting the post-processing stage's impact. Similarly, Table 6 shows a notable decrease in average rule length after post-processing, indicating the stage's ability to merge and create more comprehensive rules. Figs. 10 and 11 provide visual representations of the data presented in Tables 5 and 6, respectively, enhancing data interpretation and analysis.

As mentioned in Section 3, we employed the metaheuristic optimization algorithm ABC to search the solution space and discover classification rules. There are three factors in the ABC algorithm that make it a strong candidate to contribute to the algorithm's objectives. First, the equations defined for the real-value space are relatively simple and with a little change, they can be adapted for the integer-value space. Most meta-heuristic optimization algorithms are introduced for problems with real-valued space. The more complex the search equations are, the more difficult it is to match them with the integer-value space of the proposed algorithm. For example, algorithms such as BRADO [48] and Evolutionary Strategy [49], which have relatively complex equations, seem unsuitable for searching the solution space of the proposed algorithm. The second factor is related to the search method. In ABC, in each iteration, changes are applied to one dimension of the problem space, which helps the feature selection process. In contrast, algorithms that apply changes to multiple dimensions in each iteration have the potential to destroy the effect of feature selection. The third factor is how to choose the new generation. In ABC, every member of the population is evaluated immediately after the change and compared with its previous state, and in case of improvement, it replaces the previous state. These conditions allow the evaluation of changes related to the feature selection operator to be separated from other changes related to the main equations, enabling better management of the feature selection operator. In most other algorithms, managing the feature selection operator is not easily possible. For example,

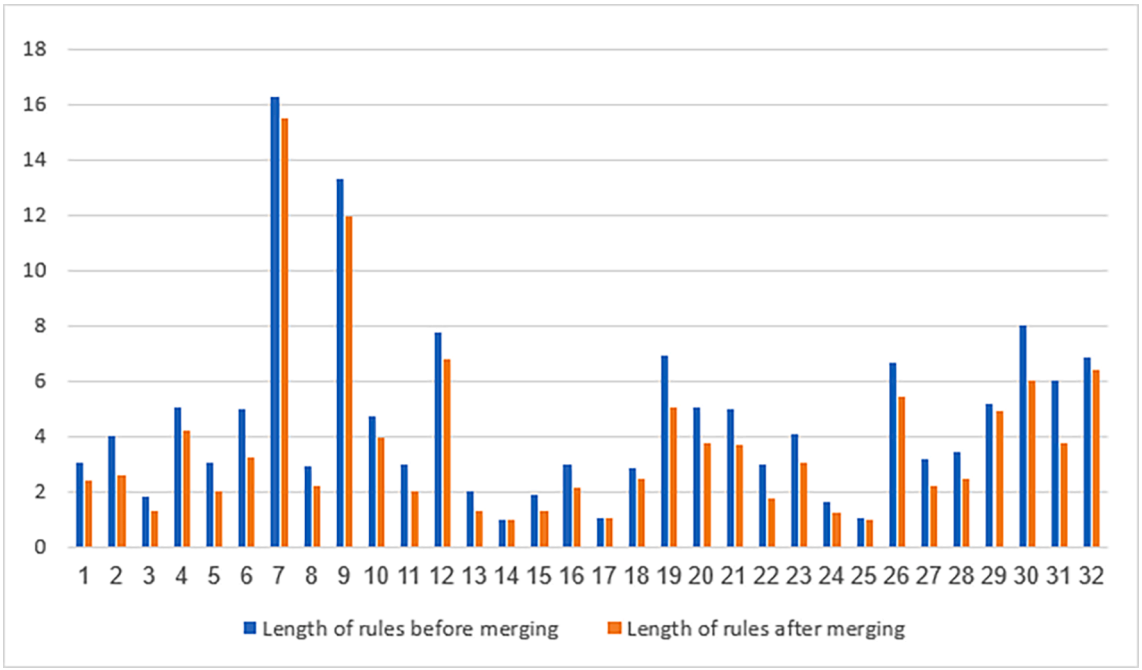

Fig. 11. Comparison of the average length of rules extracted by the MCRM algorithm in each dataset, before and after the post-processing stage.

Table 7
Comparing the accuracy obtained from the implementation of the proposed algorithm with the ABC algorithm and three other meta-heuristic optimization algorithms.

	MCRM-GA	MCRM-ICA	MCRM-SLPSO	MCRM-ABC
1	**100**	**100**	**100**	**100**
2	**84.88**	**84.88**	82.59	**84.88**
3	**98.38**	**98.38**	96.32	**98.38**
4	**79.17**	72.27	62.12	73.11
5	92.27	93.68	89.27	**96.66**
6	**97.21**	96.48	95.46	96.92
7	99.62	**100**	90.19	**100**
8	62.5	**63.75**	58.75	**63.75**
9	90.69	91.02	87.68	**91.3**
10	84.46	80.18	78.93	**90.18**
11	**55.12**	**55.12**	53.42	**55.12**
12	**96.1**	**96.1**	94.41	**96.1**
13	74.49	**76.7**	73.82	**76.7**
14	**99.29**	**99.29**	98.57	**99.29**
15	97.65	97.65	95.88	**98.24**
16	**75.44**	**75.44**	**75.44**	**75.44**
17	83.62	83.62	**84.62**	83.62
18	**94.67**	94	**94.67**	**94.67**
19	79.69	79.79	78.52	**80.63**
20	59.24	82.29	50.41	**86.11**
21	50.81	52.5	51.65	**85.73**
22	**83.61**	**83.61**	**83.61**	**83.61**
23	84.71	86.18	78.5	**94.39**
24	**85.56**	85	85.33	84.67
25	**65.71**	**65.71**	**65.71**	**65.71**
26	**85.36**	**85.36**	85.22	84.64
27	84.81	**85.19**	69.26	84.81
28	91.45	92.39	92.08	**92.7**
29	73.54	76.73	67.11	**78.89**
30	95.58	96.54	87.76	**97.38**
31	**87.51**	**87.51**	84.85	**87.51**
32	95	94	**96**	94
Avg	84.00	84.73	80.88	**86.72**

Table 8
Comparison of the number of rules obtained from the implementation of the proposed algorithm with the ABC algorithm and three other metaheuristic optimization algorithms.

	MCRM-GA	MCRM-ICA	MCRM-SLPSO	MCRM-ABC
1	4.8	5	5.1	**4.5**
2	**5**	**5**	13.1	**5**
3	3.1	**2.8**	3.8	**2.8**
4	**7.1**	7.5	12.2	7.4
5	**5.4**	**5.4**	40.7	6.1
6	10.6	10.3	**9.7**	10.6
7	90.8	86.7	88.5	**79.2**
8	3.9	3.9	**3.8**	3.9
9	44.3	42.6	81.4	**41.7**
10	8.3	8.6	23.7	**7.8**
11	**4**	**4**	4.1	**4**
12	**13.9**	**13.9**	29.7	**13.9**
13	**2.9**	3	**2.9**	3
14	**2**	**2**	2.4	**2**
15	2.9	2.9	**2.5**	2.9
16	**4.9**	**4.9**	**4.9**	**4.9**
17	**2.1**	**2.1**	2.9	**2.1**
18	**3.8**	**3.8**	**3.8**	**3.8**
19	18.3	18.9	18.9	**16.6**
20	273.9	19.4	1185.6	**16**
21	1891.9	68.7	2337.6	**16**
22	4	4	**3.7**	4
23	6.2	**4.8**	13.2	5.3
24	**2**	2.3	3	2.6
25	**2**	**2**	**2**	**2**
26	8.4	9	**4.3**	9.9
27	6.2	6.3	**5**	6.3
28	4.9	**4.6**	58.1	4.8
29	7	7	**4.3**	6.3
30	61.9	**26.1**	270.8	26.6
31	**20.9**	**20.9**	21.4	**20.9**
32	11.4	**11.3**	13	**11.3**
Avg	79.34	13.12	133.63	**11.07**

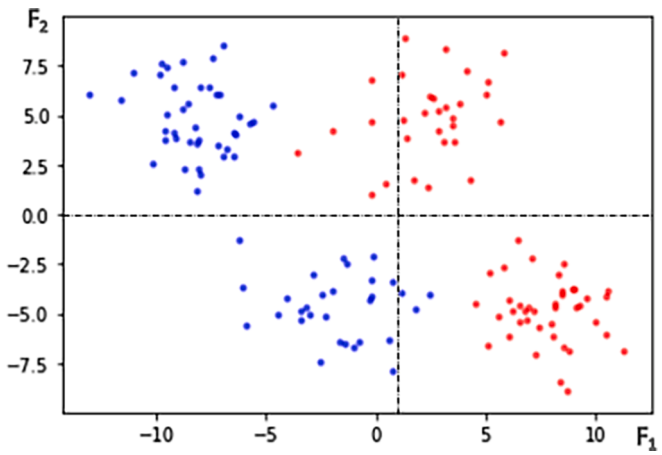

Fig. 12. Areas created on the problem space by the MCRM algorithm based on training instances.

creating a new generation in evolutionary algorithms is based on the selection operator. First, new members are produced, and then the new generation is produced by choosing between the members of the previous generation and the new members. Or in PSO [15], changes are applied to each member in each iteration, regardless of whether each member has improved compared to its previous state or not. This makes the algorithm unable to evaluate the positive or negative effect of the feature selection operator.

We implemented our proposed algorithm with three other metaheuristic optimization algorithms namely ICA [14], GA [49] and SLPSO [50]. The results regarding accuracy and the number of rules in 10 runs on each dataset are averaged and presented in Tables 7 and 8. The data in these tables suggest that the implementation of the proposed algorithm based on the ABC algorithm performed better on most datasets, especially datasets with a large number of features. This result can be attributed to the better compatibility of

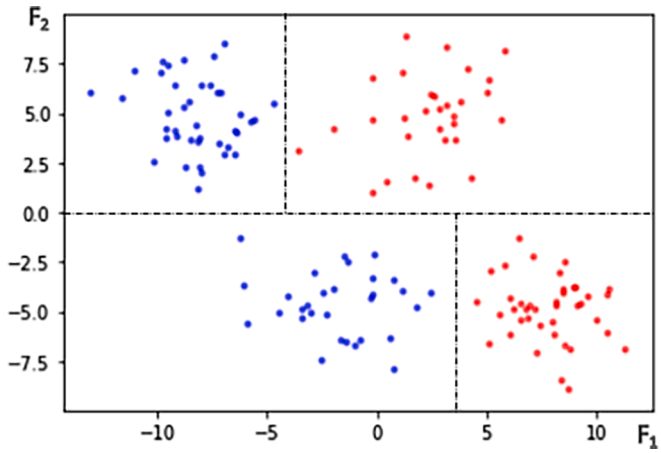

Fig. 13. Ideal areas on the problem space based on training instances.

the ABC algorithm with the proposed algorithm.

In MCRM, there is a relationship between the accuracy of the classifier and the number of rules, and sometimes focusing on one of them has a negative effect on the other. In this algorithm, the coefficient of the penalty term makes a tradeoff between accuracy and the number of rules. Another effective factor in classification accuracy and the number of rules is the limitation of MCRM in determining the boundaries of the rules. To better illustrate the effect of the mentioned limitation, consider Fig. 12. This figure shows the cut points selected by MCRM as dashed lines (Feature 2 and its horizontal line are removed by the algorithm due to ineffectiveness in data separation, but for better comparison, we have displayed them in the figure). Because MCRM is able to select only one cut point for a feature, some impurity is generated for the areas created by these cut points. If MCRM was able to select cut points as shown in Fig. 13, the classification accuracy would probably be increased in many cases compared to the current situation. Also, since MCRM takes advantage of the increase in the number of features to reduce the impurity of the areas; the flexibility of the cut points will probably lead to the selection of fewer features, followed by a reduction in the length of the rules and a reduction in the number of rules.

6. Conclusion

This paper introduces a meta-heuristic rule learning algorithm that incorporates feature selection during the rule learning process. The algorithm belongs to the Pittsburgh approach category, but it addresses some of the limitations associated with this approach. Each individual in the algorithm represents a classifier, and a penalty term is incorporated into the cost function equation to control the number of rules. This eliminates the need to pre-determine the number of rules before running the algorithm, as the algorithm dynamically adjusts the number of rules based on the penalty term coefficient. Additionally, the algorithm limits the length of individuals based on the number of features in the problem space, reducing the complexity of the solution space and facilitating the search for optimal solutions. The algorithm is designed to handle numerical, binary, and ordinal features without requiring a preprocessing step for numerical feature discretization. Instead, the discretization process is performed during the training of the classification model by considering the correlation between features. Overall, the algorithm integrates feature selection and numerical feature discretization seamlessly into the rule-learning process. Furthermore, the algorithm incorporates a special post-processing stage for pruning the classification rules. This, along with other factors such as feature selection and the specific structure of the learning rules, contributes to the generation of interpretable rules.

The results of the experiments conducted on 32 datasets from the OpenML website and the UCI machine learning repository show that MCRM performs well in terms of key evaluation indicators for rule-based classifiers, such as accuracy and the number of rules. When compared to other algorithms, MCRM has shown good performance. Additionally, this algorithm is capable of providing completely independent and interpretable rules.

When comparing MCRM with meta-heuristic rule learning algorithms and decision tree induction algorithms, several differences and limitations can be observed. Meta-heuristic rule learning algorithms often suffer from rule collision, which reduces interpretability and introduces uncertainty in determining instance labels. In contrast, decision tree induction algorithms generate independent rules, but they may not consider the combined information of multiple features, potentially leading to suboptimal performance.

MCRM overcomes the problems associated with both decision tree induction and meta-heuristic rule learning algorithms. However, it has a limitation not typically present in these algorithms. Specifically, MCRM assigns only one cut point for each feature, resulting in all rules following the same cut point for a specific feature. To mitigate this limitation and generate accurate rules, the algorithm selects additional features to increase the separation of instances and expand the search space.

Future work on MCRM could focus on eliminating this limitation to further enhance its performance. By addressing the constraint of having all rules follow the same cut point for a specific feature, the algorithm could potentially improve its rule generation capabilities.

CRediT authorship contribution statement

Mohammadreza Khosravi: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Conceptualization. **Alireza Basiri:** Visualization, Supervision, Project administration, Methodology, Conceptualization.

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

Data will be made available on request.

References

[1] J. Han, M. Kamber, J. Pei, Data mining concepts and techniques, 3rd ed., Morgan Kaufmann, 2012.
[2] J. Basiri, F. Taghiyareh, in: An Application of the CORER Classifier on Customer Churn Prediction, in, IEEE, Tehran, Iran, 2012, pp. 867–872.
[3] P.K. Giri, S.S. De, S. Dehuri, S.B. Cho, Biogeography based optimization for mining rules to assess credit risk, Intelligent Systems in Accounting, Finance and Management 28 (2021) 35–51.
[4] M. Mirsafaei, A. Basiri, in: Addressing Death from Heart Failure Using RACER Algorithm, IEEE, Tehran, Iran, 2022, pp. 636–640.
[5] G. Wang, Z. Deng, K.-S. Choi, Detection of epilepsy with Electroencephalogram using rule-based classifiers, Neurocomputing 228 (2017) 283–290.
[6] A. Chan, A.A. Freitas, in: A New Ant Colony Algorithm for Multi-Label Classification with Applications in Bioinfomatics, in, Association for Computing Machinery, Seattle, Washington, USA, 2006, pp. 27–34.
[7] W.E. Hadi, Q.A. Al-Radaideh, S. Alhawari, Integrating associative rule-based classification with Naïve Bayes for text classification, Appl. Soft Comput. 69 (2018) 344–356.
[8] R.M. Mohammad, F. Thabtah, L. McCluskey, Intelligent rule-based phishing websites classification, IET Inf. Secur. 8 (2014) 153–160.
[9] M. Celik, F. Koylu, D. Karaboga, Coabcminer: an algorithm for cooperative rule classification system based on artificial bee colony, Int. J. Artif. Intell. Tools 25 (2016) 1550028.
[10] R.J. Urbanowicz, J.H. Moore, Learning classifier systems: a complete introduction, review, and roadmap, Journal of Artificial Evolution and Applications 2009 (2009).
[11] D. Karaboga, An idea based on honey bee swarm for numerical optimization, in, Technical report-tr06, Erciyes university, engineering faculty, Computer Engineering Department (2005) 1–10.
[12] A. Faramarzi, M. Heidarinejad, B. Stephens, S. Mirjalili, Equilibrium optimizer: a novel optimization algorithm, Knowl.-Based Syst. 191 (2020) 105190.
[13] M. Dorigo, T. Stützle, The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances, in: F. Glover, G.A. Kochenberger (Eds.), Handbook of Metaheuristics, Springer, US, Boston, MA, 2003, pp. 250–285.
[14] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, in: 2007 IEEE congress on evolutionary computation, IEEE, Singapore, 2007, pp. 4661-4667.
[15] D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, in: 2007 IEEE swarm intelligence symposium, IEEE, Honolulu, HI, USA, 2007, pp. 120-127.
[16] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (2008) 702–713.
[17] J.C. Bansal, H. Sharma, S.S. Jadon, M. Clerc, Spider monkey optimization algorithm for numerical optimization, Memet. Comput. 6 (2014) 31–47.
[18] R. Storn, K. Price, Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (1997) 341–359.
[19] B.S. Yıldız, S. Kumar, N. Panagant, P. Mehta, S.M. Sait, A.R. Yildiz, N. Pholdee, S. Bureerat, S. Mirjalili, A novel hybrid arithmetic optimization algorithm for solving constrained optimization problems, Knowl.-Based Syst. 271 (2023) 110554.
[20] T. Kunakote, N. Sabangban, S. Kumar, G.G. Tejani, N. Panagant, N. Pholdee, S. Bureerat, A.R. Yildiz, Comparative performance of twelve metaheuristics for wind farm layout optimisation, Arch. Comput. Meth. Eng. (2022) 1–14.
[21] A.R. Yildiz, Optimization of cutting parameters in multi-pass turning using artificial bee colony-based approach, Inf. Sci. 220 (2013) 399–407.
[22] Z. Meng, B.S. Yıldız, G. Li, C. Zhong, S. Mirjalili, A.R. Yildiz, Application of state-of-the-art multiobjective metaheuristic algorithms in reliability-based design optimization: a comparative study, Struct. Multidiscip. Optim. 66 (2023) 191.
[23] P. Mehta, B.S. Yildiz, S.M. Sait, A.R. Yildiz, Hunger games search algorithm for global optimization of engineering design problems, Mater. Test. 64 (2022) 524–532.
[24] J.R. Quinlan, C4.5: programs for machine learning, Elsevier (2014).
[25] J. Fürnkranz, G. Widmer, Incremental Reduced Error Pruning, in: W.W. Cohen, H. Hirsh (Eds.), Machine Learning Proceedings 1994, Morgan Kaufmann, 1994, pp. 70–77.
[26] W.W. Cohen, Fast Effective Rule Induction, in: A. Prieditis, S. Russell (Eds.), Machine Learning Proceedings 1995, Morgan Kaufmann, 1995, pp. 115–123.
[27] E. Frank, I.H. Witten, Generating accurate rule sets without global optimization, in: Computer Science Working Papers, University of Waikato, Department of Computer Science, 1998.
[28] J. Basiri, F. Taghiyareh, H. Faili, RACER: accurate and efficient classification based on rule aggregation approach, Neural Comput. & Applic. 31 (2019) 895–908.
[29] I. De Falco, Differential evolution for automatic rule extraction from medical databases, Appl. Soft Comput. 13 (2013) 1265–1283.
[30] F.E. Otero, A.A. Freitas, C.G. Johnson, A new sequential covering strategy for inducing classification rules with ant colony algorithms, IEEE Trans. Evol. Comput. 17 (2013) 64–76.
[31] E. Farhana, S. Heber, Biogeography-based rule mining for classification, in: Proceedings of the Genetic and Evolutionary Computation Conference, Association for Computing Machinery, Berlin, Germany, 2017, pp. 417–424.
[32] R.S. Parpinelli, H.S. Lopes, A.A. Freitas, Data mining with an ant colony optimization algorithm, IEEE Trans. Evol. Comput. 6 (2002) 321–332.
[33] F.E.B. Otero, A.A. Freitas, C.G. Johnson, cAnt-Miner: An Ant Colony Classification Algorithm to Cope with Continuous Attributes, in: M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, A.F.T. Winfield (Eds.), Ant Colony Optimization and Swarm Intelligence, Springer, Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 48–59.
[34] H.N.K. Al-Behadili, K.R. Ku-Mahamud, R. Sagban, Hybrid ant colony optimization and genetic algorithm for rule induction, J. Comput. Sci. 16 (2020) 1019–1028.
[35] E. Zorarpacı, S.A. Özel, Privacy preserving rule-based classifier using modified artificial bee colony algorithm, Expert Syst. Appl. 183 (2021) 115437.
[36] M. Talebi, M. Abadi, Beeminer: a novel artificial bee colony algorithm for classification rule discovery, in: 2014 Iranian conference on intelligent systems (ICIS), IEEE, Bam, Iran, 2014, pp. 1-5.
[37] R. Cheruku, D.R. Edla, V. Kuppili, SM-RuleMiner: Spider monkey based rule miner using novel fitness function for diabetes classification, Comput. Biol. Med. 81 (2017) 79–92.

[38] A.H. Alkeshuosh, M.Z. Moghadam, I. Al Mansoori, M. Abdar, Using PSO algorithm for producing best rules in diagnosis of heart disease, in: 2017 international conference on computer and applications (ICCA), IEEE, Doha, Qatar, 2017, pp. 306-311.
[39] J. Basiri, F. Taghiyareh, S. Gazani, in: CORER: A New Rule Generator Classifier, IEEE, Hong Kong, China, 2010, pp. 64–71.
[40] M.M. Malik, H. Haouassi, Efficient sequential covering strategy for classification rules mining using a discrete equilibrium optimization algorithm, Journal of King Saud University-Computer and Information Sciences 34 (2022) 7559–7569.
[41] J.A. Castellanos-Garzón, E. Costa, J.L.S. Jaimes, J.M. Corchado, An evolutionary framework for machine learning applied to medical data, Knowl.-Based Syst. 185 (2019) 104982.
[42] D. Karaboga, Artificial bee colony algorithm, Scholarpedia 5 (2010) 6915.
[43] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to algorithms, 3rd ed., MIT Press, 2009.
[44] H. Liu, F. Hussain, C.L. Tan, M. Dash, Discretization: An enabling technique, Data Min. Knowl. Disc. 6 (2002) 393–423.
[45] E. Frank, M.A. Hall, I.H. Witten, The WEKA Workbench, Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", 4rd ed., Morgan Kaufmann, 2016.
[46] J. Vanschoren, J.N.V. Rijn, B. Bischl, L. Torgo, OpenML: networked science in machine learning, ACM SIGKDD Explorations Newsletter 15 (2014) 49–60.
[47] M. Kelly, R. Longjohn, K. Nottingham. The UCI Machine Learning Repository. Available at: https://archive.ics.uci.edu.
[48] J. Basiri, F. Taghiyareh, Introducing a socio-inspired swarm intelligence algorithm for numerical function optimization, in: 2014 4th international conference on computer and knowledge engineering (ICCKE), IEEE, 2014, pp. 462-467.
[49] A.E. Eiben, J.E. Smith, Introduction to evolutionary computing, Springer, 2015.
[50] R. Cheng, Y. Jin, A social learning particle swarm optimization algorithm for scalable optimization, Inf. Sci. 291 (2015) 43–60.