Expert Systems With Applications 250 (2024) 123897

Contents lists available at ScienceDirect

Expert Systems With Applications

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

ROPAC: Rule OPtimized Aggregation Classifier

Melvin Mokhtari[1], Alireza Basiri [*,2]

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

ARTICLE INFO

Dataset link: http://dx.doi.org/10.24433/CO.7399708.v2, http://colab.research.google.com/drive/1KrBPTdyYXqQqEslnS5UJ1oKho8sDfODJ, http://github.com/MelvinMo/ROPAC-Rule-OPtimized-Aggregation-Classifier

Keywords:
Rule OPtimized Aggregation Classifier (ROPAC)
Rule Aggregation ClassifiER (RACER)
Rule-based classifier
Data classification
Data mining

ABSTRACT

In the era of data-driven decision-making, extracting meaningful insights from vast amounts of information is paramount. In organizing this data, classification methods play a pivotal role. Among the existing classification techniques, rule-based classifiers have gained prominence for their effectiveness and interpretability. One such is the **R**ule **A**ggregation **C**lassifi**ER** (RACER), known for its exceptional performance but limited when dealing with high-dimensional, low-sample-size datasets. In this paper, we introduce the **R**ule **OP**timized **A**ggregation **C**lassifier (ROPAC) as an extension of RACER that incorporates two different rule optimization methods, resulting in ROPAC-L and ROPAC-M, which aim to improve overall performance and classification accuracy. We evaluated this algorithm through experimentation with fifty datasets from reputable sources, such as the OpenML website and the UCI Machine Learning Repository. Furthermore, the proposed algorithm's accuracy is compared with fifteen well-known classifiers. Our results demonstrate that ROPAC outperforms all the other algorithms in terms of accuracy, showcasing its superiority and dominance in various data scenarios.

1. Introduction

The integration of technology into our daily lives has resulted in a significant increase in data production. As a result, data classification algorithms have garnered considerable attention due to their role in organizing and extracting information from these vast data volumes (Najafabadi et al., 2015). These algorithms are crucial for effective decision-making and directly impact machine learning systems, facilitating information management. They find applications in diverse fields such as healthcare and medicine (Battista et al., 2023; Hegde et al., 2022; Thanakiattiwibun et al., 2023), business and finance (Lukáčová & Maličká, 2022; Wu et al., 2020), agriculture (Fang et al., 2023), education (Nayani et al., 2023; Wang et al., 2022), and even more, highlighting the growing demand for accurate classifiers (Bhattacharjee & Manwani, 2020).

Data classification techniques can broadly be categorized into three main groups: eager learners, lazy learners, and other classification techniques (Paul & Kumar, 2019). Eager learners, including decision trees (such as CART, C4.5, and LMT, or the newer ones like ForestPA (Adnan & Islam, 2017) and SPAARC (Yates et al., 2019)), Bayesian classification (Naïve Bayes), rule-based classification (RACER, the JRip implementation of RIPPER, PART), support vector machines (SMO in SVM), and backpropagation-based neural networks (MLP), actively construct a model from the training data before making predictions. Lazy learners, represented by the K-Nearest Neighbor (KNN) classification (like the IBk implementation of KNN), delay model construction until a new instance needs to be classified. Other classification techniques encompass fuzzy logic, genetic algorithms, and the rough set approach, offering alternative approaches to data classification. This category could also include ensemble learning techniques such as Bagging, Random Forest, and Optimized Forest (Adnan & Islam, 2016), which combine multiple models to make predictions. These diverse techniques provide a wide range of options for efficiently and accurately solving classification tasks.

In this context, rule-based data classification algorithms have emerged as an eminent approach, offering simplicity, interpretability, and transparency in decision-making (Kesavaraj & Sukumaran, 2013; Thangaraj & Vijayalakshmi, 2013). These algorithms generate rules that humans understand easily, making them particularly valuable in domains where interpretability is crucial (Liu et al., 2016). Rule-based algorithms like RIPPER (Cohen, 1995) and PART (Frank & Witten, 1998) function by creating rules that collectively define the decision boundaries for classifying data instances. By analyzing the input features of a given dataset, these algorithms generate rules in the form of "if-then" statements, where specific conditions on input features determine class labels.

Rule Aggregating ClassifiER, or RACER (Basiri et al., 2019), is another rule-based classification algorithm with similar settings that uses

[*] Corresponding author.
E-mail addresses: melvin.mokhtari@ec.iut.ac.ir (M. Mokhtari), basiri@iut.ac.ir (A. Basiri).
[1] ORCID: 0000-0003-3911-3099
[2] ORCID: 0000-0001-8073-5987

https://doi.org/10.1016/j.eswa.2024.123897
Received 22 January 2024; Received in revised form 20 March 2024; Accepted 1 April 2024
Available online 5 April 2024

training dataset records to create its initial rules. However, these rules are often overly specific, rendering them inadequate for classifying new data. This issue becomes even more pronounced when dealing with small datasets containing a limited number of distinct instances. RACER combines these initial rules to make more generalized ones that cover similar cases to address this limitation; however, this approach still falls short of improving accuracy in such scenarios. Therefore, creating better rules is a possible solution to overcoming this obstacle.

In this study, we propose the Rule OPtimized Aggregation Classifier (ROPAC), a novel rule-based classification algorithm, as an improved version of RACER. By introducing ROPAC, this paper seeks to enrich the performance of RACER in handling diverse datasets as well as high-dimensional, low-sample-size ones by integrating two new rule optimization methods, the most frequent and least frequent zero methods. The initial goal is to achieve higher accuracy than RACER, and then we hope to outperform the other widely recognized classifiers in terms of performance in both small and large datasets and in a broader scenario. As a result, we anticipate a substantial improvement in overall performance and accuracy.

The rest of the paper consists of the following sections: Section 2 reviews related works. The state-of-the-art RACER algorithm is described in Section 3. Section 4 presents a detailed description of ROPAC, our proposed classifier. A performance study on benchmark datasets and experimental results on classification accuracy are covered in Section 5. Finally, Section 6 concludes the paper by discussing the findings and future directions.

2. Related work

Rule-based classifiers are a specific category of data classification algorithms that extract a collection of rules from a provided training dataset. These rules subsequently assign newly encountered instances to their appropriate classes. Rule-based classifiers can be perceived as an expansion of decision tree classifier algorithms, not constrained to a hierarchical tree-like structure for representation (Palliser-Sans, 2021). There are plenty of examples available. Among them, PRISM (Cendrowska, 1987) is considered one of the first rule-based classifiers. This algorithm selects all examples from a specific class to generate rules. IREP is another one, which integrates reduced error pruning with a divide-and-conquer approach to minimize errors (Fürnkranz & Widmer, 1994). RIPPER is relatively similar, learning a set of rules from data by iteratively adding rules until a stopping criterion is met (Cohen, 1995). In PART, one rule is learned at a time, and there is no need for global optimization since the rules are generated repeatedly (Frank & Witten, 1998). This is only a brief history of the first algorithms in this field. In recent years, there has been a growing interest in studying these particular data classification algorithms and working on ideas for optimizing extracted rules to enhance search space exploration performance. Recent research has provided valuable information on overriding the challenges of existing algorithms.

In the rule generation and optimization domain, Hońko (2019) introduced a method for binary classification rule generation from decomposed data. This method emphasizes the division of each class into the same arbitrary small number of subtables to improve the accuracy of the classification process. In another approach, Mao et al. (2020) proposed a particle swarm optimization method that incorporated quantum qubit operations. Their objective was to construct and optimize fuzzy rule-based classifiers, underscoring the significance of addressing computational complexities and optimizing rule-based classifiers to improve prediction accuracy. Also, Fuchino et al. (2023) conducted a study to investigate the computational complexity associated with allowed rule ordering and its greedy algorithm. Their research shed light on the challenges of rule optimization and emphasized the NP-hard nature of certain rule-ordering processes by investigating computational complexity.

Fang et al. (2020) proposed a balance-adjusting approach within an extended belief-rule-based system for addressing imbalanced classification problems. This strategy aimed to reduce the sharp drop in accuracy observed for minority classes by adjusting the imbalances in the dataset. Further, Qiao et al. (2021) introduced a new approach to learning a collection of independent logical rules in disjunctive normal form. The researchers emphasized the importance of interpretable decision rule sets and focused on developing a model to explain classification outcomes.

In 2019, scientists introduced an annealing strategy to enhance a rule pruning technique in an Ant Colony Optimization (ACO)-based rule classification system. This strategy addressed the issue of generating low-coverage, complex rules with irrelevant terms, emphasizing the importance of rule pruning in achieving accurate classification results (Al-Behadili et al., 2019). Jabba (2021) took a different approach, focusing on rule induction. The iterated local search technique was used to generate effective classification rules that captured the underlying patterns and relationships in the data.

Other new rule-based algorithms include eRules (Stahl et al., 2012), which handles streaming data using a sliding window to terminate incorrect classifications immediately. A newer version of this algorithm, G-eRules (Le et al., 2014), responds to the issues of real-time rule induction. RRULES is another classifier that was presented by Palliser-Sans (2021). This inductive learning algorithm was created to enhance the RULES algorithm (Pham & Aksoy, 1995) by extracting rules from a given set of training examples. RRULES takes a straightforward approach to identifying and eliminating irrelevant rules while emphasizing the importance of verifying stopping conditions throughout the learning process.

Alcalá-Fdez et al. (2011) proposed FARC-HD, a fuzzy associative classification method that addresses scalability issues with high-dimensional data by combining a genetic algorithm approach with rule selection and lateral tuning of membership functions. In a new approach, Sanz et al. (2021) developed FARCI, which modifies all stages of the FARC-HD classifier to better handle imbalanced classification problems. FARCI used lift measures for rule generation and modified rule selection and fitness evaluation to improve rules for minority classes.

In a study by Huynh et al. (2023), a rule-learning algorithm named LORD was developed to identify the optimal rule for each training example through a greedy optimization process involving specialization and generalization loops. Also, the RUCIB algorithm was introduced by Morovatian et al. (2023). This classifier has a mechanism that simultaneously looks at all data features, with each feature within a rule being assigned multiple values. NFFT-DFSC is another algorithm that was introduced by Yin et al. (2023). This algorithm employed a stacked architecture of multiple fuzzy models with nonlinear feature transform functions to extract high-level latent fuzzy features from data, resulting in enhanced outcomes for large, high-dimensional datasets.

In another study conducted in 2019, researchers presented the aforementioned RACER algorithm as a high-performance rule-based classifier. The algorithm is distinguished by its distinct rule representation and its rule combination technique, making RACER a robust classifier (Basiri et al., 2019). Using RACER, Toulabinejad et al. (2024) proposed the integration of multiple discretization methods, like CAIM, CART, ChiMerge, and MDLP, compared with the default discretization method of the algorithm, Information Gain, to study the potential for an enhancement in the accuracy and understandability of the classifier. They demonstrate that RACER gets higher accuracy with the MDLP method.

Although RACER has notable capabilities, it struggles with analyzing high-dimensional, low-sample-size datasets. As previously mentioned, this limitation restricts algorithms' applicability to various scenarios and dataset types. This paper introduces the Rule OPtimized Aggregation Classifier (ROPAC), a novel RACER extension that aims to achieve better overall accuracy in diverse data scenarios while laterally modifying the aforementioned limitations as attainable.

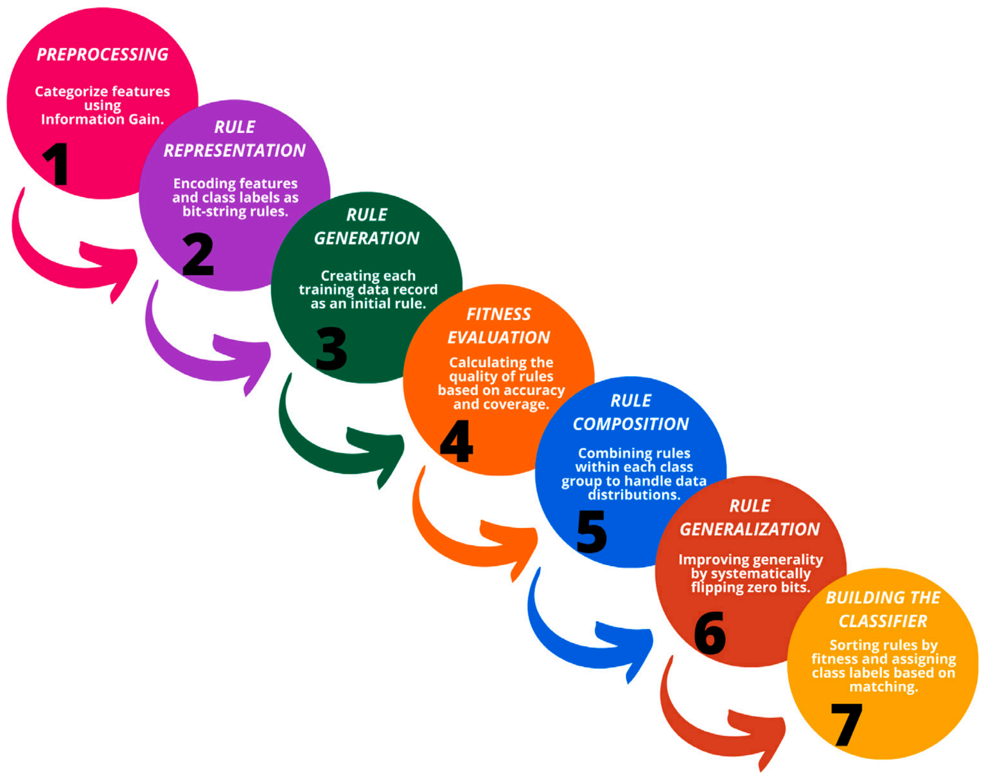
PREPROCESSING
Categorize features using Information Gain.
1
RULE REPRESENTATION
Encoding features and class labels as bit-string rules.
2
RULE GENERATION
Creating each training data record as an initial rule.
3
FITNESS EVALUATION
Calculating the quality of rules based on accuracy and coverage.
4
RULE COMPOSITION
Combining rules within each class group to handle data distributions.
5
RULE GENERALIZATION
Improving generality by systematically flipping zero bits.
6
BUILDING THE CLASSIFIER
Sorting rules by fitness and assigning class labels based on matching.
7

Fig. 1. Flow diagram of the RACER algorithm.

3. The state-of-the-art classifier: RACER

The rule-based classifier RACER transforms each training record into an initial rule and then merges to form more general rules, opening up a plethora of options for rule merging (Basiri et al., 2019). This algorithm, unlike state-of-the-art classifiers that choose features individually (Koren et al., 2020), utilizes all attributes at the same time, leading to faster and more precise classification. Hence, this remarkable, cost-free approach makes it an efficient and accurate choice for contemporary data classification. Fig. 1 depicts the workflow of this classifier.

As outlined in Fig. 1, the algorithm is divided into several phases, including preprocessing, rule representation, initial rule creation, fitness evaluation, rule composition, rule generalization, and sorting and building the classifier. The process begins with converting numerical data to categorical equivalents. RACER then uses a bit string format to represent rules. Within this style, each individual bit corresponds to a specific attribute value of presence or absence, encoding both features and class labels. This one-bit-per-value scheme yields uniform rule lengths, making comparisons and computations more efficient. For n features $(F_1, \dots, F_n)$ and m class attributes, each rule has a fixed size of $(\sum_{i=1}^{n} n_i + m)$ bits, where $\sum_{i=1}^{n} n_i$ sums the unique values across features. This representation simplifies rule analysis within the framework.

To clarify, assume we have a record R representing if conditions, with features F_1 to F_5 and a class label F_C, and the following domains:

$$F_1 \in \{\text{spring, winter}\}$$
$$F_2 \in \{\text{sunny, rainy}\}$$
$$F_3 \in \{\text{good, unhealthy}\}$$
$$F_4 \in \{\text{hot, cold}\}$$
$$F_5 \in \{\text{high, low}\}$$
$$F_C \in \{\text{true, false}\}$$

Here, R will be represented in 12 bits, i.e., 2 bits for F_1, 2 for F_2, 2 for F_3, 2 for F_4, 2 for F_5, and 2 for the class label, which could be shown in the following format:

F_1	F_2	F_3	F_4	F_5	F_C
10	10	01	10	11	10

That is, if (F_1 = spring) and (F_2 = sunny) and (F_3 = unhealthy) and (F_4 = hot) and (F_5 = high or low), then (F_C = true).

In the next phase, the algorithm generates its initial rules based on each training data instance. Each rule's validity is then evaluated using a fitness function. The rules are then randomly combined using the logical "OR" operation. In the following phase, this heuristic algorithm generalizes each rule by flipping all possible zero bits of the rule to one. As soon as the first rule with higher fitness is detected, the algorithm flips that bit, changes the original rule, and then again proceeds with the remaining bits of the same rule, disregarding other possibilities that may potentially lead to better fitness. Finally, the classifier will be constructed.

Clearly, the algorithm's performance is significantly impacted by the quality of the initial rules, which, in turn, is also contingent upon the dataset's characteristics. This is because the initial rules, originally derived from the training data instances, are composed to make the classifier, so the search space that RACER explores is confined to these rules and states.

As high-dimensional and low-sample-size datasets are involved, this will become more prominent. Technically speaking, this term refers to datasets with many features (high-dimensional) but a relatively small number of observations (low-sample-size). In high-dimensional datasets, there are many more variables or features than observations, while low-sample-size datasets have a limited number of observations or instances available for analysis. The combination of high dimensionality and a low-sample-size poses additional challenges, and traditional methods may not be suitable for analyzing such datasets due to issues like multicollinearity and the lack of sufficient observations to estimate model parameters accurately. This poses challenges for many algorithms, including RACER. Let us clarify this through an example.

We have the following dataset with 6 features and class labels, each with two variables but only 3 instances:

F_1	F_2	F_3	F_4	F_5	F_C
10	10	01	10	11	10
01	10	01	11	01	10
11	10	01	11	11	10

In this example, the size of the search space is $2^{10+1} - 2^6 - 1$, which accounts for a total of 2^{10} ways to represent our five features, each with two bits, added by one because a class label can logically be only one bit at a time, while excluding all-zero non-defined states in each of our six features and class labels. This value is equal to the sum of binomial coefficients from $i = 1$ to n, where each binomial coefficient is denoted as "n choose i", defined as $\sum_{i=1}^{n} \binom{n}{i} = 2^n - 1$, here calculated as $\sum_{i=1}^{6} \binom{6}{i} = 2^6 - 1$. By subtracting this amount from the total number of possible states, 2^{11}, we obtain the number of valid states in this expansive search space, $2^{11} - 2^6 - 1 = 1985$. In this context, we possess a total of 3 instances out of the 1985 possible states, resulting in a mere 0.15% accessibility of the model to the search space. Let us momentarily set this aside and explore this event at other stages.

In the rule composition phase, rules are randomly merged. If the composition results in a rule with higher fitness, the original rule updates, and the composition process continues with the remaining rule(s). Careful observations reveal that this phase will be completed with a maximum of $(N - 1)^2$ steps, where $N > 2$ is the number of rules in the rule set. In our example, the rule set consists of rules $\{R_1, R_2, R_3\}$, which allow a maximum of $(3 - 1)^2 = 4$ ways of composition. These combinations are $\{R_1, R_2\}$, $\{R_1, R_3\}$, $\{R_2, R_3\}$, and $\{NewRule, RemainingRule\}$. That means we can explore 4 more states out of the total of 1985 states in our example. Now, let us step further. Assume that the first two initial rules are randomly selected to be composed together, creating the following new rule:

F_1	F_2	F_3	F_4	F_5	F_C
11	10	01	11	11	10

We assume this rule has higher fitness, so the old rules will be replaced. Remarkably, the new rule also coincides precisely with the third rule from the rule set, so the model will have only one rule now. This rapid and seemingly uncontrolled process can be attributed to the limited availability of suitable alternative rules in low-sample-size datasets. In fact, we built this classifier on only one of the 1985 possible states and overlooked other potential options. Given its sole reliance on one rule, the model now encounters the daunting task of classifying the test set, which comprises potentially diverse and distinct records, resulting in poor accuracy. Then arise key questions:

- How can we explore more of the search space than the heuristic RACER?
- What if we were to commence the process with better initial rules?
- What tradeoffs exist between computational complexity and accuracy?

This is where we propose ROPAC.

4. The proposed classifier: ROPAC

The Rule OPtimized Aggregation Classifier (ROPAC) tries to generate enhanced initial rules through two rule optimization methods: the most frequent and least frequent zero bits. This approach yields a set of improved initial rules, a controlled step of rule generalization, which subsequently results in more secure options for rule merging. Algorithm 1 illustrates the ROPAC classifier's pseudocode.

The algorithm indeed shares several similarities with the steps briefly discussed for RACER. Following the stages of preprocessing, rule representation, and initial rule creation, ROPAC will perform its novel rule optimization phase. These two new optimization approaches are presented as ROPAC-L and ROPAC-M. Subsequently, the algorithm proceeds with fitness evaluation and the new primary rule generalization step. The remaining phases of the algorithm remain unchanged, including rule composition, the secondary rule generalization phase, and, ultimately, classifier construction. In the subsequent sections, we will delve into these steps' details and technical specifications.

Algorithm 1 ROPAC algorithm

```
Input: Training dataset $D_{train}$
Output: The set of decision rules $R_{set}$
 1: procedure ROPAC($D_{train}$)
 2:     Preprocessing
 3:     Create Initial_Rules
 4:     Extant_Rules = Initial_Rules
 5:     Rule Optimization
 6:     Primary Generalization of Extant_Rules
 7:     for $i = 1$ to $|Extant\_Rules|$ do
 8:         for $j = i + 1$ to $|Extant\_Rules|$ do
 9:             Composed_Rule = Rule Composition($R_i, R_{i+1}$)
10:             if Composed_Rule is better than both $R_i$ and $R_{i+1}$ then
11:                 Replace both $R_i$ and $R_{i+1}$ with Composed_Rule
12:                 Omit all covered rules with Composed_Rule
13:                 Update Extant_Rules
14:             end if
15:         end for
16:     end for
17:     Secondary Generalization of Extant_Rules
18:     Sort the Extant_Rules Based on Fitness
19:     $R_{set}$ = Extant_Rules
20: end procedure
```

4.1. Preprocessing

ROPAC requires categorical domains for the features in order to function effectively. For this, we used the Information Gain technique to split continuous features into discrete categories, with the feature that provides the most significant reduction in uncertainty serving as the algorithm's optimal split point (Omuya et al., 2021). The primary stage in implementing this strategy for feature F within the dataset D_{train}, which encompasses N records, entails arranging the F values in an ascending sequence. As per the Information Gain criterion, MidPoint_i is defined as the split point between m_i and m_{i+1}, two adjacent values in F:

$$\text{MidPoint}_i = \frac{m_i + m_{i+1}}{2}, \quad \text{for} \quad i = 1, 2, \ldots, N \tag{1}$$

The function $\text{Info}(D_{\text{train}})$, where c is the count of classes and P_i is the likelihood that an instance is in that particular class, is defined for each MidPoint_i in the following way:

$$\text{Info}(D_{\text{train}}) = -\sum_{i=1}^{c} P_i \log_2(P_i) \tag{2}$$

Now, to calculate $\text{Info}_F(D_{\text{train}})$, where D_1 is a group of records with $F \le \text{MidPoint}_i$ and D_2 is a group of records with $F > \text{MidPoint}_i$, the function $\text{Info}_F(D_{\text{train}})$ will be:

$$\text{Info}_F(D_{\text{train}}) = \frac{|D_1|}{|D_{\text{train}}|} \cdot \text{Info}(D_1) + \frac{|D_2|}{|D_{\text{train}}|} \cdot \text{Info}(D_2) \tag{3}$$

So, for each MidPoint_i in F, we compute $\text{Info}_F(D_{\text{train}})$, and the one with the lowest $\text{Info}_F(D_{\text{train}})$ will act as the optimal split point.

4.2. Rule representation

ROPAC utilizes a bit-wise rule representation that aligns with the approach employed by RACER. In ROPAC, each feature is assigned bits based on the cardinality of its values. A bit value of one signifies acceptance of the corresponding feature value, whereas zero indicates that it is not.

Compared to classifiers like ID3 (Quinlan, 1986), PART (Frank & Witten, 1998), and C4.5 (Salzberg, 1994), this representation provides two advantages: first, it allows for the creation of more general rules through rule merging, and second, it permits the use of multiple bits for a feature. That is what distinguishes it from other techniques.

4.3. Initial rules creation

The algorithm encodes each record in the dataset, considering its specific value as a string represented by 0s and 1s, to create initial rules. For instance, a training dataset with N data records produces N separate initial rules, forming the algorithm's rule set.

4.4. Rule optimization

The ROPAC algorithm incorporates an additional rule optimization step to improve the quality and diversity of the initial rules. To do so, it identifies the rule set's most and least frequent zero bits. Then, it selectively uses them to replace underperforming rules with optimized counterparts to strike a compromise between generality and effectiveness. In the following section, we will go over these distinct optimization techniques in depth.

4.4.1. ROPAC-M

As a variant of ROPAC, ROPAC-M focuses on finding the most frequent zero bits in a rule set. This approach involves some steps. Initially, an array is employed to track the frequency of zero bits throughout the entire rule set. This is accomplished by sequentially examining each rule and updating the *Frequency_Array* accordingly. The array contains a tally of zero-valued bits for each bit position, so it will be sorted in descending order, starting with the most frequent zeros. Subsequently, the optimization function iterates through the rules individually. The process continues by flipping the first matching bit position in the sorted *Frequency_Array* (also zero in the current rule) to one. The fitness of this rule is then calculated, and if it outperforms the original rule, it replaces it in the *Extant_Rules* set and continues with the next rule. The procedure halts if a rule with a higher fitness score is encountered; however, if this condition is not met, the process persists until all zero-valued bit positions of the current rule have been examined. In the event that no improvement is observed or a class conflict arises, the original rule will be preserved. Let us clarify this with our previous example:

F_1	F_2	F_3	F_4	F_5	F_C
10	10	01	10	11	10
01	10	01	11	01	10
11	10	01	11	11	10

Here, we have 3 rules (R_1 to R_3), each with 10 bits (b_1 to b_{10}), plus the 2 bits for the class label, which act as a control bit to ensure the new optimized rule is within the same class as the original rule. For this rule set, the ROPAC-M's *Frequency_Array* will be sorted descendingly as follows:

b_4	b_5	b_1	b_2	b_8	b_9	b_3	b_6	b_7	b_{10}
3	3	1	1	1	1	0	0	0	0

This implies that the fourth bit holds a higher priority for being flipped to one, followed by the fifth bit, and so on until the ninth bit, which is the last possible zero bit. Then the next step begins. In the first rule, ROPAC-M sets the fourth bit to one, calculating its fitness and comparing it to the original value. If the fitness is higher, the rule will be replaced, and we will proceed to the second rule. If not, we will move on to the fifth bit. This process continues until the rule with the highest fitness is discovered; if not, the original rule will be kept, and the algorithm moves on to the subsequent rule.

In this particular scenario, we have the opportunity to explore a maximum of L new possibilities for each rule, in addition to the 3 initially known states out of the 1985 possible states. Here, L represents the number of zero-bit positions in rule R, assuming an ideal scenario without any duplicate rules. In the case of R_1, R_2, and R_3, the number of zero-bit positions is 4, 4, and 2, respectively, which means that we can ideally explore up to 10 new states that have not been visited so far. This observation highlights a distinct characteristic of ROPAC-M, which in this example has quadrupled the search space exploration at the same stage compared to RACER.

As mentioned earlier, generating improved rules provides a broader scope for rule merging. This enables exploration across a wider range of scenarios within the search space, thereby increasing the potential for an optimized final rule set. On the other hand, high-frequency zero bits have been investigated less in the training process, making them less likely to have information about them in the model and indicating greater flexibility in bit position. Flipping these frequent zero bits expands the matching range, enhancing rule performance and adaptability to new and unseen data. Algorithm 2 shows the implementation of the rule optimization phase in ROPAC-M in particular.

Algorithm 2 Most Frequent Zero Bit Rule Optimization of ROPAC-M

```
Input: Extant_Rules
Output: Updated list of Extant_Rules
 1: procedure MFZB(Extant_Rules)
 2:     for each rule $i$ in Extant_Rules do
 3:         Identify all 0-valued bit indices
 4:         Increment Frequency_Array for each bit position
 5:     end for
 6:     Sort indices of Frequency_Array in descending order
 7:     for each rule $i$ in Extant_Rules do
 8:         for each Frequency_Array index $j$ do
 9:             Flip bit at index $j$ in Optimized_Rule
10:             Calculate Fitness of Optimized_Rule
11:             if Fitness improves then
12:                 Replace rule $i$ with Optimized_Rule
13:                 Update Extant_Rules
14:                 Continue with rule $i+1$
15:             end if
16:         end for
17:     end for
18: end procedure
```

4.4.2. ROPAC-L

For identifying the least frequent zero bits, we offer ROPAC-L. The whole process is similar to the previous variant, but the *Frequency_Array* will be sorted in ascending order this time. Here, we target infrequent zero bits (the most frequent one-valued bits, currently zero) to flip them to one potentially. Let us consider the previous example once more, but this time with the ascending ordered *Frequency_Array*:

b_3	b_6	b_7	b_{10}	b_1	b_2	b_8	b_9	b_4	b_5
0	0	0	0	1	1	1	1	3	3

In this case, we prioritize flipping every zero bit before bit 4 and 5. As bits 3, 6, 7, and 10 are not zero even once, the flipping process will commence from bit 1. Now reconsider the first rule:

F_1	F_2	F_3	F_4	F_5	F_C
10	**10**	**01**	**10**	**11**	**10**
01	10	01	11	01	10
11	10	01	11	11	10

It is evident that the first rule contains a non-zero bit at position 1. In the ROPAC approach, non-matching bits are initially disregarded, and the process starts with the next bit (bit 2) and continues until a bit with higher fitness is found. This method can also ideally explore up to 10 new states as ROPAC-M; however, some slight differences can be considered here.

During the rule composition phase, the ROPAC algorithm uses logical "OR" operations to merge individual rules and create new ones. So, when ROPAC-M flips the least frequent zero bits, each rule that is to be merged with the rules generated by ROPAC-M has a high chance of being affected. This is because of the nature of the logical "OR" operation with a least frequent zero bit (now flipped to one) that will flip those mainly zero bits to one easily. As a result, the algorithm may lose control over rule merging as the conversion of those bits happens rapidly. Although exploring more of the search space will be achieved, this phenomenon can lead to negative consequences in subsequent classification steps. While on the other hand, ROPAC-L handles this process more smoothly by flipping the zero bit that was predominantly set to one. This approach prevents sudden rule-merging movements, resulting in gradual exploration, more interpretable rules, and a greater likelihood of matching similar data points.

4.5. Fitness evaluation

Each rule's quality will be assessed using a fitness function that takes both accuracy and coverage into account. This function is defined as follows:

$$Fitness(R_i) = \alpha \times Accuracy(R_i) + \beta \times Coverage(R_i) \tag{4}$$

Here, R_i represents a rule in the rule set, and α and β are weighting factors that determine the relative importance of Accuracy and Coverage, respectively. These weighting factors must satisfy conditions $\alpha, \beta > 0$ and $\alpha + \beta = 1$.

Accuracy(R_i) measures the proportion of correctly classified records among those covered by the rule R_i. It is computed as the ratio of correctly classified records (n_{correct}) to the total number covered (n_{covers}) by the R_i rule:

$$\text{Accuracy}(R_i) = \frac{n_{\text{correct}}}{n_{\text{covers}}} \tag{5}$$

On the other hand, Coverage(R_i) measures the proportion of records in the training dataset D_{train} that rule R_i covers. It is calculated as the ratio of the number of records the rule R_i covers (n_{covers}) to the total number of records in the training dataset ($|D_{\text{train}}|$).

$$\text{Coverage}(R_i) = \frac{n_{\text{covers}}}{|D_{\text{train}}|} \tag{6}$$

This method ensures that the resulting rule sets can make accurate predictions while also being applicable to a significant amount of data.

4.6. Primary rule generalization

ROPAC incorporates a two-step rule generalization architecture. This new primary rule generalization phase aims to enhance the generality of the rule set by considering both the initial and newly generated optimized rules. The process starts with identifying the bits within rule R with a value of zero. Setting each of these bits to one improves R's generality. Hence, it systematically modifies the first 0-valued bit by changing it to one. If this modification improves the fitness value, it will be updated and continued with the next bit of the same rule; otherwise, it will be ignored, and the next bit of the same rule will be examined. This iterative process will be repeated for all remaining 0-valued bits in R. After the completion of this process for the current rule, ROPAC moves on to the following rule and repeats the loop until all bits of all rules have been visited.

4.7. Rule composition

The initial rules are perfect when applied to the training set, but their high specificity for individual instances causes them to perform poorly when applied to the test set. To overcome this limitation, ROPAC utilizes a rule composition strategy. This involves separating the initial rules based on class labels and composing them within each class group, aiming to generate more general rules applicable to a wider range of data. Our example explains this approach:

F_1	F_2	F_3	F_4	F_5	F_C
10	10	01	10	11	10
01	10	01	11	01	10

After combining these two rules, a new rule will be generated based on the logical "OR" operation of those individual rules, which is:

F_1	F_2	F_3	F_4	F_5	F_C
11	10	01	11	11	10

This rule can also be simplified. Given that we can ignore the barren and not informative features of F_1, F_4, and F_5, which contain only 1s, the final rule resulting from merging these two initially 12-bit rules will be one 6-bit rule, as shown below:

F_2	F_3	F_C
10	01	10

To get deeper, the process begins with the first rule in each group and merges it repeatedly at random with the rest of the rules. Next, we compute the merged rule's fitness value, and if it is greater than the fitness of the initial rules, they are replaced in the rule set by the generated rule. This iterative process is repeated until no further improvements can be made, allowing the algorithm to adapt its rules to many data distributions and achieve improved performance.

4.8. Secondary rule generalization

Since the previous generalization phase of ROPAC, we have generated new generalized and composed rules, resulting in significant changes to the rule set. In this second phase, using a methodology similar to the primary phase, we aim to generalize the newly obtained rules before finally constructing the classifier.

4.9. Building the classifier

As the first step in building the classifier, the rules are organized in descending order based on their fitness values, prioritizing rules generated earlier in cases of equal fitness values. Then, for classifying new records, the algorithm sequentially evaluates the instance's adherence to the rules, starting with the rule of highest fitness. The associated class label is assigned if the instance satisfies the rule's conditions. This process continues until a covering rule is found.

Regarding complexity considerations, as ROPAC emphasizes enhancing the quality of rules rather than generating new ones, the number of rules remains fixed throughout the classification task. Moreover, the processes are performed for each rule one by one, eliminating the need for duplication and consequently avoiding additional space overhead compared to RACER. However, since new steps are introduced in ROPAC, we have slightly higher time complexity as a natural consequence. This tradeoff will be justified by ROPAC's greater exploration of the search space and the possibility of increased accuracy, making it a worthwhile approach.

Table 1
Dataset characteristics.

#	Dataset name	# of inst.	# of feat.	# of classes	# of mis. val.	# of num. attr.	# of cat. attr.
1	chscase_vine1	52	10	2	0	9	1
2	dbworld-bodies	64	4703	2	0	0	4703
3	pyrim	74	28	2	0	27	1
4	kidney	76	7	2	0	3	4
5	analcatdata_asbestos	83	4	2	0	1	3
6	baskball	96	5	2	0	4	1
7	analcatdata_chlamydia	100	4	2	0	0	4
8	fertility	100	10	2	0	9	1
9	molecular-biology_promoters	106	58	2	0	0	58
10	fruitfly	125	5	2	0	2	3
11	mux6	128	7	2	0	0	7
12	analcatdata_boxing2	132	4	2	0	0	4
13	newton_hema	140	4	2	0	2	2
14	lymph	148	19	4	0	3	16
15	tae	151	6	3	0	3	3
16	analcatdata_wildcat	163	6	2	0	3	3
17	servo	167	5	2	0	0	5
18	parkinsons	195	23	2	0	22	1
19	pwLinear	200	11	2	0	10	1
20	cpu	209	8	2	0	6	2
21	seeds	210	8	3	0	7	1
22	chatfield_4	235	13	2	0	12	1
23	heart-statlog	270	14	2	0	13	1
24	breastTumor	286	10	2	9	1	9
25	heart-h	294	14	2	782	6	8
26	cholesterol	303	14	2	6	6	8
27	cleveland	303	14	2	6	6	8
28	haberman	306	4	2	0	2	2
29	ecoli	327	8	5	0	7	1
30	liver-disorders	345	7	2	0	6	0
31	dermatology	366	35	6	8	1	34
32	braziltourism	412	9	7	96	4	5
33	pbc	418	19	2	1239	10	9
34	vote	435	17	2	392	0	17
35	thoracic-surgery	470	17	2	0	3	14
36	threeOf9	512	10	2	0	0	10
37	kc2	522	22	2	0	21	1
38	wdbc (Breast Cancer Wisconsin)	569	31	2	0	30	1
39	monks-problems-2	601	7	2	0	0	7
40	eucalyptus	736	20	2	448	14	6
41	diabetes	768	9	2	0	8	1
42	stock	950	10	2	0	9	1
43	tokyo1	959	45	2	0	42	3
44	xd6	973	10	2	0	0	10
45	flare	1066	11	2	0	0	11
46	parity5_plus_5	1124	11	2	0	0	11
47	cmc	1473	10	2	0	2	8
48	kr-vs-kp	3196	37	2	0	0	37
49	led7	3200	8	10	0	0	8
50	nursery	12 960	9	5	0	0	9

5. Experiments

We thoroughly evaluated the ROPAC algorithm, particularly its variants, ROPAC-L and ROPAC-M, using a microcomputer powered by an Intel® Core™ i5-3337U CPU running at 1.80 GHz and 8 GB of RAM. The experiments were carried out on fifty different datasets using ten-fold cross-validation with a seed value of one to calculate the mean classification accuracy. We then juxtaposed the performance of ROPAC with fifteen other classifiers, ranging from classical to the newly well-liked ones, covering a wide range of approaches such as eager learners, lazy learners, and other data classification techniques, using WEKA Workbench 3.9.6 (Hall et al., 2009) for all except RACER and ROPAC (implemented in Python 3.12). This list of classifiers included ForestPA, LMT, MLP of Neural Networks, Random Forest, Optimized Forest, SPAARC, RACER, Bootstrap Aggregation (Bagging), C4.5, PART, the JRip implementation of RIPPER, SMO in SVM, Decision Tree (CART), IBk implementation of KNN, and Naïve Bayes.

5.1. Data

The ROPAC algorithm was evaluated using a diverse collection of 50 datasets selected from two reliable sources, the UCI Machine Learning Repository (Dua & Graff, 2017) and the OpenML website (Vanschoren et al., 2014). These datasets were chosen due to their widespread use and the variety of characteristics they provide. Table 1 concisely summarizes these benchmark datasets, highlighting key details such as the number of instances, missing values, count of each type of feature, and target variable classes.

Datasets with features ranging from 4 to 4703 were selected for analysis. The collection consists of datasets with categorical features, counting from 0 to 4703, and numerical features, ranging from 0 to 42. Notably, nine datasets contain missing values varying between 6 and 1239. Furthermore, the datasets have diverse target labels, with classes ranging from 2 to 10. In terms of size, the collection includes both compact datasets like “chscase_vine1” with 52 records and larger datasets like “nursery” with 12 960 records.

Table 2
Accuracy comparison of ROPAC variants vs. RACER.

#	Dataset name	ROPAC-L	ROPAC-M	RACER
1	chscase_vine1	**83.67**	**83.67**	79.67
2	dbworld-bodies	**87.62**	**87.62**	**87.62**
3	pyrim	**89.11**	**89.11**	86.43
4	kidney	**75**	73.75	71.25
5	analcatdata_asbestos	**76.11**	**76.11**	72.5
6	baskball	**74.22**	**74.22**	59.56
7	analcatdata_chlamydia	**90**	84	83
8	fertility	**86**	**86**	85
9	molecular-biology_promoters	**68**	**68**	**68**
10	fruitfly	**64.29**	**64.29**	64.23
11	mux6	**100**	**100**	**100**
12	analcatdata_boxing2	**75.77**	72.69	69.67
13	newton_hema	**77.14**	76.43	76.43
14	lymph	**83.19**	79.86	82.43
15	tae	**57**	**57**	43.08
16	analcatdata_wildcat	**76.69**	**76.69**	71.07
17	servo	92.83	**92.87**	91.58
18	parkinsons	87.32	**87.34**	86.29
19	pwLinear	85	**87.5**	83.5
20	cpu	97.62	**98.1**	97.14
21	seeds	**91.43**	90.48	88.1
22	chatfield_4	**89.76**	89.33	88.04
23	heart-statlog	**82.59**	**82.59**	79.26
24	breastTumor	57.34	**58.42**	53.53
25	heart-h	83.67	**84.34**	81.68
26	cholesterol	**63.4**	61.39	61.04
27	cleveland	**79.88**	79.23	79.55
28	haberman	**76.17**	74.85	76.16
29	ecoli	81.95	**82.25**	73.99
30	liver-disorders	68.41	**68.7**	59.13
31	dermatology	**99.72**	92.9	99.71
32	braziltourism	**78.9**	76.24	78.43
33	pbc	**70.8**	67.93	69.62
34	vote	**95.15**	**95.15**	94.48
35	thoracic-surgery	**83.4**	83.19	83.19
36	threeOf9	**100**	99.8	99.61
37	kc2	83.74	**84.13**	83.94
38	wdbc (Breast Cancer Wisconsin)	**96.83**	96.48	95.25
39	monks-problems-2	72.73	**73.73**	67.22
40	eucalyptus	**75.82**	75.55	75.54
41	diabetes	**72.4**	70.05	66.93
42	stock	**92.84**	**92.84**	88.32
43	tokyo1	92.7	92.7	**92.91**
44	xd6	**100**	**100**	99.9
45	flare	81.51	81.7	**82.17**
46	parity5_plus_5	**100**	**100**	**100**
47	cmc	**66.73**	66.67	65.17
48	kr-vs-kp	98.78	98.97	**99.22**
49	led7	**71.59**	71.44	63.5
50	nursery	99.71	**99.75**	99.63
AVG		**82.69**	82.12	80.07

5.2. Evaluation criteria

The performance of the ROPAC algorithm was evaluated using the accuracy metric, which assesses the model's ability to assign class labels to instances correctly. Accuracy is calculated as follows:

$$\text{Accuracy} = \frac{\text{Number of correctly classified instances}}{\text{Total number of instances}} \tag{7}$$

To be more precise, it is:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

Here, True Positive (TP) represents correctly predicted positive instances, while True Negative (TN) represents correctly predicted negative instances. False Negative (FN) indicates instances incorrectly predicted as negative when they are actually positive, and False Positive (FP) indicates instances incorrectly predicted as positive when they are actually negative. Thus, accuracy ranges from 0 to 1, where a value of 0 indicates an utterly incorrect classification and a value of 1 represents a perfect classification, which can also be expressed in percentages (Bishop, 2007).

5.3. Results

The main objective of introducing the ROPAC algorithm was to surpass the performance of the RACER algorithm. Table 2 compares ROPAC (specifically, ROPAC-L and ROPAC-M) and RACER across all datasets. The best accuracy for each row is highlighted in bold.

A striking observation is that ROPAC surpasses RACER in terms of mean accuracy. Notably, ROPAC-L achieves 82.69%, while RACER reaches only 80.07%. This remarkable difference of 2.62 percentage points underscores the significant improvement ROPAC offers in this domain.

ROPAC outperforms in a resounding majority of datasets (43 out of 50). The top 5 datasets that witnessed the most significant improvements include "baskball" (14.66% accuracy gain), "tae" (13.92% gain), "liver-disorders" (9.28%–9.57% gain), "ecoli" (7.96%–8.26% gain), and "led7" (7.94%–8.09% gain). These results highlight ROPAC's versatility in handling diverse datasets, ranging from the low-dimensional "baskball" to the high-dimensional "led7". Additionally, in four of the remaining datasets (including "dbworld-bodies" and "mux6"), the accuracy obtained by RACER and ROPAC is equal.

Table 3
Accuracy comparison of classifiers.

#	ROPAC-L	ROPAC-M	ForestPA	LMT	MLP	Random Forest	Optimized Forest	SPAARC	RACER	Bagging	C4.5	PART	JRip	SMO	CART	IBk	Naïve Bayes
1	83.67	83.67	84.62	76.92	**84.62**	82.69	82.69	76.92	79.67	82.69	73.08	76.92	80.77	**84.62**	75	76.92	82.69
2	**87.62**	**87.62**	85.94	81.25	81.77	82.81	82.81	84.38	**87.62**	82.81	79.69	81.25	84.38	87.5	79.69	59.38	75
3	89.11	89.11	83.78	**94.59**	90.54	87.84	87.84	81.08	86.43	81.08	82.43	87.84	83.78	90.54	86.49	87.84	86.49
4	75	73.75	72.37	65.79	71.05	**76.32**	75	72.37	71.25	75	68.42	69.74	67.11	53.95	60.53	71.05	69.74
5	**76.11**	**76.11**	75.90	73.49	72.29	71.08	69.88	69.88	72.5	68.67	72.29	73.49	71.08	72.29	72.29	69.88	73.49
6	74.22	74.22	71.88	**76.04**	69.79	67.71	67.71	67.71	59.56	64.58	69.79	68.75	67.71	70.83	67.71	60.42	71.88
7	90	84	87	**91**	86	89	87	88	83	75	81	80	74	85	77	82	87
8	86	86	88	89	**90**	86	86	88	85	88	85	88	87	88	84	83	88
9	68	68	67.92	60.38	64.15	67.92	67.92	67.92	68	67.92	49.06	60.38	62.26	62.26	67.92	66.98	**69.81**
10	**64.29**	**64.29**	59.2	60	48.8	49.6	51.2	60	64.23	56	61.6	53.6	58.4	60.8	61.6	48.8	58.4
11	**100**	**100**	98.44	90.63	97.66	**100**	**100**	78.13	**100**	89.84	90.63	98.44	**100**	67.97	82.03	98.44	57.81
12	75.77	72.69	79.55	81.06	76.52	81.06	80.30	82.58	69.67	**83.33**	82.58	81.06	81.06	**83.33**	**83.33**	72.73	79.55
13	77.14	76.43	72.14	75	**81.43**	75	75	76.43	76.43	77.86	77.14	75	62.86	74.29	75.71	68.57	74.29
14	83.19	79.86	80.41	83.11	84.46	83.78	83.78	77.70	82.43	76.35	77.03	76.35	77.7	**86.49**	72.3	82.43	83.11
15	57	57	60.26	53.64	54.3	**68.87**	67.55	56.29	43.08	56.95	59.6	55.63	41.06	54.3	53.64	62.25	54.3
16	76.69	76.69	77.91	75.46	**78.53**	76.07	76.07	77.91	71.07	74.23	74.23	76.69	74.23	72.39	71.78	73.01	74.85
17	92.83	92.87	**94.61**	91.62	93.41	92.81	92.81	93.41	91.58	91.02	91.62	94.01	92.81	93.41	89.82	92.22	92.81
18	87.32	87.34	88.72	86.15	90.77	92.82	92.31	86.67	86.29	88.72	80.51	81.03	87.69	87.18	85.64	**96.41**	69.23
19	85	87.5	89.5	87.5	88.5	**91**	**91**	88	83.5	85.5	87	88	87	85.5	81	83.5	71.5
20	97.62	98.1	96.17	**99.52**	97.13	96.17	96.65	95.69	97.14	96.17	97.61	95.69	95.22	95.22	93.3	94.74	95.22
21	91.43	90.48	89.52	**95.24**	**95.24**	94.29	94.29	90.95	88.1	92.86	91.9	92.86	90.48	93.81	90	94.29	91.43
22	89.76	89.33	88.09	87.66	88.94	87.66	88.51	**90.21**	88.04	88.51	89.36	84.26	85.96	86.38	88.09	83.4	87.66
23	82.59	82.59	82.59	83.33	78.15	81.48	80.74	78.89	79.26	79.26	76.67	73.33	78.89	**84.07**	76.67	75.19	83.7
24	57.34	58.42	55.24	**63.64**	45.8	50.7	50.35	56.99	53.53	53.5	54.9	52.1	55.94	**63.64**	57.69	47.55	60.49
25	83.67	84.34	82.65	**85.03**	**85.03**	81.97	81.97	77.55	81.68	79.25	80.95	80.95	78.91	82.65	77.55	76.87	83.67
26	**63.4**	61.39	53.80	56.44	53.14	55.12	54.46	52.48	61.04	55.45	54.79	53.47	58.42	57.1	51.49	50.5	57.76
27	79.88	79.23	81.52	82.51	77.56	81.85	82.84	78.22	79.55	81.52	76.57	78.55	76.57	**83.17**	77.56	75.58	82.51
28	**76.17**	74.85	75.82	73.86	69.28	68.95	69.28	72.88	76.16	72.22	72.88	69.61	72.88	73.53	73.53	68.3	76.14
29	81.95	82.25	88.07	**88.99**	86.85	86.54	87.16	82.26	73.99	85.32	83.49	84.4	85.93	84.1	82.57	82.26	87.46
30	68.41	68.7	69.86	66.38	71.59	73.04	**74.20**	66.38	59.13	69.57	68.7	63.77	64.64	58.26	64.06	62.9	55.36
31	**99.72**	92.9	97.54	97.81	96.17	95.9	96.17	93.99	99.71	93.17	93.99	94.54	86.89	95.36	91.53	94.54	97.27
32	**78.9**	76.24	75.49	76.46	68.69	74.27	74.51	77.18	78.43	75.97	76.46	74.27	75.24	77.91	76.46	66.02	70.63
33	**70.8**	67.93	67.70	70.1	65.31	67.7	67.94	67.46	69.62	70.33	66.99	66.99	65.55	70.1	65.79	60.77	68.66
34	95.15	95.15	95.63	**96.78**	94.71	96.09	96.09	95.40	94.48	95.63	96.32	94.71	95.4	96.09	95.4	92.41	90.11
35	83.4	83.19	**85.11**	84.68	80.21	83.4	83.40	**85.11**	83.19	83.83	84.47	79.57	84.89	84.89	84.68	77.23	78.51
36	**100**	99.8	99.61	98.83	97.66	99.61	99.61	95.90	99.61	99.02	97.85	99.02	**100**	80.08	95.9	99.8	80.66
37	83.74	84.13	83.52	84.29	**84.67**	83.33	83.14	82.95	83.94	83.72	81.42	82.18	82.18	82.76	81.61	80.46	83.52
38	96.83	96.48	94.55	97.19	96.13	95.96	95.96	93.50	95.25	95.25	93.32	93.5	92.62	**97.72**	92.27	95.96	92.97
39	72.73	73.73	68.72	76.54	**77.54**	44.43	45.92	74.71	67.22	48.42	62.23	61.56	57.07	43.26	49.92	50.25	45.59
40	75.82	75.55	**76.36**	75.95	73.64	74.86	75	75.41	75.54	71.2	74.46	75	72.01	75.27	70.38	69.84	65.63
41	72.4	70.05	73.05	**77.47**	75.39	75.78	75.39	74.48	66.93	75.78	73.83	75.26	76.04	77.34	75.26	70.18	76.3
42	92.84	92.84	95.47	95.89	94.84	96.74	**96.84**	94.84	88.32	95.37	95.79	95.37	95.37	84.21	94.21	96.32	70.84
43	92.7	92.7	92.60	92.28	91.66	92.7	92.81	91.76	**92.91**	92.28	91.35	90.51	90.93	91.87	92.28	91.35	90.62
44	**100**	**100**	**100**	99.9	98.77	**100**	**100**	99.90	99.9	**100**	99.9	**100**	**100**	78.83	98.87	**100**	81.4
45	81.51	81.7	82.08	82.55	79.74	80.86	81.14	82.93	82.17	82.55	82.08	81.33	82.46	**83.68**	82.74	80.58	80.02
46	**100**	**100**	80.52	53.02	94.66	69.75	67.97	59.79	**100**	73.31	92.62	91.01	48.49	46	58.19	59.34	40.21
47	66.73	66.67	69.59	69.31	67.35	67.82	66.60	70.88	65.17	**71.15**	68.97	66.46	70.88	67.14	69.72	59.67	65.78
48	98.78	98.97	99.09	**99.75**	99.34	99.09	99.09	99.31	99.22	99.06	99.44	99.06	99.19	95.43	99	96.28	87.89
49	71.59	71.44	**73.94**	73.66	73.56	73.13	73.28	73.16	63.5	72.88	73.34	73.56	69.16	73.56	72.97	73.31	73.16
50	99.71	**99.75**	99.63	98.99	99.73	99.07	99.05	99.58	99.63	97.34	97.05	99.21	96.84	93.08	95.96	98.38	90.32
AVG	**82.69**	82.12	81.83	81.53	81.26	81.01	80.95	80.08	80.07	79.91	79.85	79.77	78.52	78.34	78.06	77.2	76.23

Further analysis reveals that between the two variants of ROPAC, the Least Frequent Zero Bits variant outperforms the Most Frequent Zero approach. ROPAC-L achieves an overall average accuracy of 82.69%, while ROPAC-M trails slightly behind with an average accuracy of 82.12%. This difference of 0.57 percentage points exposes that the least frequent zero bits carry more discriminative information for most classification scenarios and could generally be preferred.

We also wanted to see how ROPAC performed compared to other popular data classification techniques, such as eager learners, lazy learners, and other classification algorithms, in terms of mean accuracy. Table 3 presents a comprehensive summary of these results. The evaluation included ROPAC variants, ROPAC-L and ROPAC-M, alongside fifteen other classifiers. These classifiers encompass a variety of approaches, including probabilistic methods (Naïve Bayes), support vector machines (SMO in SVM), instance-based learning (IBk algorithm of KNN), neural networks (Multilayer Perceptron (MLP)), ensemble methods (Bootstrap Aggregation (Bagging), Random Forest, and Optimized Forest), rule-based classifiers (JRip algorithm of RIPPER, PART, and RACER), and decision tree classifiers (C4.5, CART, ForestPA, LMT, and SPAARC).

According to Table 3, ROPAC-L outperforms every other algorithm in terms of average accuracy, ranging from 0.86% for ForestPA to 6.46% for Naïve Bayes. Similarly, ROPAC-M also demonstrated commendable performance, with higher average accuracy rates ranging from 0.29% for ForestPA to 5.89% for Naïve Bayes. This also applies when compared to other rule-based classifiers, such as JRip and RACER. While JRip achieves an accuracy rate of 78.52% and RACER gets an 80.07% gain, ROPAC-L and ROPAC-M have demonstrated impressive 82.69% and 82.12%, respectively.

Table 4 breaks it down by dataset perspective and shows that ROPAC-L outperforms the other classifiers across 12 datasets, either by a significant margin or by achieving similar results when considering the number of cases instead of accuracy. Algorithms such as Naïve Bayes, IBk, PART, Bagging, and JRip, on the other hand, consistently underperform the leading methods. Notably, Naïve Bayes and IBk have shown low accuracy in most datasets. Decision tree algorithms exhibit

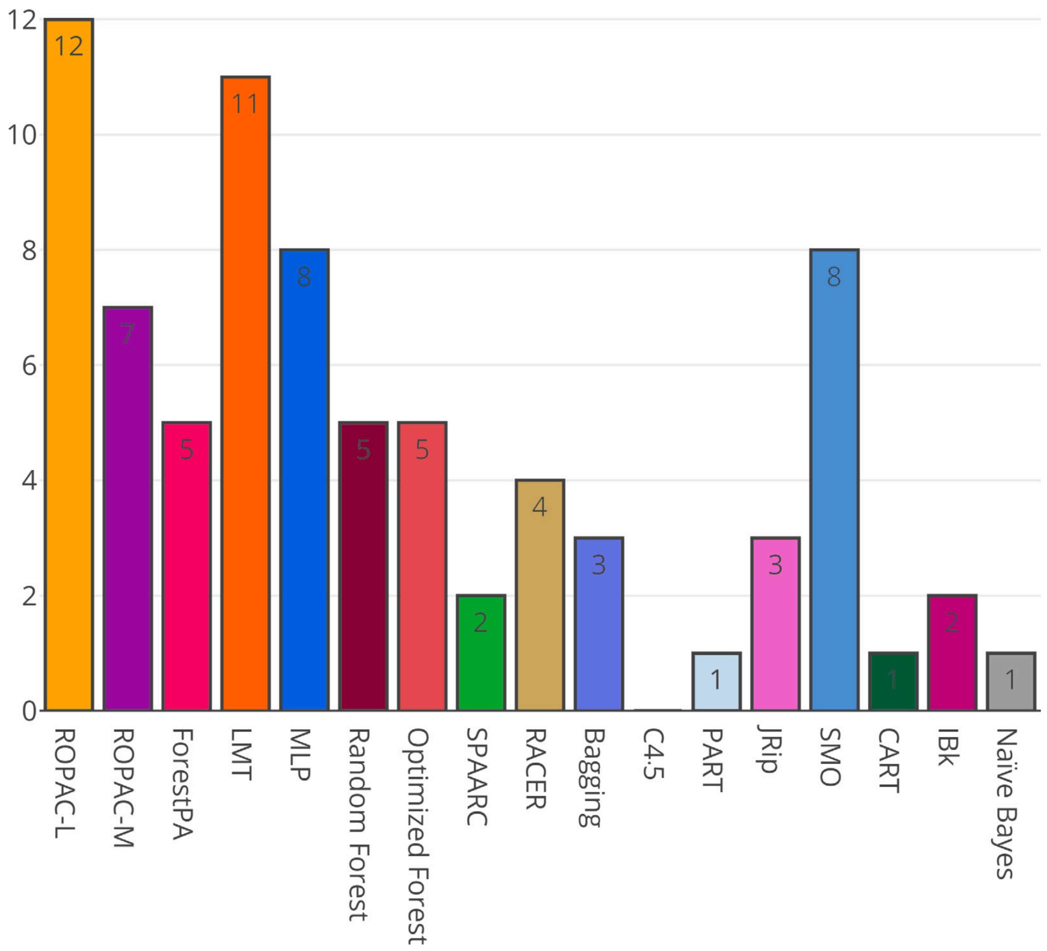

Fig. 2. Number of superiority or matching cases.

Table 4
Superiority or matching cases.

Algorithm	# of Superiority or matching cases
ROPAC-L	**12**
ROPAC-M	7
ForestPA	5
LMT	11
MLP	8
Random Forest	5
Optimized Forest	5
SPAARC	2
RACER	4
Bagging	3
C4.5	0
PART	1
JRip	3
SMO	8
CART	1
IBk	2
Naïve Bayes	1

considerable variation; traditional ones like C4.5 and CART tend to be outperformed by contemporary techniques like ForestPA, which perform better. This implies that conventional decision trees may be better suited to general-purpose applications, whereas ensemble methods may be superior in certain situations (Bennet et al., 2014).

In some datasets, like “mux6” and “xd6”, most algorithms achieve near-perfect scores due to their inherent linear separability. Despite this, ROPAC still stands out by matching the performance of the top methods, showcasing its capability to excel even in scenarios with linearly separable data. Fig. 2 presents a general summary of the comparison between different classifiers regarding the number of superiority or matching datasets.

Accordingly, Fig. 3 provides an overview by combining the metrics of average accuracy and the count of superiority or matching. The chart’s axis values have been scaled from the minimum to 0 and from the maximum to 10 to improve visualization and offer a more integrated understanding of the results. As anticipated, ROPAC-L continues to maintain its leading position in analysis.

When the overall performance of algorithms is compared, some patterns emerge, such as neural networks (MLP) frequently ranking high and lazy learners (IBk of KNN) ranking near the bottom. However, there is no single best algorithm, as situations vary by dataset (Decoux et al., 2023; Wu & Jhou, 2017). As a general matter, based on our experiments and the satisfactory results offered by ROPAC and ROPAC-L in particular, it can be claimed that this algorithm is a reliable choice for a wide range of classification tasks and scenarios.

Our research indicates that ROPAC’s outstanding performance is attributed to its unique rule optimization process. This algorithm distinguishes itself from other classifiers by introducing a novel methodology to expand search space and using an iteratively targeted approach to rule refinement. While conventional classifiers concentrate on a solitary feature or, at best, adopt a holistic perspective, they often overlook the crucial step of analyzing the initial rules and generalizing them correctly. ROPAC’s inclusive performance across datasets of varying sizes and characteristics demonstrates its adaptability to a broad spectrum of classification tasks. This versatility positions ROPAC as a strong contender for various real-world applications, regardless of the specific nature of the data.

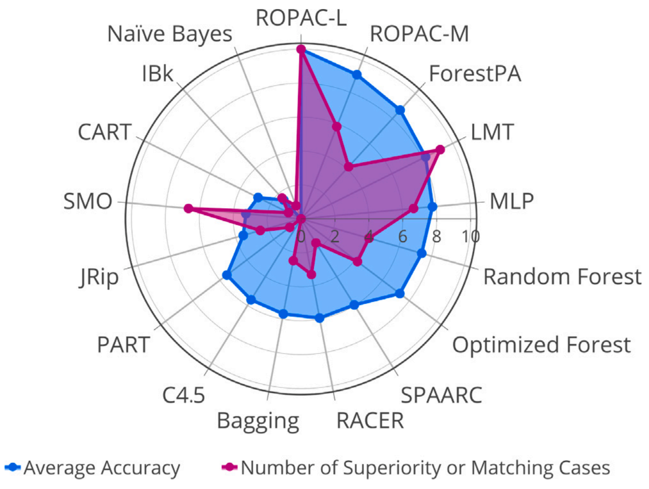

Fig. 3. Overall comparison of classifiers.

6. Conclusion

Rule OPtimized Aggregation Classifier (ROPAC) is a novel rule-based classification algorithm inspired by RACER that merges two innovative rule optimization techniques with a two-level rule generalization strategy. This paper extensively evaluates ROPAC on a diverse collection of fifty datasets. The objective was to demonstrate ROPAC's superiority over RACER, its improvement in multiple data scenarios, including high-dimensional and low-sample-size ones, and to assess its effectiveness and accuracy in comparison to a range of widely known data classification algorithms. The experimental results reveal that ROPAC's proposed enhancements enable it to surpass a multitude of popular algorithms, including Naïve Bayes, Multi-Layer Perceptron (MLP), Sequential Minimal Optimization (SMO), Instance-Based Learning (IBk), Bagging, JRip (RIPPER), PART, C4.5, Logistic Model Tree (LMT), Forest by Penalizing Attributes (ForestPA), Random Forest, Optimized Forest, Classification and Regression Trees (CART), SPAARC, and RACER, both in terms of average classification accuracy and number of superiority or matching cases.

Future studies on improving ROPAC can explore different avenues. One approach involves introducing a dynamic fitness function that could be fine-tuned during an iterative cycle of training processes. Additionally, exploring alternative rule optimization and initial rule generation techniques may lead to a more extensive rule set, potentially providing greater flexibility in exploring the search space. Also, studying other possible discretization techniques beyond the use of Information Gain may open up new possibilities. The pursuit of further methodologies for the rule composition phase also presents an intriguing direction to consider.

CRediT authorship contribution statement

Melvin Mokhtari: Conceptualization, Methodology, Software, Validation, Formal Analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Alireza Basiri:** Conceptualization, Methodology, Investigation, Resources, Data curation, Writing – review & editing, Supervision.

Declaration of competing interest

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

Data availability

The code and data supporting this study can be accessed via a CodeOcean capsule with the DOI http://dx.doi.org/10.24433/CO.7399708.v2 under the MIT license and Attribution (CC BY). The code is executable in Google Colab notebooks as well at http://colab.research.google.com/drive/1KrBPTdyYXqQqEslnS5UJ1oKho8sDfODJ. Updates will also be made available on the paper's GitHub page at http://github.com/MelvinMo/ROPAC-Rule-OPtimized-Aggregation-Classifier.

References

Adnan, N., & Islam, Z. (2016). Optimizing the number of trees in a decision forest to discover a subforest with high ensemble accuracy using a genetic algorithm. *Knowledge-Based Systems*, *110*, 86–97. http://dx.doi.org/10.1016/j.knosys.2016.07.016.

Adnan, N., & Islam, Z. (2017). Forest PA: Constructing a decision forest by penalizing attributes used in previous trees. *Expert Systems with Applications*, *89*, 389–403. http://dx.doi.org/10.1016/j.eswa.2017.08.002.

Al-Behadili, H. N. K., Ku-Mahamud, K. R., & Sagban, R. (2019). Annealing strategy for an enhance rule pruning technique in ACO-Based rule classification. *Indonesian Journal of Electrical Engineering and Computer Science*, *16*(3), 1499. http://dx.doi.org/10.11591/ijeecs.v16.i3.pp1499-1507.

Alcalá-Fdez, J., Alcalá, R., & Herrera, F. (2011). A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning. *IEEE Transactions on Fuzzy Systems*, *19*(5), 857–872. http://dx.doi.org/10.1109/tfuzz.2011.2147794.

Basiri, J., Taghiyareh, F., & Faili, H. (2019). RACER: accurate and efficient classification based on rule aggregation approach. *Neural Computing and Applications*, *31*(3), 895–908. http://dx.doi.org/10.1007/s00521-017-3117-2.

Battista, K., Diao, L., Patte, K. A., Dubin, J. A., & Leatherdale, S. T. (2023). Examining the use of decision trees in population health surveillance research: an application to youth mental health survey data in the COMPASS study. *Health Promotion and Chronic Disease Prevention in Canada*, *43*(2), 73–86. http://dx.doi.org/10.24095/hpcdp.43.2.03.

Bennet, J., Ganaprakasam, C. A., & Kannan, A. (2014). A discrete wavelet based feature extraction and hybrid classification technique for microarray data analysis. *The Scientific World Journal*, *2014*, 1–9. http://dx.doi.org/10.1155/2014/195470.

Bhattacharjee, R., & Manwani, N. (2020). *Lecture notes in computer science*, *Online algorithms for multiclass classification using partial labels* (pp. 249–260). http://dx.doi.org/10.1007/978-3-030-47426-3_20.

Bishop, C. M. (2007). Pattern recognition and machine learning. *Journal of Electronic Imaging*, *16*(4), Article 049901. http://dx.doi.org/10.1117/1.2819119.

Cendrowska, J. (1987). PRISM: An algorithm for inducing modular rules. *International Journal of Man-machine Studies*, *27*(4), 349–370. http://dx.doi.org/10.1016/s0020-7373(87)80003-2.

Cohen, W. W. (1995). *Fast effective rule induction* (pp. 115–123). Elsevier eBooks, http://dx.doi.org/10.1016/b978-1-55860-377-6.50023-2.

Decoux, A., Duron, L., Habert, P., Roblot, V., Arsovic, E., Chassagnon, G., Arnoux, A., & Fournier, L. (2023). Comparative performances of machine learning algorithms in radiomics and impacting factors. *Research Square*, http://dx.doi.org/10.21203/rs.3.rs-2677455/v1.

Dua, D., & Graff, C. (2017). *UCI machine learning repository*: *Technical report*, University of California, Irvine, School of Information and Computer Sciences, http://archive.ics.uci.edu/ml.

Fang, W., Gong, X., Liu, G., Wu, Y., & Fu, Y. (2020). A balance adjusting approach of extended belief-rule-based system for imbalanced classification problem. *IEEE Access*, *8*, 41201–41212. http://dx.doi.org/10.1109/access.2020.2976708.

Fang, S., Tu, Y., Kang, L., Chen, H., Chang, T., Yao, M., & Kuo, B. (2023). CART model to classify the drought status of diverse tomato genotypes by VPD, air temperature, and leaf–air temperature difference. *Scientific Reports*, *13*(1), http://dx.doi.org/10.1038/s41598-023-27798-8.

Frank, E., & Witten, I. H. (1998). *Generating accurate rule sets without global optimization* (pp. 144–151). University of Waikato, Department of Computer Science, http://hdl.handle.net/10289/1047.

Fuchino, T., Harada, T., Tanaka, K., & Mikawa, K. (2023). Computational complexity of allow rule ordering and its greedy algorithm. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *E106.A*(9), 1111–1118. http://dx.doi.org/10.1587/transfun.2022dmp0006.

Fürnkranz, J., & Widmer, G. (1994). *Incremental reduced error pruning* (pp. 70–77). Elsevier eBooks, http://dx.doi.org/10.1016/b978-1-55860-335-6.50017-9.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software. *SIGKDD Explorations*, *11*(1), 10–18. http://dx.doi.org/10.1145/1656274.1656278.

Hegde, H., Glurich, I., Panny, A., Vedre, J., VanWormer, J. J., Berg, R. L., Scannapieco, F. A., Miecznikowski, J. C., & Acharya, A. (2022). Identifying Pneumonia subtypes from electronic health records using rule-based algorithms. *Methods of Information in Medicine*, *61*(01/02), 029–037. http://dx.doi.org/10.1055/a-1801-2718.

Hońko, P. (2019). Binary classification rule generation from decomposed data. *International Journal of Intelligent Systems*, *34*(12), 3123–3138. http://dx.doi.org/10.1002/int.22181.

Huynh, V. Q. P., Fürnkranz, J., & Beck, F. (2023). Efficient learning of large sets of locally optimal classification rules. *Machine Learning*, *112*(2), 571–610. http://dx.doi.org/10.1007/s10994-022-06290-w.

Jabba, A. (2021). Rule induction with iterated local search. *International Journal of Intelligent Engineering and Systems*, *14*(4), 289–298. http://dx.doi.org/10.22266/ijies2021.0831.26.

Kesavaraj, G., & Sukumaran, S. (2013). A study on classification techniques in data mining. In *2013 fourth international conference on computing, communications and networking technologies*. http://dx.doi.org/10.1109/icccnt.2013.6726842.

Koren, D., őrincz, L. L., Sándor, K., Kun-Farkas, G., Hegyes, B. V., & Sípos, L. (2020). Comparison of supervised learning statistical methods for classifying commercial beers and identifying patterns. *Journal of Chemometrics*, *34*(4), http://dx.doi.org/10.1002/cem.3216.

Le, T. A., Stahl, F., Gomes, J. B., Gaber, M. M., & Di Fatta, G. (2014). *Computationally efficient rule-based classification for continuous streaming data* (pp. 21–34). Springer eBooks, http://dx.doi.org/10.1007/978-3-319-12069-0_2.

Liu, R., Gegov, A., & Cocea, M. (2016). Complexity control in rule based models for classification in machine learning context. *Advances in intelligent systems and computing*, 125–143. http://dx.doi.org/10.1007/978-3-319-46562-3_9.

Lukáčová, J., & Maličká, L. (2022). *Real estate tax revenue and it's place in local government tax revenue*. Univerzita Pavla Jozefa Šafárika, Vydavatel'stvo ŠafárikPress eBooks, http://dx.doi.org/10.33542/spf22-0145-2-14.

Mao, L., Chen, Q., & Sun, J. (2020). Construction and optimization of fuzzy rule-based classifier with a swarm intelligent algorithm. *Mathematical Problems in Engineering*, *2020*, 1–12. http://dx.doi.org/10.1155/2020/9319364.

Morovatian, I., Basiri, A., & Rezaei, S. (2023). RUCIB: a novel rule-based classifier based on BRADO algorithm. *Computing*, http://dx.doi.org/10.1007/s00607-023-01226-1.

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, *2*(1), http://dx.doi.org/10.1186/s40537-014-0007-7.

Nayani, S., Rao, P. S., & Lakshmi, D. R. (2023). Optimized ensemble learning-based student's performance prediction with weighted rough set theory enabled feature mining. *Concurrency Computations: Practice and Experience*, *35*(7), http://dx.doi.org/10.1002/cpe.7601.

Omuya, E. O., Okeyo, G., & Kimwele, M. (2021). Feature selection for classification using principal component analysis and information gain. *Expert Systems with Applications*, *174*, Article 114765. http://dx.doi.org/10.1016/j.eswa.2021.114765.

Palliser-Sans, R. (2021). *RRULES: an improvement of the RULES rule-based classifier*. arXiv (Cornell University), http://dx.doi.org/10.48550/arxiv.2106.07296.

Paul, Y., & Kumar, N. (2019). *Lecture notes in electrical engineering*, *A comparative study of famous classification techniques and data mining tools* (pp. 627–644). http://dx.doi.org/10.1007/978-3-030-29407-6_45.

Pham, D. T., & Aksoy, M. S. (1995). RULES: A simple rule extraction system. *Expert Systems with Applications*, *8*(1), 59–65. http://dx.doi.org/10.1016/s0957-4174(99)80008-6.

Qiao, L., Wang, W., & Lin, B. (2021). Learning accurate and interpretable decision rule sets from neural networks. *Vol. 35*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 4303–4311). http://dx.doi.org/10.1609/aaai.v35i5.16555.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. http://dx.doi.org/10.1007/bf00116251.

Salzberg, S. L. (1994). *C4.5: programs for machine learning by j. ross quinlan* (pp. 235–240). Morgan Kaufmann Publishers, Inc., http://dx.doi.org/10.1007/bf00993309.

Sanz, J., Sesma-Sara, M., & Bustince, H. (2021). A fuzzy association rule-based classifier for imbalanced classification problems. *Information Science*, *577*, 265–279. http://dx.doi.org/10.1016/j.ins.2021.07.019.

Stahl, F., Gaber, M. M., & Salvador, M. M. (2012). *ERules: a modular adaptive classification rule learning algorithm for data streams* (pp. 65–78). Springer eBooks, http://dx.doi.org/10.1007/978-1-4471-4739-8_5.

Thanakiattiwibun, C., Siriussawakul, A., Virotjarumart, T., Maneeon, S., Tantai, N., Srinonprasert, V., Chaiwat, O., & Sriswasdi, P. (2023). Multimorbidity, healthcare utilization, and quality of life for older patients undergoing surgery: A prospective study. *Medicine*, *102*(13), Article e033389. http://dx.doi.org/10.1097/md.0000000000033389.

Thangaraj, M., & Vijayalakshmi, C. R. (2013). Performance study on rule-based classification techniques across multiple database relations. *International Journal of Applied Information Systems*, *5*(4), 1–7, http://www.ijais.org/archives/volume5/number4/432-0608/.

Toulabinejad, E., Mirsafaei, M., & Basiri, A. (2024). Supervised discretization of continuous-valued attributes for classification using RACER algorithm. *Expert Systems with Applications*, *244*, Article 121203. http://dx.doi.org/10.1016/j.eswa.2023.121203.

Vanschoren, J., Van Rijn, J. N., Bischl, B., & Torgo, L. (2014). Openml. *SIGKDD Explorations Newsletter*, *15*(2), 49–60. http://dx.doi.org/10.1145/2641190.2641198.

Wang, C., Wei, X., Yang, A., & Zhang, H. (2022). Construction and analysis of discrete system dynamic modeling of physical education teaching mode based on decision tree algorithm. *Computational Intelligence and Neuroscience*, *2022*, 1–11. http://dx.doi.org/10.1155/2022/2745146.

Wu, W., & Jhou, M. J. (2017). MVIAeval: a web tool for comprehensively evaluating the performance of a new missing value imputation algorithm. *BMC Bioinformatics*, *18*(1), http://dx.doi.org/10.1186/s12859-016-1429-3.

Wu, D., Wang, X., Su, J., Tang, B., & Wu, S. (2020). A labeling method for financial time series prediction based on trends. *Entropy*, *22*(10), 1162. http://dx.doi.org/10.3390/e22101162.

Yates, D., Islam, Z., & Gao, J. (2019). SPAARC: a fast decision tree algorithm. *Communications in Computer and Information Science*, 43–55. http://dx.doi.org/10.1007/978-981-13-6661-1_4.

Yin, R., Pan, X., Zhang, L., Yang, J., & Lu, W. (2023). A rule-based deep fuzzy system with nonlinear fuzzy feature transform for data classification. *Information Sciences*, *633*, 431–452. http://dx.doi.org/10.1016/j.ins.2023.03.071.