Assessing the Effectiveness of a Genetic Programming Algorithm in Supervised Classification Tasks

Daniel Soto
Université Marie et Louis Pasteur, FEMTO-ST Institute, DISC Department
Besançon, France
daniel.soto_forero@univ-fcomte.fr

and

Wilson Soto
Intitución Universitaria Politécnico Grancolombiano, ETIC Research Group
Bogotá, Colombia
wsoto@poligran.edu.co

Abstract

Evolutionary Algorithms (EAs) are population-based stochastic search methods that emulate the principles of natural selection and biological evolution. Over the years, EAs have been successfully applied to numerous classification problems across diverse application domains. This paper proposes an enhanced evolutionary algorithm for the automatic synthesis of classifiers in supervised learning scenarios, grounded in the Genetic Programming (GP) paradigm. Unlike conventional GP approaches, the proposed method evolves a Directed Acyclic Graph (DAG) representation for each training class, enabling flexible alignment of test instances with their most compatible class structure through a compact graph-based encoding. The evolutionary search is guided by a multiobjective fitness function that jointly optimizes two complementary criteria: the cumulative evaluation scores of test samples with respect to each class DAG, and the total count of misclassified instances across all classes. The proposed algorithm is benchmarked against fourteen established methods spanning classical machine learning, ensemble approaches, deep learning, and evolutionary computation techniques. Experimental results on twelve publicly available UCI benchmark datasets demonstrate competitive classification performance, achieving accuracies above 70 % across all evaluated datasets while maintaining a low-complexity design that ranks the proposed algorithm among the top-performing methods in terms of average rank across the full benchmark suite.

Keywords: genetic programming, evolutionary algorithms, supervised classification.

1. Introduction

The genetic programming is a type of evolutionary algorithm inspired by the Darwin's theory of biological evolution [1] and belongs to the branch of the machine learning. These algorithms are oriented for solving optimization problems, where the goal is to evolve a set of individuals for finding an approximation solution.

A classical genetic algorithm includes the following phases: initialization, crossover, mutation, evaluation, and selection. First, the initialization phase consists of generating determinate individuals where each one contains random information. Second, the crossover phase generates new individuals with information mixed from two or more individuals. Third, the individuals can randomly change information due to the mutation phase. In the evaluation phase, individuals are assigned a value based on an evaluation function (*fitness function*) [2, 3], which represents their ability to solve the problem. Finally, the individuals with the best value are selected [4].

The proposed algorithm is a genetic programming algorithm where each individual represents an arithmetic expression DAG. For creating the arithmetic expression DAG, the proposed algorithm requires a set of arithmetic or logical functions with different values of arity [5].

The proposed algorithm aims to classify instances into corresponding categories based on predefined features, a process known as supervised classification. This technique builds a model using a training set and a label set. The training set consists of input objects paired with output labels, which are used to classify new samples. The classification process attempts to predict a discrete target label. If there are two labels, the classification is binary; otherwise, it is multi-class [6].

The supervised classification is a technique used in several domains for detecting patterns or predicting failures. Many algorithms use this technique, such as Bayesian classification, support vector machines (SVM), k-nearest neighbor, decision trees, naive Bayes, neural networks, evolutionary algorithms, random forest, and boosting [7, 5].

The related work is provided in Section 2. The proposed algorithm is introduced in Section 3. Section 4 shows the experimental evaluation. Finally, Section 5 brings the conclusions and proposes future work.

2. Related Work

Genetic programming (GP) has emerged as a robust technique for addressing classification problems in machine learning. Its versatility allows it to be effectively applied across various domains, including optimization, automatic programming, and machine learning. GP can be implemented using different representations, such as tree-based, linear-based, and graph-based structures, each suited to different types of problems [8]. These diverse representations enable GP to tackle complex multi-category pattern classification tasks efficiently, as demonstrated in applications like intrusion detection systems [9]. By framing classification as an optimization problem, GP employs genetic operators to generate and evaluate new hypotheses [10]. This methodology has been successfully applied to numerous real-world challenges, such as credit scoring, bankruptcy prediction, and medical diagnosis [11], showcasing its efficacy and adaptability in solving intricate classification issues.

As demonstrated in [12], M4GP introduces a novel multiclass classification method using genetic programming to perform multidimensional feature transformations. The method transforms the feature space and uses a distance function for classification in the transformed space. The study explores a new program representation for multidimensional features and compares it to existing methods, assessing the use of a distance metric for classification against simpler techniques. M4GP demonstrates competitive test accuracies and produces concise models, indicating good scalability to large feature spaces. The method employs a stack-based program representation, various initialization sizes and dimensionalities, and three population selection methods: tournament selection, lexicase selection, and age-fitness Pareto selection. M4GP uses mutation and crossover operators from M3GP and a nearest centroid classification strategy, maintaining a Pareto archive to balance accuracy and complexity. The results show that M4GP outperforms previous GP-based multiclass classification methods, producing accurate and generalizable classifiers across various problems, including simulated genetics datasets.

In [13], the focus is on the limitations of traditional GP methods in multi-class classification. The research introduced a multidimensional GP approach that extends the traditional GP tree structure to handle multiple dimensions, enhancing its capability to address the complexities of multi-class problems. This innovative method shows promising results in various multi-class classification tasks, providing a robust alternative to conventional GP techniques. However, the approach has limitations, including its computational complexity and the challenge of balancing exploration and exploitation in large behavior spaces. The study also acknowledges that the approach struggles with class imbalance issues, which can lead to suboptimal solutions and larger evolved programs.

There are studies focused on specific unbalanced data sets, such as the one presented in [4]. The study proposed a novel fitness function in GP for data classification to handle the problem of unbalanced data. They applied this method to four benchmark datasets from the UCI machine learning repository [14]. The proposed technique achieved the best accuracy and AUC compared to other GP and SVM methods, demonstrating its efficiency. However, the approach has limitations, such as potential overfitting if the model is trained for too many iterations. Furthermore, the high computational cost and need for domain expertise in handling unbalanced data can be challenging.

Another relevant study focusing on unbalanced data sets is presented in [15]. This work proposes integrating machine learning and metaheuristic algorithms to optimize the PROAFTN classification method. The study highlights the benefits of using hybrid metaheuristics, such as Particle Swarm Optimization (PSO) and Differential Evolution (DE), to enhance PROAFTN's performance. However, it also notes limitations, including the complexity of parameter tuning and the computational cost associated with metaheuristics. The authors suggest future research should explore parallel computation and alternative weight determination methods to mitigate these issues.

In [2], the core focus is binary classification, with modifications to the standard algorithm that include

adding a node to all population trees to ensure binary responses. The research employs GP, feature engineering, and symbolic regression for binary classification tasks. The GP algorithm, implemented with the GPLAB Matlab tool, encompasses phases such as initialization, adaptation, evaluation, genetic operations, and completion. Performance is assessed using a specific objective function. The results demonstrate that the GP algorithm achieves perfect accuracy in simple problems and over 95 % accuracy in more complex problems. Convergence graphs illustrate the relationship between the objective function's performance and the program size across generations. Despite its efficacy, the study notes a limitation in the increased model size for complex problems, impacting resource usage.

In some cases, pre-processing improves classification, as demonstrated in [16], which explores an output-based transfer learning system using GP for document classification. The system employs a linear model that combines GP programs evolved from a source domain with new mutated GP programs to enhance accuracy in a target domain. Using GP programs from the source domain as features facilitates effective classifier training in the target domain. The linear model, combining multiple GP programs from the source domain, outperforms single GP programs directly obtained from the target domain. The GP system uses a tree-based representation with random constants and word frequencies as terminals and arithmetic and relational operators as functions, prioritizing classification accuracy as the fitness function. The evolved GP programs are combined using a hybrid particle swarm optimization and differential evolution technique, with random mutations enriching the feature set. Future research aims to evolve GP programs using general words across multiple domains and integrate GP with deep learning techniques to improve document classification performance.

Another technique proposed in [3] involves a hybrid GP and correlation-based evaluation method for feature construction and selection tasks. This paper introduces FCMFS, a feature construction and feature selection approach that constructs multiple features using GP before selecting effective ones. The study demonstrates that the multiple feature construction method (FCM) significantly improves classification performance compared to single feature construction (FCS). FCMFS maintains high classification performance with fewer features than FCM alone and outperforms FSFCM, which reverses the order of stages. FCMFS employs a filter-based multiple feature construction approach and a filter-based feature selection approach using a correlation-based evaluation method. Future work aims to enhance the GP algorithm and apply it to high-dimensional datasets, improving the synergy between feature construction and selection for better classification performance.

In the study in [17], an ensemble classification method using GP was developed to address the challenge of feature construction in skin cancer image classification. Their approach constructs new features from pre-extracted texture, color, frequency-based, local, and global features. The study demonstrates that FCM significantly improve classification performance compared to traditional methods and single classifiers. However, a limitation of this approach is its potential to fit the training data too closely, resulting in poorer performance on test data due to overfitting. Additionally, the reliance on pre-extracted features may constrain the model's flexibility and generalization.

Studies have demonstrated that GP-based classifiers can achieve comparable or superior accuracy to other machine learning algorithms when given similar training times [8].

Thus, GP algorithms have demonstrated remarkable efficacy in supervised classification, underscoring their prowess in exploration and adaptability. Their capacity to iteratively evolve intricate models makes them particularly well-suited for tasks demanding dynamic feature construction and selection. Recent studies outlined in this section exemplify how hybrid approaches combining GP can surpass conventional methods, resulting in substantial enhancements in classification accuracy and model efficiency.

3. Proposed Algorithm

Our approach utilizes a genetic programming methodology designed for effective supervised classification tasks. Central to this methodology is the generation of a numerical metric using sample features to distinguish clear and discernible groups within the dataset. This metric evolves iteratively through the refinement of arithmetic expressions derived from the features of training samples. This iterative process enables the algorithm to construct sophisticated decision rules that adapt intricately to the unique characteristics of the dataset.

During classification, our algorithm computes a numeric score for each new sample. A positive score indicates belonging to the trained class, while a negative score indicates non-inclusion.

The numeric score results from an arithmetic expression structured as a Directed Acyclic Graph (DAG), individually created for each class during the training phase and refined using test samples. This expression DAG exclusively comprises constants (π, e, φ) and arithmetic functions from the set S. Let $S = \{ f(x,y), g(x,y), h(x,y), k(x,y), p(x,y), \mu(x,y), \exp(x), \sin(x), \cos(x), \tan(x)\}$, where, $f(x,y) = x+y$, $g(x,y) = x-y$,

$h(x, y) = x \times y$, $k(x, y) = x/y$, $p(x, y) = x \hat{} y = x^y$, and $\mu(y, x) = \sqrt[y]{x}$. The definition of S simplifies the search space of the problem, ensuring efficient exploration and adaptation of the expression DAGs for accurate classification.

Classification of a new sample involves evaluating it against all generated DAGs and assigning it to the class corresponding to the DAG that yields the highest computed value. Each DAG, formed by an arithmetic expression tailored during training, aims to distinguish samples belonging to its respective class while excluding others. This process leverages an iterative refinement of arithmetic expressions using genetic programming techniques, where elements from the defined set S are strategically combined to optimize classification accuracy.

Algorithm 1 outlines our approach, detailing the steps of the genetic algorithm, including initialization, crossover, mutation, evaluation, and selection. It illustrates how these evolutionary processes, applied to the construction and evaluation of DAGs, enable effective supervised classification across diverse datasets.

Algorithm 1 Proposed Algorithm

Require:
T : Number of iterations, $T \in \mathbb{N}$
I : Number of individuals (population size), $I \in \mathbb{N}$
P : Probability of creation of new individuals, $P \in \mathbb{R}_+$
Q : Probability of crossover in the population, $Q \in \mathbb{R}_+$
α_0 : Weight of function F (Eq. 2), $\alpha_0 \in \mathbb{R}_+$
α_1 : Weight of function E (Eq. 4), $\alpha_1 \in \mathbb{R}_+$
β_0 : Weight of misclassification error, $\beta_0 \in \mathbb{R}_+$
β_1 : Weight of correct classification, $\beta_1 \in \mathbb{R}_+$
Ensure:
$\mathcal{M}$: Mathematical expression

```
Initialization                                          ▷ See Subsection 3.1
i ← 0
while i ≤ T do
    if X_[0,1) ≤ P then
        Creation of new individuals
    end if
    Crossover(Q)                                        ▷ See Subsection 3.2
    Evaluation(α0, α1, β0, β1)
    while F = inf or F = NaN do                         ▷ F: fitness function
        Mutation                                        ▷ See Subsection 3.3
        Evaluation(α0, α1, β0, β1)                      ▷ See Subsection 3.4
    end while
    Selection                                           ▷ See Subsection 3.5
    i ← i + 1
end while
```

Genetic programming (GP) offers more interpretable models compared to standard machine learning models like neural networks, thanks to its symbolic structure. However, the aspect of interpretability in GP has only recently been considered with the rise of explainable artificial intelligence (XAI).

In [18], review the approaches aimed at enhancing model interpretability in genetic programming. These approaches are categorized into two groups: intrinsic interpretability, which focuses on directly evolving interpretable models, and posthoc interpretability, which uses GP to explain black-box models or to simplify GP-evolved models using linear models.

A key aspect of the study is the incorporation of parsimony pressure in the fitness function to prevent "bloat,"where models grow excessively without corresponding improvements in accuracy. The augmented fitness function combines the original fitness metric (e.g., mean squared error) with a penalty term based on model size.

In this sense, our algorithm (Algorithm 1) includes a parameter α governs the balance between the fitness function $\mathcal{F}$ used to evaluate instances and the error function $\mathcal{E}$. Specifically, setting α_0 and α_1 within the range 1 and 10, where α_1 is typically 2 to 3 times greater than α_0, enhances accuracy. Additionally, β regulates the impact of classification outcomes, with β_0 set between 10 and 50 to prioritize correct classifications, and β_1 approximately 3 to 4 times smaller to penalize misclassifications.

3.1. Initialization

In this phase, individuals are initialized by creating arithmetic expression DAGs (Directed Acyclic Graphs) that incorporate all features of the training dataset. These DAGs are randomly structured to introduce diversity into the population. Initialization is a crucial step in genetic programming as it establishes the initial population from which solutions evolve. Randomly generating these initial DAGs ensures a broad exploration of the search space, which is vital for discovering effective classification models. Additionally, the algorithm includes a mechanism for generating new individuals, controlled by parameter P, to further facilitate exploration within the search space and enhance the algorithm's ability to find optimal solutions (see line 1 in Algorithm 1) [13].

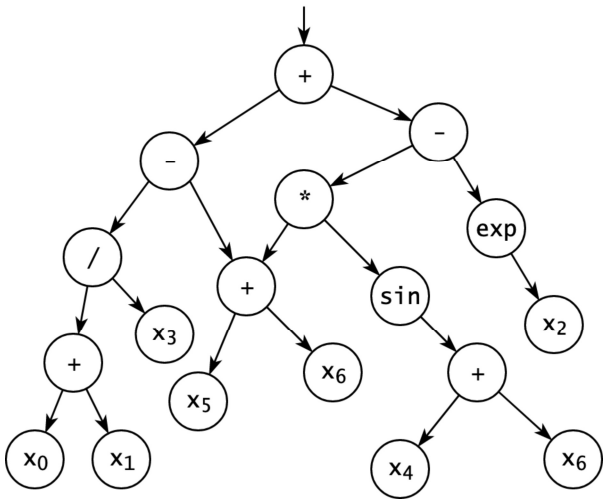

Figure 1: An Arithmetic Expression Directed Acyclic Graph (DAG) for $\left(\frac{(x_0 + x_1)}{x_3} - (x_5 + x_6)\right) + (((x_5 + x_6) * \sin(x_4 + x_6)) - (\exp(x_2)))$

3.2. Crossover

This phase involves the crossover operation, a fundamental genetic programming technique for generating new individuals. It begins by randomly selecting two individuals from the current population. A node is then randomly chosen from each individual's DAG (see Fig. 2(a) and 2(b)). These nodes act as the starting points for extracting sub-DAGs from each parent individual. The sub-DAGs are then exchanged between the parents to create two new offspring individuals by recombining them (see Fig. 2(c)). This crossover process ensures that the resulting expression DAGs maintain the diversity of features and potentially inherit beneficial traits from both parents, which aids in the exploration of the search space and the evolution of more effective classification models (see Fig. 2(d)).

The parameter Q determines the percentage of the population that undergoes the crossover stage in each iteration. This parameter is critical in controlling the genetic diversity of the population by specifying how many individuals will participate in creating new offspring through the crossover process (see line 7 in Algorithm 1).

3.3. Mutation

The mutation phase introduces random changes to an individual's expression DAG to maintain genetic diversity and explore new areas of the search space. In this step, an operator within the arithmetic expression is altered, with the mutation constrained to operators of the same type to preserve the structural integrity of the DAG (see line 10 in Algorithm 1). For example, if a particular node in the DAG represents an addition operation, it may be randomly replaced with another operation. This phase is triggered particularly when the arithmetic expression DAG produces errors during evaluation, such as resulting in undefined values like `inf` or `NaN`. This serves as a protective mechanism to prevent the propagation of invalid or poor-performing individuals in the population, ensuring that only functional and effective expressions are retained and refined.

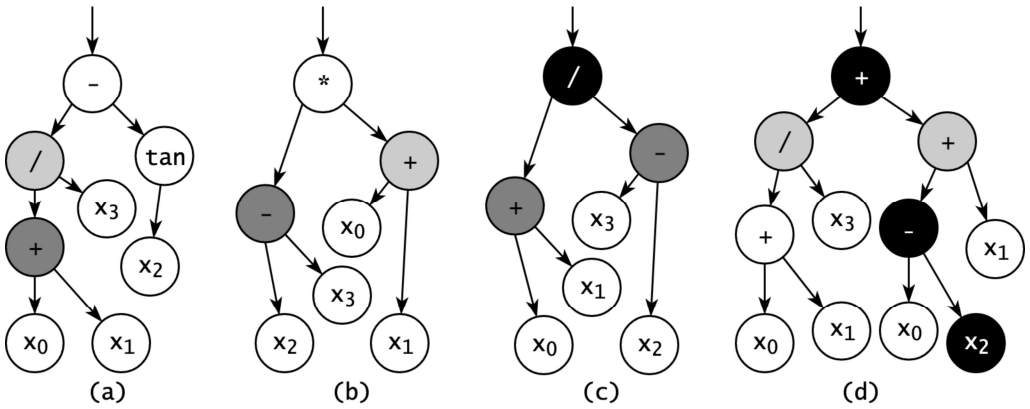

Figure 2: Crossover phase. (a) Individual for $((x_0+x_1)/x_3)-\tan(x_2)$. (b) Individual for $(x_3-x_2)*(x_0+x_1)$. (c) New individual $(x_0+x_1)/(x_3-x_2)$ formed by mixing (a) and (b) individuals at the dark gray node. (d) New individual $((x_0+x_1)/x_3)+((x_0-x_2)+x_1)$ formed by mixing (a) and (b) individuals at the clear gray node. Black nodes represent new nodes resulting from the combination of individuals (a) and (b).

3.4. Evaluation

The evaluation phase is crucial for assessing the performance of individuals within the population. This phase involves computing the result based on the input variables and the expression defined by the DAG, using the formula provided in equation (1). In our approach, we employ a multiobjective evaluation strategy that balances two key objectives: the sum of the evaluation values of the test samples (as described by equation (2)) and the count of classification errors (outlined in equation (4)). This multiobjective approach is encapsulated in a weighted assessment function that combines these terms, allowing us to quantify the effectiveness of each individual based on both accuracy and error metrics. Additionally, equation (6) specifies how to determine the membership of a sample j to a specific class C (see lines 8 and 11 in Algorithm 1). This step ensures that each individual's performance is thoroughly evaluated, considering both their predictive accuracy and error rates, which guides the evolution of more effective classification models.

$$D(v_j) = 1 + \text{distance } (Y_C(v_j), 0) \tag{1}$$

$$\mathcal{S}(C, v) = \sqrt{\frac{1}{M}\sum_{j=1}^{M}\begin{cases}\beta_0 * D(v_j) & \text{if } \mathcal{K}(v_j, C) = 1\\ \beta_1 / D(v_j) & \text{if } \mathcal{K}(v_j, C) = 2\end{cases}} \tag{2}$$

$$\mathcal{K}(v_j, C) = \begin{cases}1 & \text{if } (ce_j = C \wedge Y_C(v_j) \leq 0) \ \vee \ (ce_j \neq C \wedge Y_C(v_j) \geq 0)\\ 2 & \text{if } (ce_j = C \wedge Y_C(v_j) > 0) \ \vee \ (ce_j \neq C \wedge Y_C(v_j) < 0)\end{cases} \tag{3}$$

$$\mathcal{E}(C) = \sum_{i=1}^{M}(\mathcal{K}(v_i, C) - 1) \tag{4}$$

$$\mathcal{F} = \min \ (\alpha_0 \mathcal{S}(C, v) + \alpha_1 \mathcal{E}(C)) \tag{5}$$

$$co_j = \text{argmax } (Y_\rho(v_j)), \ \forall \rho \in C \tag{6}$$

Here, D represents the class separation function, which is used to measure how well different classes are distinguished by the model. The parameter M denotes the total number of instances in the dataset, while Z represents the number of features. The variable C indicates the value associated with a specific class, and v refers to an individual instance within the matrix $M \times Z \in \mathbb{R}$, which contains the feature values of all instances. $Y_C(v_j)$ denotes the value of the j-th feature for instance v in class C. The term ce_j is the expected class value for the j-th sample, and co_j represents the obtained class value for the j-th sample. These variables are integral to evaluating the performance of the classification model by comparing predicted class values with actual values and assessing class separation across the dataset.

3.5. Selection

The selection phase is crucial for ensuring that the most fit individuals are retained for subsequent generations. In our approach, we employ an elitist tournament selection method. This involves randomly selecting two individuals from the population and comparing their fitness values. The individual with the lower fitness value is deleted, while the one with the higher fitness value is retained. This process is repeated iteratively until the population size is reduced to the desired number, as specified by the variable I [13] (see line 13 in Algorithm 1). This method ensures that only the best-performing individuals are preserved, thereby maintaining the quality of the population and promoting the evolution of increasingly effective solutions over successive generations.

4. Experiments and Results

The experimental analysis has been done comparing the proposed algorithm with the supervised classification literature shows in the article [15]. The article has a set of 14 algorithms and 12 data sets. The data sets description are presented in table 1 and are available in UCI Machine Learning Repository [14].

Table 1: Summary of benchmark data sets used in the experimental evaluation.

	Data set	Instances	Features	Classes
1	BCancer	699	9	2
2	Blood	748	4	2
3	Heart	270	13	2
4	Hepatitis	155	19	2
5	HM	306	3	2
6	Iris	150	4	3
7	Liver	345	7	2
8	MM	961	5	2
9	Pima	768	8	2
10	STAust	690	14	2
11	TA	151	5	3
12	Wine	178	13	3

The proposed algorithm is benchmarked against fourteen established classification methods, spanning classical machine learning, ensemble approaches, deep learning, and evolutionary computation techniques [15]. The compared algorithms are as follows:

- **A1 – C4.5 (J48):** A decision tree induction algorithm based on information gain ratio, widely adopted as a baseline in classification benchmarks due to its interpretability and competitive performance on tabular data.
- **A2 – Naive Bayes:** A probabilistic classifier based on Bayes' theorem with a strong conditional independence assumption among features, known for its computational efficiency and effectiveness on small datasets.
- **A3 – SVM–SMO:** A Support Vector Machine trained via Sequential Minimal Optimization, capable of constructing high-dimensional hyperplane-based decision boundaries with strong theoretical generalization guarantees.
- **A4 – MLP:** A Multilayer Perceptron neural network trained with backpropagation, representing the classical feedforward architecture for supervised classification.
- **A5 – K-NN (k=3):** A non-parametric, instance-based learner that classifies samples by majority vote among the three nearest neighbors in the feature space, serving as a distance-based .
- **A6 – PART:** A rule-based classifier derived from partial decision trees, combining the interpretability of rule sets with the efficiency of decision tree construction.
- **A7 – Random Forest (500 trees):** An ensemble method based on bagging of decision trees with random feature subsampling, recognized as one of the most robust general-purpose classifiers for tabular data.

- **A8 – Generalized Linear Models (GLM):** A flexible extension of linear regression that accommodates non-normal response distributions through a link function, providing a statistically principled linear baseline.
- **A9 – Deep Learning:** A deep neural network architecture with multiple hidden layers, representing the current state of the art in feature learning from raw data and serving as a high-capacity nonlinear reference.
- **A10 – GA-PROAFTN:** A Genetic Algorithm applied to optimize the parameters of the PROAFTN multicriteria classification method, representing an evolutionary baseline for the family of PROAFTN-based approaches.
- **A11 – PSO-PROAFTN:** A Particle Swarm Optimization variant of PROAFTN, leveraging swarm intelligence to guide parameter search within the multicriteria classification framework.
- **A12 – DE-PROAFTN:** A Differential Evolution variant of PROAFTN, employing mutation and crossover operators over continuous parameter spaces to improve classification boundary estimation.
- **A13 – PSO-PROAFTN + RVNS:** A hybrid method combining Particle Swarm Optimization with Reduced Variable Neighborhood Search, designed to balance global exploration and local exploitation within the PROAFTN parameter optimization process.
- **A14 – DE-PROAFTN + RVNS:** A hybrid method combining Differential Evolution with Reduced Variable Neighborhood Search, extending the DE-PROAFTN approach with a local search intensification phase to refine solution quality.
- **A15 – Proposed Algorithm:** The evolutionary-based multi-class classifier introduced in this work, designed for low complexity, robustness to data heterogeneity, and competitive performance across diverse supervised classification scenarios.

This comparative framework covers a broad methodological spectrum, including decision tree induction (A1, A6), probabilistic models (A2), kernel methods (A3), neural architectures (A4, A9), instance-based learning (A5), ensemble methods (A7), linear models (A8), and a family of evolutionary PROAFTN-based approaches (A10–A14), providing a rigorous and representative basis for evaluating the contribution of the proposed algorithm.

The parameters of the proposed algorithm (A15) are shown in the table 2. The parameters of the remaining algorithms are the recommended in the article [15].

Table 2: Configuration of parameter settings used in the experimental evaluation.

T	I	P	Q	α_0	α_1	β_0	β_1
100	250	0.1 (10 %)	0.6 (60 %)	2.0	5.0	30.0	10.0

The algorithm addresses missing values in feature datasets using a technique known as *Imputation* [5], where missing values are replaced with the average of the available data points.

For experimental purposes, all algorithms were subjected to standard 10-fold cross-validation.

Table 3 presents accuracy comparisons across 12 datasets involving 15 algorithms. Our proposed algorithm achieved the second-highest average ranking, tying with algorithms A8 and A13. Specifically, our algorithm demonstrated superior accuracy on datasets such as Iris, Pima, and TA. Conversely, its lowest accuracy was observed on the Hepatitis dataset, showing a 24.32 % deviation from the highest accuracy achieved by algorithm A9.

In our comparative analysis, the proposed algorithm (A15) consistently achieves an accuracy rate surpassing 70 % across all datasets (refer to Table 3). Additionally, the performance of algorithms A10 to A15 underscores the efficacy of metaheuristic-based learning techniques in tackling supervised classification problems, thanks to their robust exploratory capabilities.

The figure 3 displays the boxplot comparing the percentage accuracy of the proposed algorithm. The plot reveals that the algorithm yields similar probability distributions across most datasets. However, notably larger standard deviations are observed for the `Hepatitis` and `Liver` datasets. This dispersion can be attributed to the complexity of the search space, which is influenced by the number of features in these datasets. As a result, our algorithm tends to allocate more time to the exploration phase

Additionally, it is evident that the number of instances does not significantly impact the overall behavior of the algorithm.

Table 3: Accuracy (%) of the proposed and benchmark algorithms across the experimental datasets. Bold values indicate the best result per dataset.

Dataset	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15
BCancer	94.56	95.99	96.70	95.56	97.00	97.05	97.40	**97.90**	**97.90**	96.76	97.14	96.97	97.33	97.05	96.98
Blood	77.81	75.40	76.20	78.74	74.60	**79.61**	76.10	74.90	78.70	75.43	79.25	79.59	79.46	**79.61**	78.95
Heart	76.60	83.70	84.10	78.10	78.89	73.33	57.60	60.40	54.90	71.95	86.04	84.17	**87.05**	85.37	84.18
Hepatitis	80.00	85.81	83.87	81.94	84.52	82.58	90.10	92.60	**94.80**	73.84	75.73	80.36	76.27	76.10	71.74
HM	71.90	74.83	73.52	72.87	70.26	72.55	73.10	69.20	67.20	83.85	84.27	83.74	**84.36**	83.81	77.52
Iris	96.00	96.00	96.00	97.33	95.33	94.00	95.30	96.70	90.70	96.57	96.21	96.47	96.30	96.66	**97.87**
Liver	68.70	56.52	58.26	71.59	61.74	63.77	71.80	73.00	**74.10**	71.83	69.31	71.01	70.97	70.99	71.36
MM	82.10	78.35	79.24	82.10	77.21	82.21	80.80	84.90	84.70	**84.92**	82.31	84.33	84.07	84.77	80.80
Pima	71.48	75.78	77.08	75.39	73.44	73.05	77.40	78.30	75.40	72.19	77.47	75.37	77.42	77.23	**83.81**
STAust	85.22	77.25	85.51	84.93	83.62	83.62	86.70	**88.90**	86.80	81.78	86.09	85.62	86.10	86.04	86.17
TAE	59.60	52.98	54.30	54.30	50.33	58.28	66.10	52.30	39.60	52.44	60.55	61.80	60.62	62.72	**71.25**
Wine	91.55	97.40	**99.35**	97.40	95.45	92.86	97.80	98.90	97.70	97.33	96.79	96.87	96.72	97.10	97.50
Avg. Rank	8.8	8.13	6.94	6.67	9.33	7.8	5.4	4.6	6.0	7.27	5.13	5.47	4.6	**4.4**	4.6
Num. Best	0	0	1	0	0	1	0	2	**3**	1	0	0	2	1	**3**

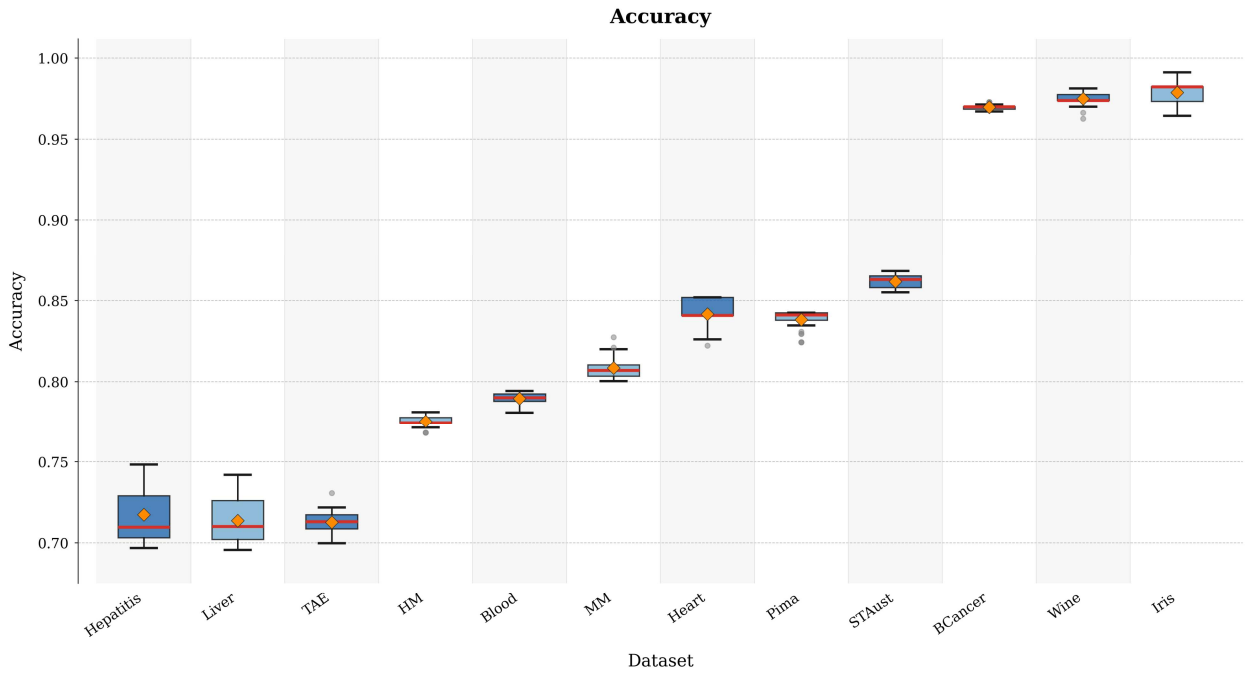

Figure 3: Box plots of Accuracy across 12 benchmark datasets (30 runs). Overall accuracy remains stable across most datasets, with BCancer, Wine and Iris reaching the highest median values and the narrowest interquartile ranges, indicating robust and consistent performance. Box: IQR (25th–75th percentile). Red line: median. Orange diamond (◊): mean. Whiskers: 1.5×IQR. Circles: outliers.

The most common measures used to compare performances are recall (sensitivity), specificity, precision, and F1-score (see Figures 4, 5, 6, 7, respectively).

Overall, the model correctly identifies a majority of the positive results in each dataset (sensitivity). Additionally, the trade-off between precision and recall (F1-score) is consistently high across all datasets.

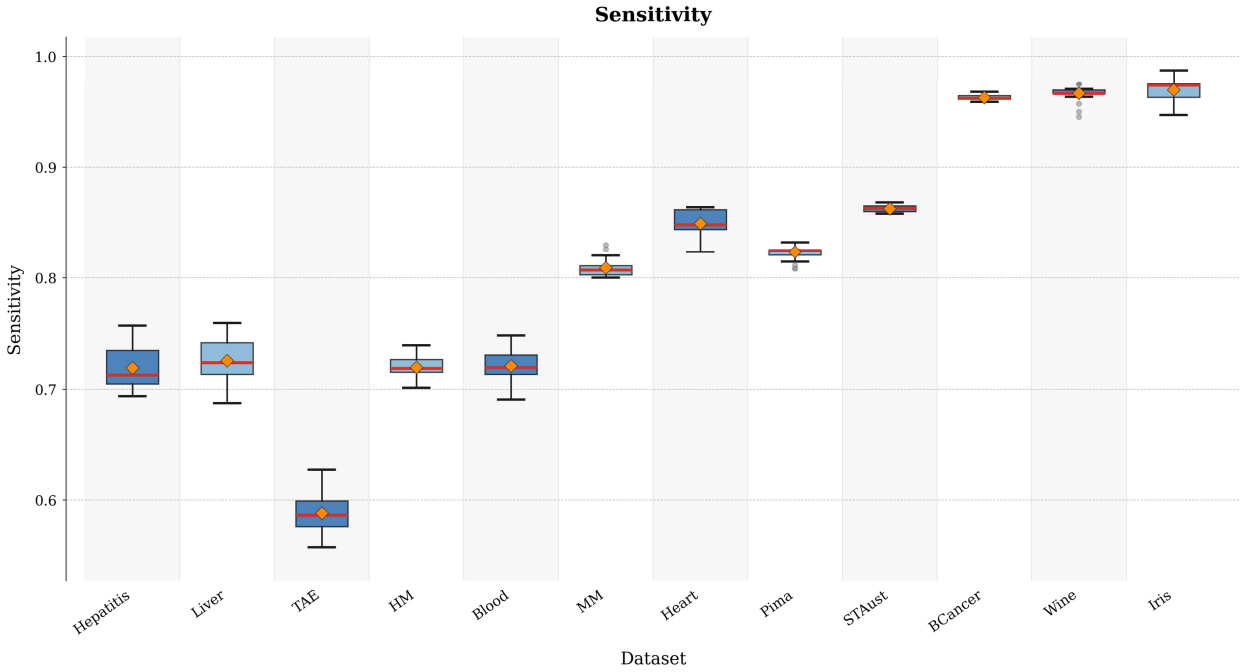

Figure 4: Box plots of Sensitivity across 12 benchmark datasets (30 runs). The results show higher sensitivity for smaller datasets such as Wine and Iris, while the TAE dataset reaches only about 0.6, indicating poorer detection of positive instances. Box: IQR (25th–75th percentile). Red line: median. Orange diamond (◊): mean. Whiskers: 1.5×IQR. Circles: outliers.

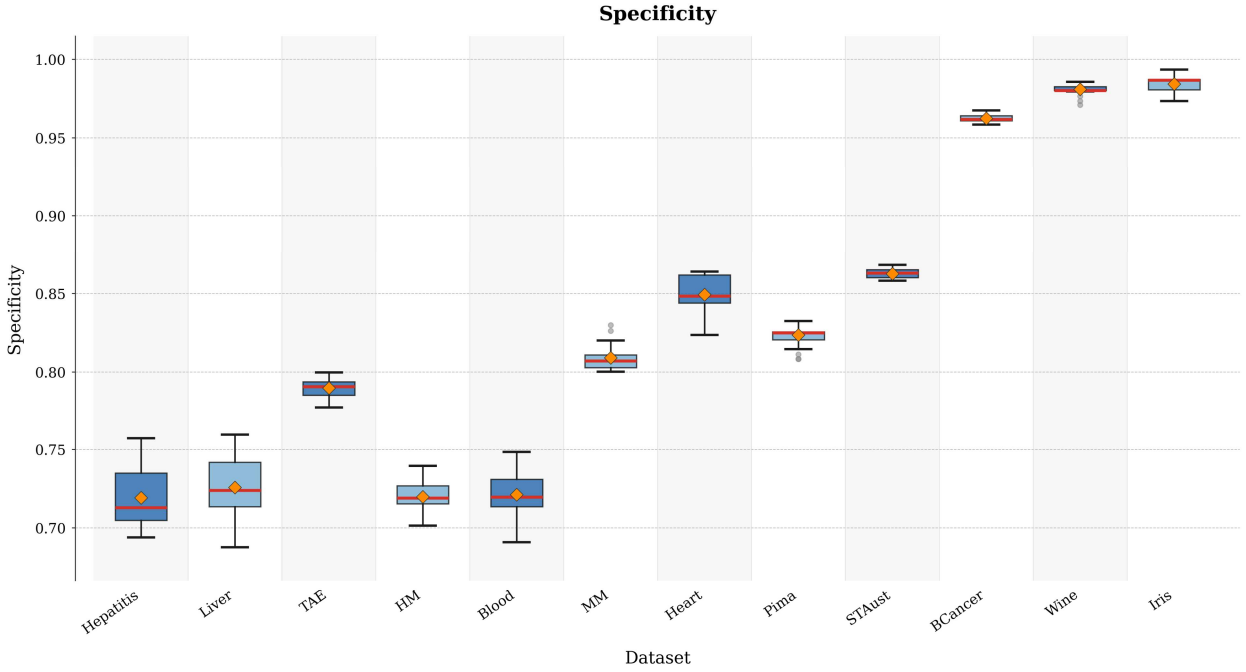

Figure 5: Box plots of Specificity across 12 benchmark datasets (30 runs). Datasets with balanced class distributions, such as BCancer and Wine, achieve the most stable specificity, whereas imbalanced datasets show wider interquartile ranges. Box: IQR (25th–75th percentile). Red line: median. Orange diamond (◊): mean. Whiskers: 1.5×IQR. Circles: outliers.

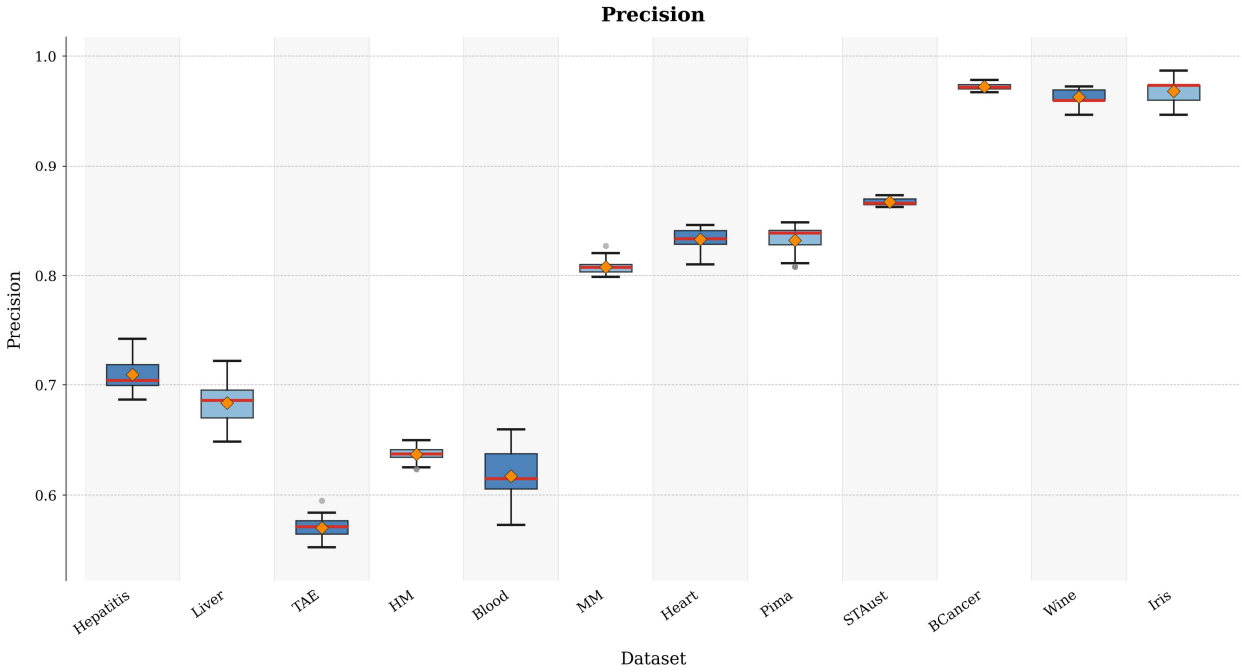

Figure 6: Box plots of Precision across 12 benchmark datasets (30 runs). Iris, Wine and BCancer consistently yield the highest precision scores, reflecting low false positive rates, while Blood and Liver present greater dispersion across runs. Box: IQR (25th–75th percentile). Red line: median. Orange diamond (◊): mean. Whiskers: 1.5×IQR. Circles: outliers.

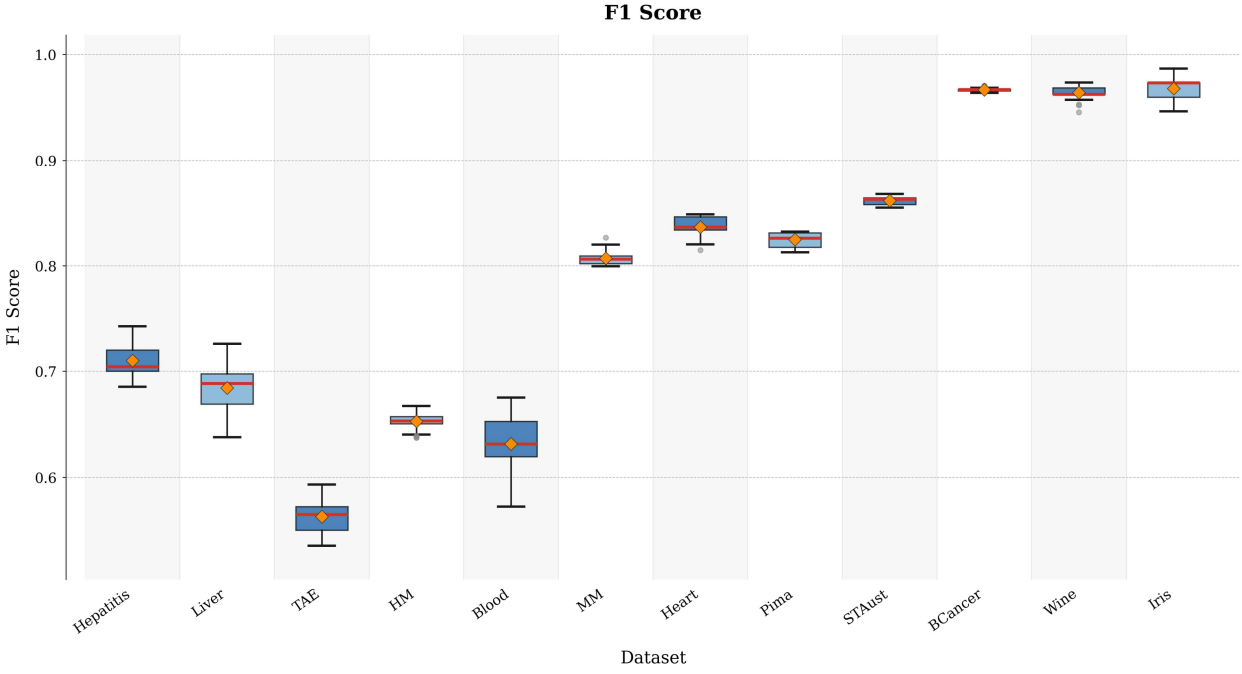

Figure 7: Box plots of F1 Score across 12 benchmark datasets (30 runs). The harmonic balance between precision and recall is highest for Iris, Wine and BCancer, whereas TAE, HM and Blood show lower median F1 values with wider spread, suggesting greater sensitivity to data partitioning in those datasets. Box: IQR (25th–75th percentile). Red line: median. Orange diamond (◊): mean. Whiskers: 1.5×IQR. Circles: outliers.

As the number of iterations increases, the confidence intervals progressively narrow and converge toward the mean, reflecting the stabilization of the classification process. This behavior indicates that the proposed algorithm consistently improves its predictive performance over successive generations, reducing result variability as the optimization advances. Specifically, the interquartile range (IQR) decreases with increasing iterations across most datasets, suggesting that the algorithm converges toward stable and reliable solutions

regardless of the underlying data characteristics. Datasets with larger sample sizes and well-separated class boundaries, such as `BCancer`, `Iris`, and `Wine`, exhibit the fastest convergence and the narrowest final IQR, demonstrating the algorithm's ability to exploit discriminative features efficiently. In contrast, datasets with overlapping class distributions or high feature noise, such as `Liver` and `TAE`, show slower convergence yet still achieve a measurable reduction in dispersion by the final generation. A notable exception is the `Hepatitis` dataset, which presents persistent variability across iterations without a clear convergence trend. This behavior can be attributed to the small sample size and high class imbalance inherent to this dataset, which introduce instability into the fitness evaluation and hinder the algorithm's ability to consistently identify optimal classification boundaries. These observations collectively suggest that the proposed algorithm is robust to dataset heterogeneity, achieving convergence in the majority of cases while highlighting the particular challenges posed by highly imbalanced and small-scale datasets (see Figures 8 and 9).

Table 4 summarizes the competitive profile of A15 against the fourteen benchmark methods grouped by methodological family. A15 shows consistent advantages across all groups, particularly on datasets with nonlinear boundaries, limited instances, and high noise, where classical, kernel, and neural methods are most constrained. Against the evolutionary PROAFTN family, A15 achieves comparable or superior results without domain-specific constraints or secondary search phases. Performance gaps occur only on datasets whose structure inherently favors the competing method, confirming that A15 provides a robust accuracy-complexity trade-off across a heterogeneous benchmark suite.

Table 4: General advantages and disadvantages of the proposed algorithm (A15) with respect to the benchmark method groups across 12 datasets.

Method Group	**Advantages of A15**	**Disadvantages of A15**
Classical ML *(A1, A2, A5, A6, A8)*	Outperforms classical methods on most datasets, especially those with nonlinear boundaries and feature interactions, where fixed inductive biases and linear assumptions structurally limit competing approaches.	Underperforms on datasets whose distribution directly suits the competing method, such as problems with strong feature independence or compact rule-representable class structures.
Kernel & Neural *(A3, A4, A9)*	Substantially superior on small and noisy datasets where kernel sensitivity, weight initialization instability, and overfitting constrain the performance of both shallow and deep neural architectures.	Kernel and neural methods achieve higher accuracy on larger datasets where sufficient training data allows gradient-based optimization and feature extraction to reach their full representational capacity.
Ensemble *(A7)*	Achieves competitive accuracy with a single evolutionary model, avoiding the computational overhead of constructing and aggregating hundreds of independent decision trees.	Random Forest's bagging mechanism provides a variance reduction advantage on high-noise datasets, where ensemble averaging consistently stabilizes predictions beyond what a single model can achieve.
Evolutionary PROAFTN *(A10, A11, A12, A13, A14)*	Achieves broader generalization without domain-specific multicriteria constraints, matching or exceeding hybrid two-phase methods with a simpler and more computationally efficient single-stage design.	PROAFTN-based methods retain an advantage on datasets structurally aligned with multicriteria classification frameworks, where domain-informed parameter constraints reduce the search space and improve convergence.
Overall	Consistent competitive performance across the full benchmark suite, outperforming the majority of methods on most datasets with a low-complexity single-stage evolutionary design.	Consistently challenged on imbalanced, and high-noise datasets, where the limited statistical support hinders reliable evolutionary convergence across all evaluated method families.

The appendix A presents the arithmetic expression for each class and the metrics of accuracy, sensitivity, specificity, precision, and F1-score for the best result achieved in each dataset.

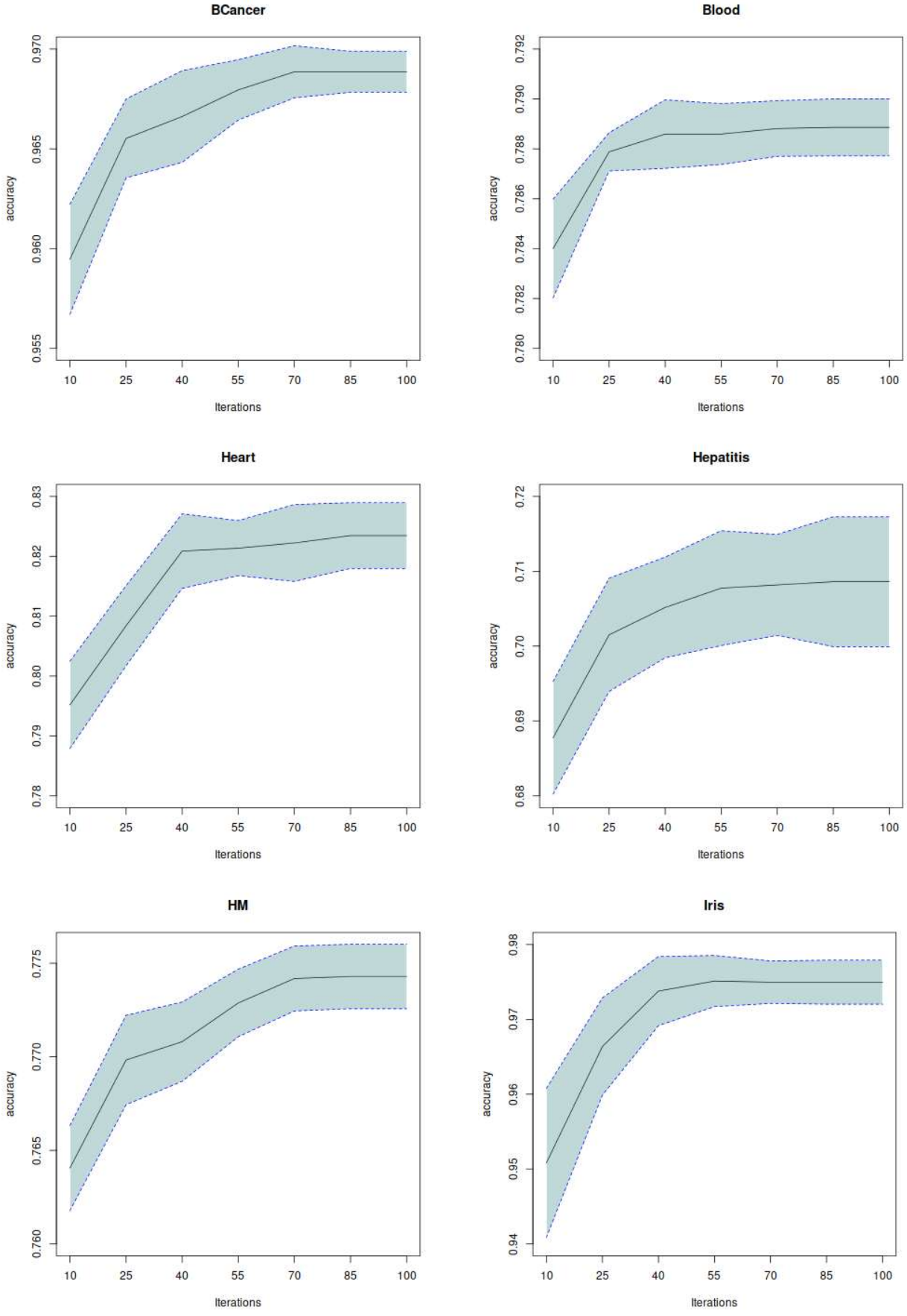

Figure 8: Accuracy convergence curves over 100 iterations for datasets `BCancer`, `Blood`, `Heart`, `Hepatitis`, `HM`, and `Iris`. Each curve represents the mean accuracy computed over 30 independent runs; the shaded region denotes the 95 % confidence interval, reflecting the variability of the evolutionary search across independent initializations.

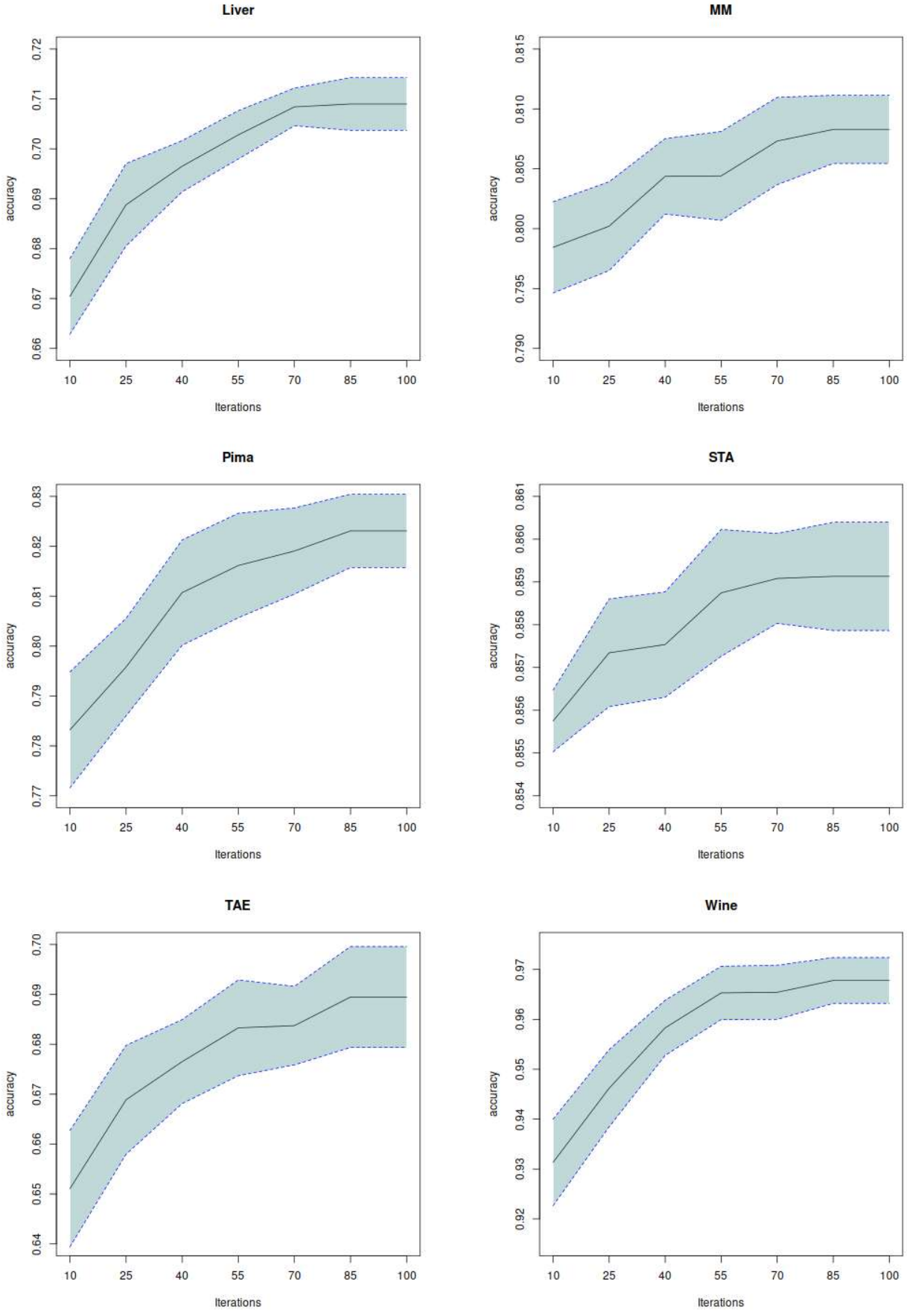

Figure 9: Accuracy convergence curves over 100 iterations for datasets `Liver`, `MM`, `Pima`, `STA`, `TAE`, and `Wine`. Each curve represents the mean accuracy computed over 30 independent runs; the shaded region denotes the 95 % confidence interval, reflecting the variability of the evolutionary search across independent initializations.

4.1. Classification Performance Analysis

The experimental results obtained across the 12 benchmark datasets reveal a consistent and structured performance pattern that reflects the intrinsic characteristics of each dataset and validates the effectiveness of the proposed evolutionary-based algorithm. Based on the observed accuracy, sensitivity, specificity, precision, and F1 score values, three distinct performance tiers can be identified, each associated with specific dataset properties.

4.1.1. Tier I: High-Performance Datasets (above 0.91)

The proposed algorithm achieves its highest classification performance on `BCancer`, `Wine`, and `Iris`, consistently surpassing the 0.90 threshold across all five evaluated metrics. The `BCancer` dataset, which comprises 699 instances and 9 features describing cytological characteristics of breast tissue, exhibits well-separated class boundaries and low feature noise, enabling the algorithm to identify highly discriminative classification rules with minimal variance across the 30 independent runs. Similarly, the `Iris` dataset, a widely adopted multiclass benchmark with 150 instances and 4 continuous features, presents linearly separable classes for two of its three categories, which the evolutionary search exploits efficiently through the weighted multiobjective fitness function. The `Wine` dataset, comprising 178 instances derived from chemical analysis of wines from three cultivars, benefits from high inter-class variability and low intra-class dispersion, conditions particularly favorable for population-based optimization. Across this tier, the narrow interquartile ranges observed in the box plot analysis confirm that the algorithm converges rapidly and reliably, producing stable solutions regardless of the random initialization of the evolutionary process.

4.1.2. Tier II: Moderate-Performance Datasets (0.81–0.90)

A second group of datasets – `MM`, `Heart`, `Pima`, and `STAust` – yields classification results consistently within the 0.81 – 0.90 range, reflecting a moderate level of classification complexity that the algorithm navigates effectively while exhibiting slightly greater result dispersion than Tier I. The `Heart` dataset, containing 270 instances and 13 heterogeneous clinical attributes related to cardiovascular disease diagnosis, introduces mixed feature types and moderate class overlap, requiring the algorithm to construct more nuanced decision boundaries. The `Pima` dataset, derived from medical records of 768 female patients for diabetes prediction using 8 physiological features, presents a notable class imbalance ratio and a non-trivial degree of feature correlation, both of which increase the complexity of the fitness landscape navigated by the evolutionary search. The `STAust` dataset, a statlog variant of the Australian credit approval problem with 690 instances and 14 attributes, combines continuous and discrete features with partially overlapping class distributions, challenging the algorithm's ability to generalize consistently. The `MM` dataset further illustrates the algorithm's capacity to handle moderate-complexity problems, maintaining competitive performance despite its structural heterogeneity. Across this tier, the box plot analysis reveals wider interquartile ranges compared to Tier I, yet the median values remain stable and the algorithm demonstrates a clear convergence trend as the number of iterations increases, confirming the robustness of the proposed evolutionary mechanism under moderately complex conditions.

4.1.3. Tier III: Challenging Datasets (0.70–0.80)

The most demanding classification scenarios are represented by `Hepatitis`, `Liver`, `TAE`, `HM`, and `Blood`, for which the algorithm achieves performance values in the 0.70 – 0.80 range. These datasets share several characteristics that intrinsically limit classification performance: small sample sizes, significant class imbalance, high feature noise, and substantial intra-class variability. The `Hepatitis` dataset, with only 155 instances and 19 features, several of which contain missing values, presents a particularly adverse learning scenario in which the fitness evaluation is inherently unstable, contributing to the persistent result variability identified in the convergence analysis. The `Liver` dataset (345 instances, 7 features) is characterized by weak discriminative signals and overlapping class distributions, limiting the ability of any classifier to consistently distinguish between positive and negative cases. The `TAE` dataset, a teaching assistant evaluation benchmark with only 151 instances distributed across three ordinal classes, suffers from low statistical support per class and subjective labeling, both of which introduce noise into the evolutionary training process. The `HM` and `Blood` datasets present analogous challenges related to feature redundancy, class distribution skewness, and limited instance counts, resulting in wider interquartile ranges and less consistent convergence behavior across independent runs. Nevertheless, the proposed algorithm maintains competitive performance relative to the compared baseline methods even under these adverse conditions, which underscores the generalization capability introduced by the stochastic search mechanism and the collective intelligence exploited through population diversity.

4.1.4. General Discussion

Taken together, the three-tier performance structure observed across the 12 benchmark datasets provides strong evidence for the consistency and adaptability of the proposed evolutionary algorithm. The results confirm that classification performance is primarily modulated by dataset-intrinsic factors, namely sample size, class balance, feature discriminability, and intra-class variability, rather than by limitations of the algorithmic design itself. The algorithm's low-complexity structure and weighted multiobjective fitness function enable effective adaptation across the full spectrum of dataset complexities, from the highly separable distributions of `Iris` and `BCancer` to the noise-prone and imbalanced conditions of `Hepatitis` and `TAE`. Furthermore, the consistent performance observed across both standard and normalized data representations reinforces the practical applicability of the proposed method in real-world scenarios where preprocessing conditions are heterogeneous and not always controlled. These findings collectively position the proposed algorithm as a reliable and computationally efficient solution for supervised multi-class classification across diverse application domains.

5. Conclusions

This study proposed an evolutionary-based algorithm for solving multi-class classification problems, systematically evaluated across a diverse set of benchmark datasets exhibiting heterogeneous characteristics, including varying numbers of instances, presence of missing data, and differing feature dimensionalities. The experimental results demonstrate that the proposed approach achieves competitive classification performance, attaining the highest accuracy in three datasets when compared against widely-used state-of-the-art algorithms. These results confirm the suitability of the proposed method for diverse supervised classification tasks across multiple application domains.

The effectiveness of the algorithm can be attributed to four complementary mechanisms: efficient utilization of training information, exploitation of collective intelligence through population-based search, inherent algorithmic stochasticity enabling exploration of complex solution spaces, and a weighted multiobjective fitness function that balances competing classification objectives. A particularly relevant finding is the consistent performance observed across both standard and normalized data representations, demonstrating the algorithm's robustness to preprocessing variations and reducing the dependency on extensive data preparation pipelines. Furthermore, the low-complexity design of the proposed method constitutes a practical advantage in resource-constrained deployment scenarios, distinguishing it from computationally intensive alternatives such as deep neural networks and large-scale ensemble methods.

Despite these contributions, several avenues for future research remain open. First, the function set will be extended to incorporate arithmetic, logical, and conditional operators of varying arity, broadening the expressive power of the evolved classifiers and potentially improving performance on high-dimensional and nonlinear datasets. Second, hybridization strategies combining the proposed evolutionary algorithm with both metaheuristic methods such as, particle swarm optimization and differential evolution; and exact optimization techniques will be investigated to further enhance solution quality and convergence speed. Third, future work will include a rigorous comparative evaluation against recent deep learning architectures, including convolutional neural networks and transformer-based models, to situate the proposed algorithm within the current landscape of data-driven classification methods and identify problem domains where evolutionary approaches offer a decisive advantage. Fourth, the applicability of the proposed method will be extended to real-world domains such as medical diagnosis, fault detection in industrial systems, and bioinformatics, where interpretability, low computational overhead, and robustness to class imbalance are critical requirements. Finally, advancements in hybrid evolutionary algorithms, particularly those integrating neuro-evolutionary strategies and automated machine learning (AutoML) frameworks, represent a promising direction for developing self-adaptive classifiers capable of jointly optimizing both model structure and hyperparameters without manual intervention.

References

[1] M. Ahvanooey, Q. Li, M. Wu, and S. Wang, "A survey of genetic programming and its applications," *KSII Transactions on Internet and Information Systems*, vol. 13, no. 4, pp. 1765–1794, April 2019.

[2] L. Santoso, B. Singh, S. Rajest, R. Regin, and K. Kadhim, "A genetic programming approach to binary classification problem," *EAI Endorsed Transactions on Energy Web*, 2020.

[3] J. Ma and X. Gao, "A filter-based feature construction and feature selection approach for classification using genetic programming," *Knowledge-Based Systems*, vol. 196, p. 105806, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095070512030191X

[4] A. Kumar, N. Sinha, and A. Bhardwaj, "A novel fitness function in genetic programming for medical data classification," *Journal of Biomedical Informatics*, vol. 112, 2020. [Online]. Available: https://doi.org/10.1016/j.jbi.2020.103623

[5] E. Dufourq and N. Pillay, "Hybridizing evolutionary algorithms for creating classifier ensembles," in *Sixth World Congress on Nature and Biologically Inspired Computing (NaBIC)*, 2014, pp. 84–90.

[6] J. Bell, *Machine Learning: Hands-on for Developers and Technical Professionals.* Indianapolis, USA: Wiley, 2020.

[7] U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu, and M. Stanley, "A brief survey of machine learning methods and their sensor and iot applications," in *8th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 2017, pp. 1–8.

[8] T. Loveard and V. Ciesielski, "Representing classification problems in genetic programming," in *Proceedings of the Congress on Evolutionary Computation*, vol. 2, 2001, pp. 1070–1077.

[9] *Genetic Programming for Machine Learning.* John Wiley & Sons, Ltd, 2017, ch. 6, pp. 183–216. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119136378.ch6

[10] S. Winkler, M. Affenzeller, and S. Wagner, "Advances in applying genetic programming to machine learning, focussing on classification problems," *Parallel and Distributed Processing Symposium*, p. 267, 2006.

[11] P. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *IEEE Transactions on Systems, Man, and Cybernetics Part C*, vol. 40, pp. 121–144, 2010.

[12] W. La Cava, S. Silva, K. Danai, L. Spector, L. Vanneschi, and J. Moore, "Multidimensional genetic programming for multiclass classification," *Swarm and Evolutionary Computation*, vol. 44, pp. 260–272, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210650217309136

[13] E. Naredo, L. Trujillo, P. Legrand, S. Silva, and L. Munoz, "Evolving genetic programming classifiers with novelty search," *Information Sciences*, vol. 369, pp. 347–367, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002002551630473X

[14] D. Dua and C. Graff, "Uci machine learning repository," 2019. [Online]. Available: https://archive.ics.uci.edu/ml

[15] F. Al-Obeidat, N. Belacel, and B. Spencer, "Combining machine learning and metaheuristics algorithms for classification method proaftn," in *Combining Machine Learning and Metaheuristics Algorithms for Classification.* Cham: Springer, 2019, pp. 53–79.

[16] W. Fu, B. Xue, X. Gao, and M. Zhang, "Output-based transfer learning in genetic programming for document classification," *Knowledge-Based Systems*, vol. 212, p. 106597, 2021.

[17] Q. U. Ain, H. Al-Sahaf, B. Xue, and M. Zhang, "A genetic programming approach to feature construction for ensemble learning in skin cancer detection," in *Genetic and Evolutionary Computation Conference (GECCO).* ACM, 2020, pp. 1186–1194.

[18] Y. Mei, Q. Chen, A. Lensen, B. Xue, and M. Zhang, "Explainable artificial intelligence by genetic programming: A survey," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 3, pp. 621–641, 2023.

A. Best Outcomes

Table 5: Best accuracy and the arithmetic expressions for each data set.

Data set	Accuracy	Sensibility	Specificity	Precision	F1
BCancer	97.28	96.64	96.64	97.43	97.02
$C_1 = ((x_6 + (x_2 - ((\tan((((\sin(0.931))\hat{}0)))) - (0.350076 + x_1)))) * (x_4 - (((x_0 * ((x_9 * (\mu(((x_2 * x_5) - 0.214286), x_8))) * (x_9 * (\mu(((x_2 * x_5) - 0.214286), x_8))))) + (1.06962 * 1.41667)) + ((\mu((\cos(0.473)), x_7))/x_3))))$ $C_2 = \mu(((x_2 - (((((((((x_0x_3)) + ((x_0x_3)))x_6)) - x_5)/x_1) - (x_8 + (\mu(0.922658, x_7))))) + (\mu(((x_0 - x_0) + x_4), x_9))), (x_8 - (((x_0x_3)) - x_6))$					
Blood	79.41	74.83	74.83	60.41	61.79
$C_1 = ((x_3 * x_0) * (\mu(x_0, ((x_3 * (\mu(0.655991, x_2))) - x_1))))$ $C_2 = ((x_2 - (\mu(x_0, x_3))) + ((x_2 * (\sin(x_1))) - x_0))$					
Heart	85.18	85.37	85.37	84.58	84.85
$C_1 = \mu((((((\mu((((\mu((((\mu((\tan((x_2+2.52941))), ((x_{11}\hat{}x_{11})))\hat{}(1.08405-x_{11})))+(\cos((1.29425/x_{12})))), (\sin((\sin(x_9))))))+(x_8/1.76471)) - (\tan(0.203))), ((((\sin((((((\mu((\mu(((((x_{11} - 0.130952)/0.513514) * (\cos(x_0))) * ((x_57))), (0.754545/x_3))), (((\mu((((\mu((((0.849254 + (\cos(x_{12})))\hat{}(2.79195 * 0.0166667))), x_3)) * (\mu((((0.849254 + (\cos(x_{12})))\hat{}(2.79195 * 0.0166667))), x_3))) * (3.30769 * x_1)) * ((\mu((((((1.35556x_7))/0.942857)/(\mu(x_{11}, 0.5))) - ((\mu((\tan(((((((\cos((\sin(x_8)))) * (x_0 + 0.606383))/(x_{12} + x_8)) - (((\cos((\sin(x_8)))) * (x0 + 0.606383))/(x_{12} + x_8))) - (\cos((\cos(x_9))))) - (((((((\cos(x_{12})) + (((0.36246x_{12})) + (x_2 * (x_8/1.28205)))) + (1.54513 * x_{11})) + (\sin((x_{10} + 1.65116))))/((((\cos(x_{12})) + (((0.36246x_{12})) + (x_2 * (x_8/1.28205)))) + (1.54513 * x_{11})) + (\sin((x_{10} + 1.65116))))) + ((x_9\hat{}x_{12}))/x_{12})))), (\mu((\cos(((x_1 + 0.0609756)/0.0526316))), ((x_7\hat{}(0.715286/1.34426)))))))) * (\cos(x_3)))), (((\sin((\tan(0.851))))\hat{}x_5)))) + (\mu(0.230248, x_2)))), (x_4 + x_{12}))) * (x_4 - 2.4)) * (\mu(x_7, x_{11}))))) + (0.319149/3.47619))/(0.370725 + 2.55556))/0.202899)/(\tan(0.707))))) * (x_8/0.489796)) * (\tan(0.096))) - (\cos(0.712))))) * (x_{11} - 3.875)) - (\mu(0.567568, 1.23256)))/(\mu(x_{10}, x_7))) - ((((\cos((\tan(0.82)))) * (\tan(x_2))) * ((x_{11}\hat{}x_4)))/(x_7 * 0.55))), (\mu(x_6, x_3)))$ $C_2 = (x_{11} + (((\tan(x_{12})) * ((x_6 + (x_3 + (x_2 * (x_1 + (((x_9 + (x_7 - (\cos(0.235)))) + x_8) - x_{10}))))) * x_4)) - x_0)) - (x_5/((\cos(0.235))/x_{12}))$					
Hepatitis	74.83	74.8	74.8	74.16	74.32
$C_1 = (x_8 - (((x_{15}/(\mu(((\mu(x_{13}, ((((x_7/(\mu(x_4, (x_{10} * (x_6 - (\mu(x_0, (\mu((0.0970378/1.60526), x_{16})))))))))\hat{}x_9)) - x_{14}))) * x_{17}), x_{18}))) - x_2) + ((x_1\hat{}(\mu(x_3, ((((0.297125 * x_{11}) + x_4) - x_{12}) * x_5))))))$ $C_2 = ((x_9 * (x_{18} * ((((((x_8\hat{}(\mu(x_{10}, ((((x_7 * (x_4 - (\cos(0.904))))/x_{12})\hat{}x_{13})))))))/x_{16}) * x_0)\hat{}x_2))))/((x_{10}/((\mu((((x_7 - (x_{12} * (\mu((x_{11}/(\tan(x_{15}))), x_{13})))) + x_{14}) - x_{18}), x_4)) * x_8)) - ((x_0\hat{}(\mu((((\mu(((((\tan(x_5))\hat{}x_6)), x_3))/x_{17}) * x_1), x_{16})))))$					
HM	78.1	73.95	73.95	63.38	65.05
$C_1 = (((x_1 - (\mu((\cos(0.119)), x_0))) - x_2) + ((0.13875/1.70833)/(x_1 - x_2)))$ $C_2 = ((\cos(1.61803)) - (x_2 * (x_1 - (x_0/0.542373))))$					
Iris	99.11	98.72	99.34	98.66	98.66
$C_1 = (((((\cos((2.16502-x_3)))/(\sin(x_2))) + (\mu(0.866279, 0.142857))) * (((\mu((\cos((\tan(0.481)))), (x_1/x_3))) + ((((\sin(x_2)) - (\mu(((x_2\hat{}7)), ((x_2\hat{}7))))))/((x_1 - 1.38235)/(x_0 * (\cos(x_2))))) - (x_1 * x_1))) - ((3.09948\hat{}5))))/(((x_2\hat{}0)) * 0.348837))$ $C_2 = (((\mu((\sin(x_1)), x_0)) - x_2)/((x_3 - x_0) + x_2))$ $C_3 = ((((x_3 * 5) - x_1) * (x_2 + (0.112513 - 0.507042))) + (x_0 * ((\tan(2.71828)) - x_1)))$					
Liver	74.2	74.19	74.19	72.16	72.58
$C_1 = (((\mu((x_2 * ((1.03188/x_4) * x_0)), x_5))/(\mu(((((\cos(0.351))/x_1)/1.32877)/x_2), x_4))) - x_3)$ $C_2 = ((x_4 * x_3) + (((\mu(x_0, (x_5 + ((x_4\hat{}(x_1 * ((1.131\hat{}1)))))))) * x_3) - x_2))$					
MM	82.73	82.62	82.62	82.68	82.65
$C_1 = ((0.508791/1.16418)/(\mu((x_3 + (((((\mu(x_0, 0.425287))\hat{}(\mu(x_4, (6.824/1.61803))))) - (\mu(((((0.492063\hat{}0))/x_1) + 1), (x_0 + 0.643678))))/(x_0 - 0.0843373))), x_2)))$ $C_2 = \mu(((1.69376 \hat{} (\mu((0.432265 + (\tan(x_3))), ((((0.388167 + x_0) - ((x_0 \hat{} 6)))/(0.184506 - x_2)) + (((((\cos(0.896))\hat{}5))/(\cos(0.324))) * ((0.0672269\hat{}x_0)))))))), (x_1 * x_3)))) + ((((\mu(((((x_0 + (\cos(x_0)))\hat{}(1.08921 + 29.6667))), (x_4 - 8))) * (\mu(0.131092, x_3))) * (((((\mu((\cos((\sin(0.492)))), (\mu((\cos(0.419)), (x_0/0.62))))) + (\tan(x_3))) - ((x_0\hat{}4)))\hat{}(x_0 * x_1)))) - (\cos((\tan(x_0))))$					
Pima	84.24	82.5	82.5	84.09	83.11
$C_1 = (((x_3 + (((x_4 - ((x_5\hat{}((x_0 * 0.54023)/x_2)))) - x_6)/x_1)) - x_7)/(x_4 - 0.169811))$ $C_2 = ((\mu(x_0, (x_2 + (x_7 - ((((x_3\hat{}(\cos(0.65))))/(x_1 * (\sin(x_4)))) - x_6)))))) + x_5)$					
STAust	86.81	86.7	86.7	87.12	86.75
$C_1 = (((((\mu(((((((((\tan(0.423))\hat{}x_9)) + x_1)\hat{}x_2))\hat{}((\mu(x_{10}, ((x_{11} * ((x_3/((0.311005 - x_{12}) + (0.311005 - x_{12}))) - x_6)) + x_5))) + x_0))), x_{13})) - x_8) - x_4)\hat{}x_7))$ $C_2 = ((x_7 - (x_{12}/((\mu(x_9, (x_{10}, (x_{11} * (\mu(((x_0 - x_6) + x_1), (((x_{13} * ((\mu((\tan(x_6)), x_4)) * x_2))\hat{}x_8)))))))) + x_5))) - x_3)$					
TA	73.07	59.6	79.94	59.48	59.31
$C_1 = ((\mu((\mu(0.915306, 0.8875)), (x_0 - 1.28889))) - (x_4 + x_3))/(\mu(0.325581, 0.236842))) * (\sin(x_2))) + ((((\mu(((x_0\hat{}6)), (0.410042-x_2))) + (\mu(x_4, x_0))) - (0.88342*0.308511)) - (\cos(x_3)))) * ((x_4 * (\tan(0.781)))/((x_3\hat{}(1.38086+x_1)$ $C_2 = (((\tan(x_2))/(\sin((((((((4.69444/3.77273)/x_3)/0.277778) + x_0) + x_2)\hat{}x_1))))) - x_4)$ $C_3 = (((\sin(0.686)) - ((x_3 + x_2) * x_0)) * (((x_1 * (\sin(0.215)))\hat{}((x_3\hat{}(((\mu(x_0, (x_2/0.457627)))\hat{}x_4)))))))$					
Wine	98.12	97.48	98.56	97.24	97.35
$C_1 = \mu((x_{10} * (((x_{11} - ((x_{12} + x_9) * x_8)) + x_5) + x_2)), (x_6 - ((\mu(((((x_{10}\hat{}(x_5 * (\mu(x_3, ((x_{11} * (x_1/2.09091))/x_7)))))))\hat{}x_0)), x_4))/x_{12})))$ $C_2 = (\mu((x_8/((((\mu(x_{11}, ((((x_{12} * x_{10}) - x_2) + x_7) + x_4)))\hat{}(\mu(((\mu(x_3, (x_5 - 0.808511))) * x_0), x_1))))/x_6)), ((0.23716x_{11})))) - x_9$ $C_3 = (x_{12} - ((((x_{11}/(x_3 * (\mu(x_1, ((\mu(((((x_0 + ((\cos(0.813))/x_4))/x_2) + x_{10}) - x_7), x_5)) - x_8)))))/x_9) * x_6) - (\tan(3.14161))))$					