SN Computer Science (2025) 6:510
https://doi.org/10.1007/s42979-025-04048-4

ORIGINAL RESEARCH

An Empirical Study of GM4OS for Imbalanced Binary Classification

Davide Farinati[1] · **Leonardo Vanneschi**[1]

Received: 25 June 2024 / Accepted: 13 May 2025

Abstract
Addressing the challenge of class imbalance in binary classification, this paper introduces Genetic Methods for OverSampling (GM4OS), an innovative technique leveraging the combined capabilities of Genetic Algorithms (GAs) and Genetic Programming (GP). Traditional oversampling methods like SMOTE and its variants depend on the selected data points and fixed synthetic data generation processes, often leading to suboptimal results. GM4OS advances this field by simultaneously evolving a resampling set and a synthetic data generation function. Individuals in GM4OS are made of two components, the GA component selects minority class observations for resampling, while the GP component evolves functions to create synthetic observations. This dual evolution process aims to optimize both the selection of data points and the creation of synthetic samples, enhancing the performance of classifiers on imbalanced datasets. We studied the performance of GM4OS across ten different test datasets and against five oversampling approaches commonly used in the literature. The results highlight how GM4OS is able to outperform the baseline methods in three out of ten test datasets, improving the algorithm performance.

Keywords Oversampling · Imbalanced data · Genetic programming · Genetic algorithms

Introduction

A common obstacle in real-world classification tasks is the disparity in the number of observations belonging to the different classes, this problem is often referred to as imbalance [1]. Classification models typically achieve higher accuracy when dealing with observations from the more common class, known as the majority class, compared to the less frequent class, called the minority class.

Common techniques to handle this issue involve the creation of synthetic observations and to label them as minority class, making the dataset balanced. The approach is called oversampling [1, 2]. There are different oversampling approaches, g.e., Synthetic Minority Oversampling Technique (SMOTE) [3], Borderline-SMOTE (BSMOTE) [4] and Adaptive Synthetic Sampling Approach (ADASYN) [5], that select pair of points from the minority class and use them as reference to sample a new synthetic observation. It's clear to understand how these methods rely on the selected points and on the function that creates the new synthetic observation.

In this paper, we introduce the Genetic Methods for OverSampling (GM4OS), a novel oversampling method that joins the representation power of two evolutionary algorithms, Genetic Algorithms (GAs) [6] and Genetic Programming (GP) [7], evolving at the same time a resampling set and a function that creates synthetic observations. In GM4OS, individuals are represented as pairs of objects. The first object is a function, represented as a standard GP individual, while the second is a string, represented as a traditional GA individual. The GA component determines which existing observations from the minority class will be included in the resampling set. The GP component evolves an oversampling function that combines points from the resampling set to create new synthetic points, balancing the dataset. The fitness of a GM4OS individual is measured by the performance of a model, trained using a pre-selected machine learning algorithm, on the newly balanced training set. The goal of the evolutionary process is to simultaneously find the best resampling set and the most effective oversampling function.

✉ Davide Farinati
dfarinati@novaims.unl.pt

Leonardo Vanneschi
lvanneschi@novaims.unl.pt

[1] NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisbon, Portugal

Published online: 30 May 2025

GM4OS was first introduced in [8]. This paper extends and completes the work, by introducing the following elements of novelty: new classification metrics as fitness functions and new models as classifiers for the fitness evaluation process of GM4OS.

The paper is organized as follows: "Literature Review" section provides an overview of existing methods used for imbalance binary classification tasks. "Genetic Methods for OverSampling" section introduces GM4OS. "Experimental Settings and Test Problems" section presents the experimental study. "Experimental Results" section shows and comments the obtained results. "Conclusion and Future Work" section concludes the paper and suggests ideas for future research.

Literature Review

Imbalance refer to classification problems in which one class is more represented than the others. In the literature, both internal and external approaches have been studied to alleviate the impact of this type of data on the model's performance. Internal approaches (algorithm level) refer to methods that modify the algorithm to handle imbalanced classification task. This includes a different range of approaches, e.g., assigning a weight to each class or using multiple classifiers simultaneously, also refereed as ensemble learning [9, 10]. External approaches (data level) refer instead to methods that modify the dataset making it balanced. This is either done by removing observations from the majority class (undersampling) or by adding new synthetic observations to the minority class (oversampling) [1, 2].

The Synthetic Minority Oversampling Technique (SMOTE) is an oversampling approach designed by Chawla et al. [3] in 2002. SMOTE balances the dataset by sampling a random point belonging to the minority class, finding a neighbor of the aforementioned point, and sampling a new observation along the segment. This synthetic point is then labeled as minority class and added to the dataset. This process is repeated until the dataset is balanced. Since borderline points are more important for classification, Borderline-SMOTE (BSMOTE) [4], identifies borderline minority instances and applies the standard SMOTE algorithm to them. The work by Douzas and Bacao [11] introduces Geometric-SMOTE (GSMOTE), extending SMOTE data generation mechanism. This method samples a random point from the minority class and then generates synthetic observation from the hyper-spheroid constructed around this given point. Another extension of SMOTE is the one presented by Nguyen et al. [12], that utilizes Support Vector Machines (SVM) [13] to estimate the borderline area. SVMSMOTE randomly creates instances along the lines joining each minority class support vector with a number of its nearest neighbors using the interpolation or extrapolation technique depending on the density of majority class instances around it. Another oversampling algorithm is Adaptive Synthetic Sampling Approach (ADASYN) [5], in which harder to learn minority samples are used for generating more synthetic data, compared with easier to learn observations. In ADASYN algorithm density distribution is used to automatically decide the number of synthetic samples that are needed to be generated for each minority class sample.

Generative models have gained significant attention in the field of imbalanced classification, particularly Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs). GANs, introduced by Goodfellow et al. [14], employ a two-network system comprising a generator and a discriminator that compete in a min-max game to generate realistic synthetic data. This adversarial training process enables GANs to learn complex data distributions, making them suitable for oversampling tasks in imbalanced datasets. VAEs, proposed by Kingma and Welling [15], take a probabilistic approach by learning a latent space representation of the data and sampling new instances from it. Unlike GANs, VAEs optimize a variational lower bound and ensure smooth interpolation between data points, which can be beneficial in generating diverse synthetic samples. Both approaches have been successfully applied in domains such as medical diagnostics and fraud detection, highlighting their potential as alternative oversampling strategies.

Examples of GANs and VAEs being applied to imbalanced datasets include Frid Adar et al. [16], where GANs were used to generate synthetic medical images for liver lesion classification. Douzas and Bacao [17] explored the use of GANs for fraud detection, demonstrating improved model performance in highly imbalanced scenarios.

GM4OS is not the first oversampling methodology in the use of evolutionary algorithms. In the literature is possible to find many applications of GP to imbalanced classification tasks, for instance the work by Frank and Bacao [18], where GP and some of its variance have been successfully applied to several classification tasks. While the contributions of Kumar [19] and Pei et al. [20] introduced novel fitness functions tailored specifically for GP. These metrics have been tailored to improve GP's performance when working with imbalanced classification tasks. Another example of the application of evolutionary algorithms to oversampling is GenSample [21], where GA is used to create synthetic observations. GenSample iteratively identifies the most suitable minority samples for resampling, and the authors have reported promising results when compared to a range of state-of-the-art models.

Genetic Methods for OverSampling

Genetic Methods for OverSampling (GM4OS) is an oversampling approach for imbalanced binary classification problems, founded on the evolution process of GP and GAs.

Individual Structure

GM4OS utilizes the standard evolutionary approach. Its peculiarity is found in the structure of the evolved solutions. Each individual is composed of two parts, one that resembles a standard GP tree and the other one a standard GA individual. The GP part is a function that is able to combine two vectors (existing observations) into a new vector (synthetic observation).

The GA part of every individual is a string of length n, where n is the number of observations needed to balance a binary imbalanced dataset, therefore n will correspond to the difference in number of observation belonging to the majority class with the number of observation belonging to the minority class. Each allele of the GA part of the solution is composed of two pointers to observations that belong to the minority class. By applying the GP tree (function) to all the pairs of pointers present in the GA part, it is possible to create n synthetic points. These observations can be labeled as minority class and be added to the dataset, making it balanced.

The GP component of GM4OS evolves an oversampling function that combines two vectors (existing observations) to create a single one (new synthetic observation). This approach resembles the vectorial GP introduced by Azzali et al. [22, 23]. For example, similar to the GP component of a GM4OS individual, vectorial GP incorporates aggregate functions and vector operations within the primitive GP set, and utilizes vector variables as terminals.

Example We can think of an arbitrary function:

$$\mathcal{P}(\mathbf{x_1}, \mathbf{x_2}) = \mathbf{x_1}/(\mathbf{x_1} \cdot \mathbf{x_2})$$

where $\mathbf{x_1}$ and $\mathbf{x_2}$ are two vectors of the same dimension as the feature space, and the operators / and · represent the element-by-element vectorial division and multiplication, respectively.

Table 1 represents an arbitrary binary imbalanced dataset, composed of six observations, each composed of two variables, and two classes, namely 0 and 1. This dataset is imbalanced, since 4 observation belong to the class 1 (majority class) and 2 observations belong to the class 0 (minority class).

Table 1 Arbitrary binary imbalanced dataset

Index	Input Variables	Target class
obs_1	[1, 3]	1
obs_2	[2, 1]	0
obs_3	[−1, 8]	1
obs_4	[5, 10]	1
obs_5	[7, 24]	1
obs_6	[5, 3]	0

In order to balance the dataset presented in Table 1, two synthetic observations labeled as the minority class are needed. Let's assume that an arbitrary GM4OS individual is composed of the function previously presented and of a vector of pairs of pointer to observations belonging to the minority class, i.e., $[(obs_2, obs_2), (obs_2, obs_6)]$. By applying the function to the pairs of pointer GM4OS will create two new synthetic points:

- $\mathcal{P}(obs_2, obs_2) = [2/(2 \cdot 2),\ 1/(1 \cdot 1)] = [0.5, 1]$
- $\mathcal{P}(obs_2, obs_6) = [2/(2 \cdot 5),\ 1/(1 \cdot 3)] = [0.2,\ 0.\overline{3}]$

These newly created synthetic observations will be labeled as minority class and added to the dataset presented in Table 1, making it balanced.

Individual Evaluation

The evaluation process of GM4OS is based on a training-validation, and optionally testing, split. As described in "Individual Structure" section, all the individuals evolved by the GM4OS evolution process create n synthetic observations, where n is the number of observations that are missing in order to balance the training dataset. These observations are then added to the training dataset with the minority class label, making it balanced.

A model is then fitted on this balanced dataset and evaluated on the validation split of the dataset, that is still imbalanced. The error of the model on the validation split is used as fitness of the individual. Optionally the best individual in the population at each generation, i.e., elite, can also be evaluated on the test split of the dataset, in order to have information on the performance of the model on unseen data. Figure 1 is a graphical representation of the evaluation process of a GM4OS individual.

Genetic Operators

During the evolution process, the genetic operators, namely crossover and mutation, are applied to both part of the individual simultaneously, according to the probabilities presented in "Experimental Settings and Test Problems"

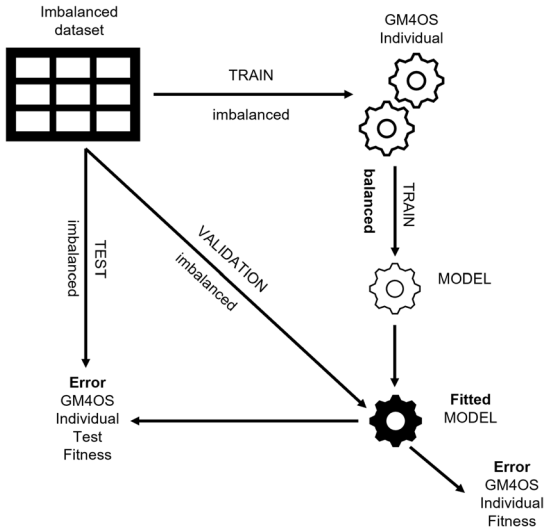


Fig. 1 Graphical representation of the evaluation process of a GM4OS individual

section. Given the structure of individuals, different mutations and crossovers are applied to each part. The different operators for GA and GP used in this experiment, are specified in "Experimental Settings and Test Problems" section. Traditionally, GA individuals are represented as vectors of single values. The fact that the GA part of GM4OS individual are vectors of pair of pointers still allows to use standard GA operators, treating each pair as a single indivisible piece of information.

Once a certain operator is chosen to be applied, either crossover or mutation, that operator is applied to both parts of the individual. For example, if mutation is chosen, a single node mutation will be applied to the GP part of the individual, while a one point mutation will be applied to the GA part of the individual, given the parameters presented in "Experimental Settings and Test Problems" section.

Experimental Settings and Test Problems

Table 2 presents the GM4OS parameters used in our experiments.

Table 3 presents the binary classification datasets used as benchmarks in this study, all taken from the Penn Machine Learning Benchmarks (PMLB) library [27]. The imbalance ratio presented in Table 3 is calculated as the number of observations belonging to majority class divided by the number of observations belonging to the minority class.

Given that in our experiment all the test problems are imbalance, this ratio is always a number strictly greater than one.

As shown in Fig. 1, the evaluation process of every GM4OS individual is based on a classification model that is fitted on the balanced training partition and evaluated on the imbalanced validation partition. Any classification model can be used in this step, e.g., Decision Tree Classifier [28] or Support Vector Classifier [13].

In this work, we decided to use Logistic Regression (LR) [29] as it was used in the original work that introduced GM4OS [8]. A recent study by Elor and AverbuchElor [30] showed how oversampling methods have different performance on models that use a priori HyperParameters (HPs) and the ones the used *tuned* HPs. In the original experiment that introduced GM4OS [8], only a

Table 2 GM4OS parameters

Parameter	Value
Population size	50
Generations	50
Mutation probability	0.2
Crossover probability	0.8
Elitism	True
# of elites	1
Selection algorithm	Tournament [7]
Tournament size	2
GP initialization	Ramped-half-half [7]
GP constant set	{-1, 2, 3, 4, 5}
GP constant probability	0.3
GP function set	{add, sub, mul, div, mean}
GP crossover	Subtree single point swap crossover [7]
GP mutation	Single node mutation [24]
Maximum GP tree depth for initialization	7
Maximum GP tree depth for evolution	17
GA initialization	Random
GA crossover	One point crossover [25]
GA mutation	One point mutation [26]

Table 3 Specifications of the datasets used for model comparison

Dataset	Number of observations	Number of features	Imbalance ratio
flare	1066	10	4.8
haberman	306	3	2.7
spect	267	22	3.8
spectf	349	44	2.7
ionosphere	351	34	1.8
hungarian	294	13	1.77
diabetes	768	8	1.8
hepatitis	155	19	3.84
appendicitis	106	7	4.04
analcatdata	264	4	21.8

Table 4 Grid search specification

Parameter	Values
gamma	0, 0.1, 0.4, 1.6, 3.2, 12.8, 51.2, 200
learning_rate	0.01, 0.06, 0.1, 0.2, 0.3, 0.4, 0.6, 0.7
max_depth	5, 7, 10, 13, 15
n_estimators	50, 80, 115, 150
reg_alpha	0, 0.1, 0.4, 1.6, 3.2, 12.8, 51.2, 200
reg_lambda	0, 0.1, 0.4, 1.6, 3.2, 12.8, 51.2, 200

Table 5 Tuned HyperParameters for the eXtreme Gradient Boosting classifier for each one of the binary classification datasets used as benchmarks

Dataset	gamma	learn. rate	max. depth	L1 reg.	L2 reg.
flare	1.6	0.6	7	0.1	0.4
haberman	3.2	0.7	5	0.1	1.6
spect	0	0.06	5	0.1	0.1
spectf	0	0.3	7	3.2	1.6
ionosphere	0.1	0.6	5	0	3.2
hungarian	0	0.7	7	1.6	0.4
diabetes	0.4	0.7	5	3.2	1.6
hepatitis	0	0.6	5	0	1.6
appendicitis	1.6	0.3	5	0.1	0.1
analcatdata	0	0.06	5	0	0

LR with a priori HPs was used, limiting the investigation to a weak classifier. In this experiment the study is also extended to two version of eXtreme Gradient Boosting (XGB) [31], one with a priori HPs, from now on referred as XGB, and one with *tuned* HPs, from now on referred as tuned-XGB, found by running a grid search, presented in Table 4, on each one of the binary classification datasets used as benchmarks.

The HPs found by running the grid search are presented in Table 5, the # of estimators was always set to 50.

In order to assess the performance of GM4OS, we employed 6 baseline methods. The first one is the model, either LR, XGB or tuned-XGB, fitted on the imbalanced training dataset. The other 5 are the same model, still either LR, XGB or tuned-XGB, fitted on the training dataset balanced with the techniques explained in "Literature Review" section SMOTE, B-SMOTE, G-SMOTE, SVM-SMOTE or ADASYN.

It is possible to use different metrics when it comes to classification tasks, g.e., recall, accuracy, F1-score. In the work that introduced GM4OS [8], the F1-score was used as fitness function for the optimization process of GM4OS. In this study, we extend GM4OS by using five different fitness functions for the optimization process, i.e., F1-score, recall, precision, G-score, accuracy, making a comparison against the baselines on each benchmark for each metric [32].

In order to ensure statistical significance each experiment, consisting of each algorithm/dataset combination, was run 30 times. Every run had a different train/validation/test partition, composed of 60%, 20% and 20% and consistent across all the runs. The proportion between majority and minority class observations is kept constant across the different splits. For all the baseline oversampling methods the validation partition was joined with the training one. This does not provide an advantage for those method since the validation partition is used for the evaluation process of GM4OS. All the statistical test presented in the following section are Mann–Whitney U test for pairwise comparison with significance level of $\alpha = 0.05$.

Experimental Results

General Results Analysis

Table 6 (Tables 7 and 8, respectively) is a summary of the comparison of the performance on the test set of GM4OS against the baselines across all the benchmarks when using LR (XGB and tuned-XGB) as classification model. Each row is a different version of GM4OS with a different classification metric as fitness function. Every columns is a different classification metric used for comparison. In every cell the first value, always positive or 0, indicates for how many times GM4OS statistically outperformed the baselines. The second value present in every cell, always negative or

Table 6 Summary of the results for each metric when using LR as classification model

Model	F1-score	precision	recall	G-score	accuracy	total
GM4OS (F1-score)	**+23, 0**	+10, −8	+28, −1	+6, −18	+28, −1	+95, −28
GM4OS (precision)	+4, −17	**+5, −11**	+14, −9	+5, −30	+14, −9	+42, −76
GM4OS (recall)	+13, −1	+6, −11	**+27, −1**	+7, −3	+27, −0	+80, −16
GM4OS (G-score)	+2, −14	+5, −1	+6, −16	**+7, −3**	+6, −16	+26, −50
GM4OS (accuracy)	+14, 0	+6, −10	+27, −0	+3, −31	**+27, 0**	+77, −41

Each row represents a different version of GM4OS with a different fitness function. In bold are the results for the metric used as fitness function. The first number, always positive, it indicates how many times GM4OS significantly outperforms the baselines. While the second number, always negative, it indicates how many times GM4OS is significantly outperformed by the baselines

Table 7 Summary of the results for each metric when using XGB as classification model

Model	F1-score	precision	recall	G-score	accuracy	total
GM4OS (F1-score)	**+2, −4**	+0, −5	+3, −7	+0, −14	+3, −7	+7, −37
GM4OS (precision)	+1, −1	**+1, −0**	+0, −1	+1, −0	+0, −1	+3, −3
GM4OS (recall)	+6, −5	+1, −8	**+6, −6**	+1, −14	+6, −6	+22, −39
GM4OS (G-score)	+0, −3	+2, −2	+0, −7	**+6, −0**	+0, −7	+8, −19
GM4OS (accuracy)	+6, −5	+1, −8	+6, −6	+1, −14	**+6, −6**	+20, −39

Each row represents a different version of GM4OS with a different fitness function. In bold are the results for the metric used as fitness function. The first number, always positive, it indicates how many times GM4OS significantly outperforms the baselines. While the second number, always negative, it indicates how many times GM4OS is significantly outperformed by the baselines

Table 8 Summary of the results for each metric when using tuned-XGB as classification model

Model	F1-score	Precision	Recall	G-score	Accuracy	Total
GM4OS (F1-score)	**+13, −1**	+5, −9	+7, −2	+0, −7	+8, −2	+33, −21
GM4OS (precision)	+4, −0	**+6, −6**	+2, −5	+8, −1	+1, −5	+21, −17
GM4OS (recall)	+23, −1	+10, −6	**+18, −2**	+3, −9	+20, −2	+74, −20
GM4OS (G-score)	+2, −8	+2, −7	+1, −14	**+10, −4**	+1, −14	+16, −47
GM4OS (accuracy)	+23, −1	+10, −6	+18, −2	+3, −9	**+20, −2**	+74, −20

Each row represents a different version of GM4OS with a different fitness function. In bold are the results for the metric used as fitness function. The first number, always positive, it indicates how many times GM4OS significantly outperforms the baselines. While the second number, always negative, it indicates how many times GM4OS is significantly outperformed by the baselines

0, indicates how many times the baselines outperformed GM4OS. Both values can reach a maximum of 60, as each test was conducted for every dataset-baseline combination, with a total of 10 datasets and 6 baselines. The number of tests that were not statistically significant for each cell can be determined by subtracting the sum of the values in the cell from 60. The values in the cell are presented in bold when the fitness function used in GM4OS is the same as the metric used for comparison. The last column is the total for every presented GM4OS model.

LR Looking at Table 6 we immediately see how the versions of GM4OS that use as fitness function F1-score and accuracy always outperform or have comparable performance with the baselines methods when the same metric is used as comparison. This is not true for the remaining three metrics, that are precision, recall, G-score. Although the version of GM4OS that uses recall as fitness function is outperformed by the baselines only in one occasion.

XGB When using XGB as classification model the performance of GM4OS clearly deteriorates, as shown in Table 7, not being able to statistically outperform the baselines as it was doing while using LR. Only the versions of GM4OS that use precision and G-score as fitness function always significantly outperform or have similar performance to the baseline methods when using the same metric for comparison with the benchmarks.

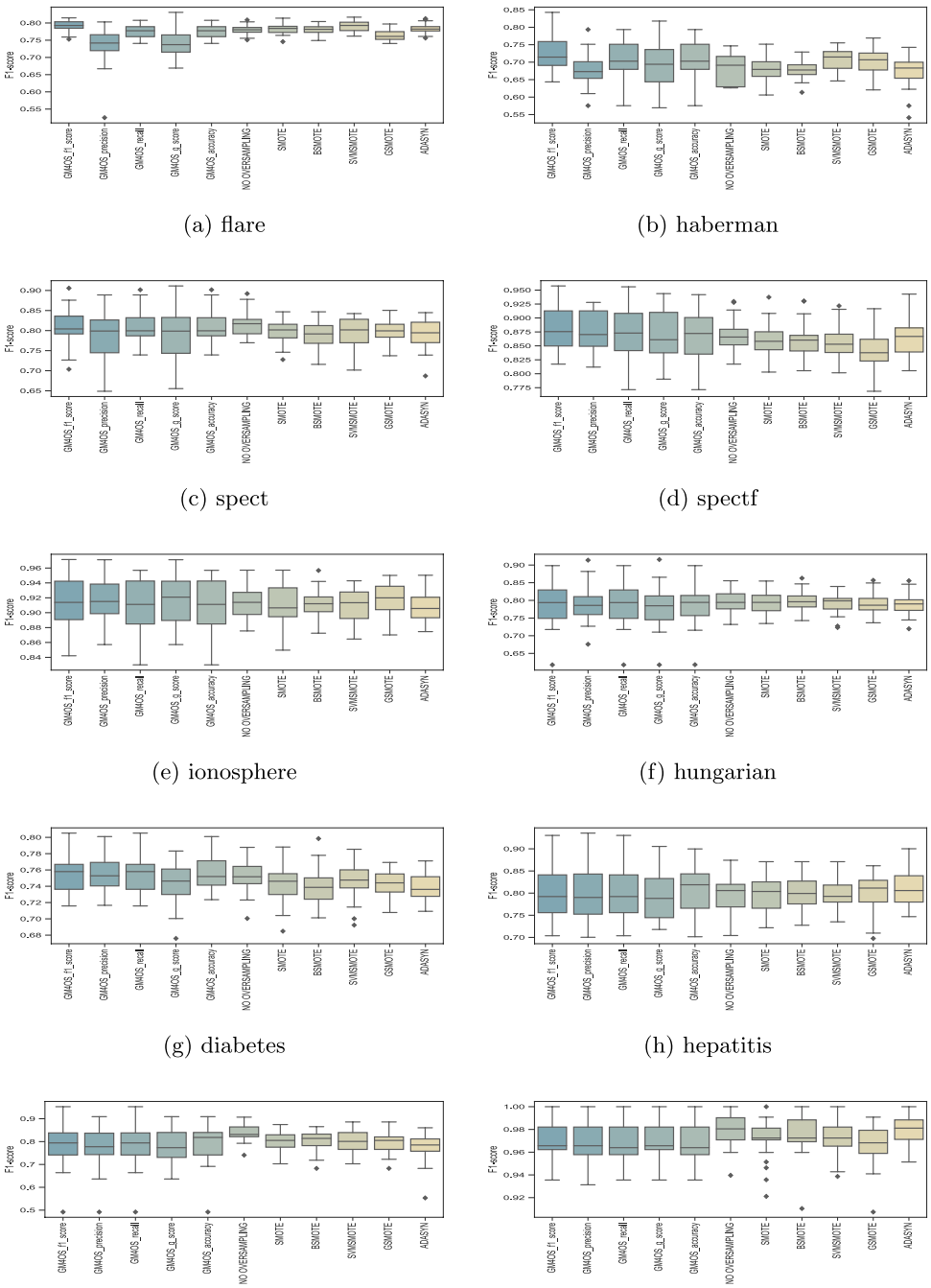
Tuned-XGB In the last group of experiments, in which tuned-XGB is used as classification model, it is not possible to identify any pattern. As shown in Table 8 all the version of GM4OS have mixed performance compared to the baselines.

Discussion The aforementioned results behave accordingly to what was introduced by Elor and Averbuch-Elor [30]. That is, oversampling methods, including GM4OS, improve the performance of *weak* classifiers, like LR and XGB, while are less useful when it comes to *strong* classifiers, i.e., tuned-XGB.

Focus on *Strong* Classifiers

Figure 2 presents the performance of all the oversampling methods when using tuned-XGB as classification method across all the test problems. The performance are presented as the boxplots obtained with the results of the 30 runs. The metric used for this comparison is the F1-score.

As already mentioned in "Experimental Settings and Test Problems" section, and also in [8] and in [32], the F1-score provides a balanced measure of both precision and recall. So, it is well-suited for situations where the imbalanced distribution of classes demands a focus on the accurate identification of the minority class instances, striking a balance between minimizing false positives and false negatives. The focus of this part of the analysis is only onto tuned-XGB since, as already mentioned in "Experimental Settings and Test



(a) flare (b) haberman

(c) spect (d) spectf

(e) ionosphere (f) hungarian

(g) diabetes (h) hepatitis

(i) appendicitis (j) analcatdata

Fig. 2 Box-plots of the F1-score of the minority class of GM4OS against the baselines for all the datasets when using tuned-XGB as classification model

Problems" section, can be considered as a *strong* learner, contrary to LR and XGB, that are *weak* learners [30].

By looking at the boxplots we can clearly see how the version of GM4OS that uses F1-score as fitness function always has better performance than the versions that use a different classification metric. This was expected, since the F1-score is continuously being optimized throughout the evolution process. Another important point is that no benchmark oversampling method outperforms the model trained on the imbalanced dataset, corroborating the thesis proved by Elor and Averbuch-Elor [30]. The last point that we can get from these boxplots is that in three test cases out of 10, flare, haberman and spectf, the version of GM4OS that uses F1-score as fitness function has the best performance across all the showed methods. This version of GM4OS across these three test cases, is clearly outperforming the other GM4OS versions, the oversampling baselines and the model trained on the imbalanced dataset.

Oversampled Points Location in the Search Space

Figure 3 (and Fig. 4) shows a visualization of the synthetic points generated by all the oversampling methods presented in this work for the analcatdata (and appendicitis) test problems. Given the multidimensionality of these problems, Principal Component Analysis (PCA) [33] was used in order to reduce the dimensions and making possible to plots the points in a two-dimensional space. Each plotted point has a corresponding label and marker. The points labeled as 0 and marked with a blue dot are the ones that belong to the majority class, while the ones labeled as 1 and marked with a green dot are the ones that belong to the minority class. Finally the points labeled as 1' and marked with a red cross, are the new synthetic points generated by the oversampling method. For GM4OS, both plots are the synthetic points generated with the version that uses F1-score as fitness function.

From Figs. 3 and 4 it's possible to observe that there is substantial difference in the location of the synthetic points

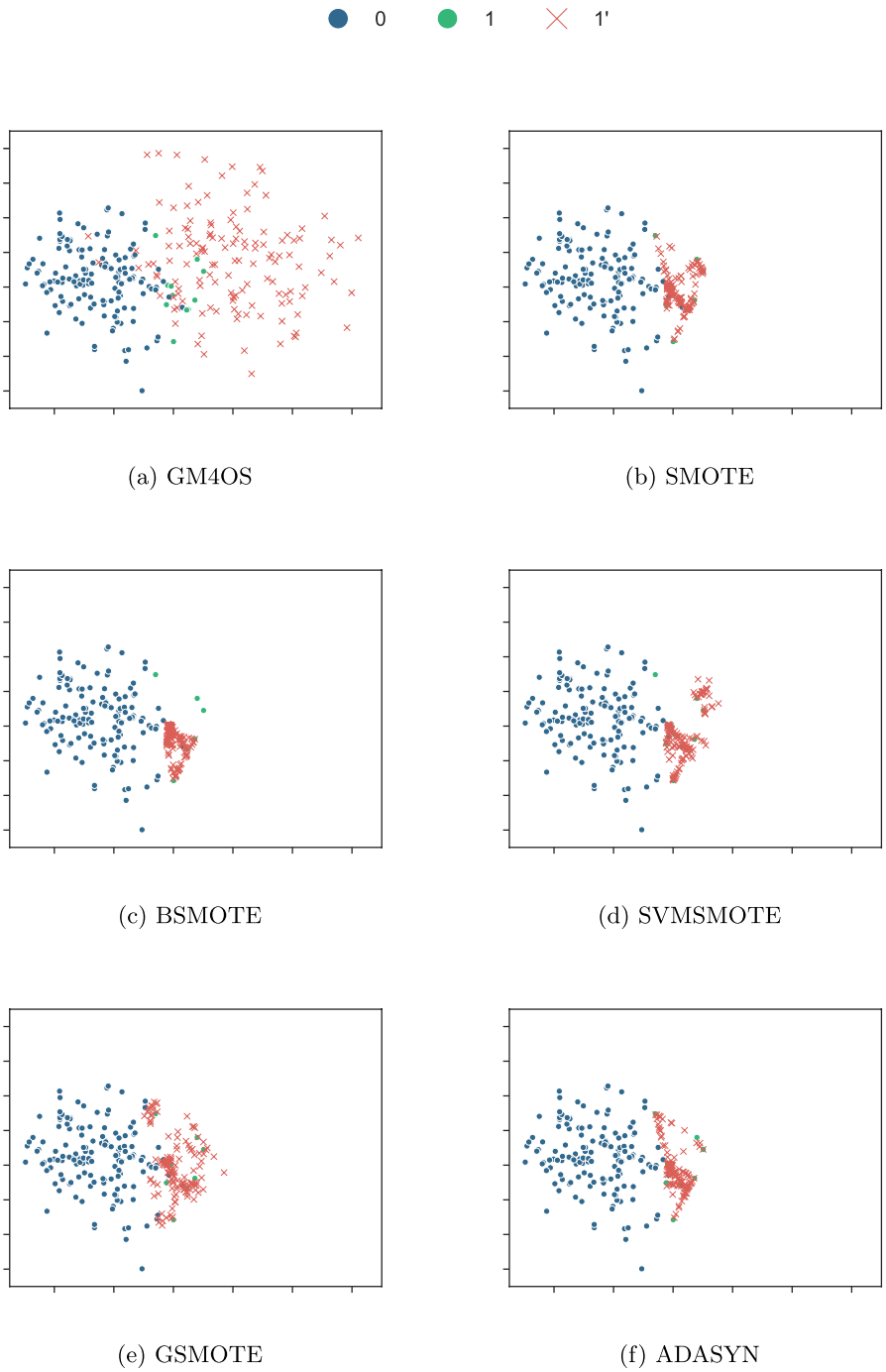


Fig. 3 Scatter plots of the synthetic points generated with the oversampling methods used in this experiment for the analcatdata test problem

Fig. 4 Scatter plots of the synthetic points generated with the oversampling methods used in this experiment for the appendicitis test problem

generated by GM4OS and the ones generated by the other presented oversampling methods. The latter ones are fairly close to the original points that belong to the minority class. While the synthetic points generated by GM4OS appear to be sampled in different regions of the search space, further away from the original minority class observations.

Given the limitations of the oversampling methods found in the literature, described by Elor and Averbuch-Elor [30], the location of the synthetic observations generated by GM4OS could be why this newly introduced oversampling method is able to overcome this limitations, as discussed previously in "Focus on Strong Classifiers" section. The location of the GM4OS generated synthetic points in different regions of the search space may help *strong* classifiers, g.e., tuned-XGB, to identify the border between classes.

Analysis of Runtime

Table 9 presents the median runtimes in seconds of all the oversampling approaches discussed in this work, with a focus on comparing GM4OS to state-of-the-art methods.

Notably, GM4OS is significantly more computationally expensive than the other methods-by four orders of magnitude-making it considerably heavy. However, in the authors' opinion, this runtime cost is justified by GM4OS's ability to positively influence the learning process of *strong* classifiers in three out of ten test cases, whereas state-of-the-art methods fail to improve the performance. It is also important to highlight that the extensive running time discussed pertains exclusively to the learning phase, which is typically conducted once and offline. The model produced during this phase, when deployed online, generally requires significantly less computational power and operates at a faster rate. Therefore, we argue that practitioners should be willing to allocate more time to this foundational learning stage, provided that it results in a superior quality of the final model.

Conclusion and Future Work

This paper introduced the Genetic Methods for OverSampling (GM4OS), an evolutionary oversampling approach for imbalanced binary classification problems. The classification of imbalanced datasets represents a common issue in machine learning.

A common approach is to add synthetic points to the minority class to balance the dataset, also referred to as oversampling. Common oversampling approaches found in the literature, g.e., SMOTE [3], BSMOTE [4] and ADASYN [5], heavily rely on a set of existing points used for resampling and a function that combines these points to create new synthetic observations.

GM4OS integrates the representation power of Genetic Algorithms (GAs) and Genetic Programming (GP), to look for the most appropriate resampling set and resampling function at the same time. GM4OS are composed of two parts, a GP-like function, that can be represented as a tree, and a GA-like string. The GP part is a function that is able to combine two vectors (existing observations) into a single new vector (new synthetic observation), resembling what introduced in [22] and [23]. The GA part controls instead which observations belonging to the minority class are used in the resampling set.

GM4OS was compared with different oversampling methods, like SMOTE, BSMOTE, SVMSMOTE [12], GSMOTE [11], ADASYN, and with a classification model trained on the imbalanced dataset. Three different models were used as basic classifiers: Logistic Regression (LR), a version of eXtreme Boosting Gradient (XGB) with a priori HyperParameters (HPs), and a version with *tuned* HPs, referred to as tuned-XGB. The former two were identified as *weak* classifiers, while the latter as *strong* classifier, according to the definition provided in [30]. The comparison was conducted on ten imbalanced binary classification test problems, taken from the Penn Machine Learning Benchmarks library [27] and using 5 classification metrics, namely F1-score, precision, recall, G-score and accuracy [32].

The results show that on *weak* classifiers GM4OS significantly improves the performance of the classification model compared with the baselines methods. This performance deteriorates when working with *strong* classifiers, where the results were more mixed and no specific pattern was found. A deeper focus was then dedicated to the study of the performance of the different oversampling methods when using tuned-XGB as base classifier, according to the F1-score. In this case the results showed how the baseline oversampling methods were not able to improve the performance of the classification model, corroborating the thesis of Elor and Averbuch-Elo [30], and also how the GM4OS version that used F1-score as fitness was the best one among all the GM4OS versions. The results also showed how, on three cases out of ten, GM4OS was able to improve the performance of the *strong* classifier.

Table 9 Median runtimes of the different oversampling approaches in seconds

Problem	ADASYN	BSMOTE	GSMOTE	LR	SMOTE	SVMSMOTE	GM4OS
analcatdata	0.072215	0.073123	0.091730	0.061844	0.073509	0.075743	138.345697
appendicitis	0.050256	0.050189	0.060976	0.048037	0.049696	0.057886	101.448145
diabetes	0.070075	0.062267	0.076469	0.057653	0.058088	0.075979	130.244718
flare	0.107965	0.150870	0.144363	0.076378	0.095156	0.124187	150.369106
haberman	0.096612	0.071833	0.074943	0.103064	0.080478	0.078273	116.995714
hepatitis	0.067654	0.073885	0.084375	0.064877	0.063793	0.075947	126.832157
hungarian	0.060918	0.060488	0.071669	0.057864	0.057820	0.070529	128.708516
ionosphere	0.118657	0.111970	0.115985	0.129721	0.107697	0.116437	158.604039
spect	0.096261	0.113007	0.123944	0.111322	0.131008	0.107208	167.300827
spectf	0.096431	0.103281	0.118107	0.077656	0.093629	0.111736	218.428549

Finally, a graphical comparison, between the location in the search space of the synthetic points created by all the presented oversampling methods, was conducted. This analysis showed how GM4OS creates synthetic points in different portion of the search space compared to the baseline oversampling methods.

Despite the positive experimental outcomes achieved, there remains significant scope for future research. One potential avenue involves adapting the GM4OS framework to address multi-class classification problems. Another area for investigation is exploring alternative fitness metrics for GM4OS, distinct from the ones used in this study. Specifically, pursuing metrics that have shown efficacy for GP applied to imbalanced classification tasks, as demonstrated in [19, 20], holds significant promise. Additionally, multi-objective optimization deserves investigation. Furthermore, as a forthcoming research endeavor, it is important to note that this study employed two classification models, Logistic Regression [29] and eXtreme Boosting Gradient [31]. However, the framework remains adaptable for experimentation with alternative models, such as Decision Tree classifiers [28] and many others. Also, our future research will encompass further comparisons with alternative methods to handle imbalanced datasets, different from SMOTE, GSMOTE, ADASYN and the other baseline algorithms used here, including undersampling and internal approaches. Another research direction should focus on reducing the algorithm's runtime by developing new approaches or optimizing resource utilization. One possible solution is to evolve only a *general* oversampling function that can be evolved once and applied across different domains, thereby improving efficiency and minimizing computational costs. Lastly, an intriguing prospect for future research involves extending GM4OS to encompass synthetic data generation. This can be achieved by modifying the resampling set governed by the GA component of GM4OS to extend its influence to the entire dataset.

Funding Open access funding provided by FCT|FCCN (b-on). This work was supported by national funds through FCT (Fundação para a Ciência e a Tecnologia), under the project - UIDB/04152 - Centro de Investigação em Gestão de Informação (MagIC)/NOVA IMS.

Data availability The datasets used from this experiment are taken from the Penn Machine Learning Benchmarks https://epistasislab.github.io/pmlb/.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no Conflict of interest.

References

1. Batista GEAPA, Prati RC, Monard MC. A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor. 2004;6:20–9.
2. Chawla N, Japkowicz N, Kołcz A. Editorial: Special issue on learning from imbalanced data sets. SIGKDD Explor. 2004;6:1–6. https://doi.org/10.1145/1007730.1007733.
3. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. Smote: Synthetic minority over-sampling technique. J Artif Int Res. 2002;16(1):321–57.
4. Han H, Wang W-Y, Mao B-H. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In: Proceedings of the 2005 international conference on advances in intelligent computing—volume part I. ICIC'05. Springer, Berlin; 2005. p. 878–887. https://doi.org/10.1007/11538059_91 .
5. He H, Bai Y, Garcia EA, Li S. Adasyn: adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence); 2008. p. 1322–1328.
6. Mitchell M. An introduction to genetic algorithms. Cambridge: MIT Press; 1996.
7. Poli R, Langdon WB, McPhee NF. A field guide to genetic programming. Morrisville: Lulu Enterprises UK Ltd; 2008.
8. Farinati D, Vanneschi L. Gm4os: an evolutionary oversampling approach for imbalanced binary classification tasks. In: Smith S, Correia J, Cintrano C, editors. Applications of evolutionary computation. Cham: Springer; 2024. p. 68–82.
9. Huang J, Ling CX. Using AUC and accuracy in evaluating learning algorithms. IEEE Trans Knowl Data Eng. 2005;17(3):299–310. https://doi.org/10.1109/TKDE.2005.50.
10. Ali A, Shamsuddin SM, Ralescu A. Classification with class imbalance problem: a review. Int J Adv Soft Comput Appl. 2015;7:176–204.
11. Douzas G, Bacao F. Geometric smote a geometrically enhanced drop-in replacement for smote. Inf Sci. 2019;501:118–35. https://doi.org/10.1016/j.ins.2019.06.007.
12. Nguyen HM, Cooper EW, Kamei K. Borderline over-sampling for imbalanced data classification. Int J Knowl Eng Soft Data Paradigm. 2011;3(1):4–21. https://doi.org/10.1504/IJKESDP.2011.039875.
13. Evgeniou T, Pontil M. Support vector machines: theory and applications. In: Advanced course on artificial intelligence, vol. 2049; 2001. p. 249–57. https://doi.org/10.1007/3-540-44673-7_12.
14. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. In: Advances in neural information processing systems; 2014. pp. 2672–2680.https://doi.org/10.48550/arXiv.1406.2661
15. Kingma DP, Welling M. Auto-encoding variational Bayes. arXiv preprint https://doi.org/10.48550/arXiv.1312.6114; 2013
16. Frid-Adar M, Klang E, Amitai M, Goldberger J, Greenspan H, Synthetic data augmentation using GAN for improved liver lesion

classification. In: 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI). IEEE. 2018.p. 04–07. https://doi.org/10.1109/ISBI.2018.8363576.
17. Douzas G, Bacao F. Effective data generation for imbalanced learning using conditional generative adversarial networks. Expert Syst Appl. 2018;91:464–71. https://doi.org/10.1016/j.eswa.2017.09.030.
18. Frank F, Bacao F. Advanced genetic programming vs. state-of-the-art AutoML in imbalanced binary classification. Emerg Sci J. 2023;7(4):1349–63. https://doi.org/10.28991/ESJ-2023-07-04-021.
19. Kumar A. A new fitness function in genetic programming for classification of imbalanced data. J Exp Theor Artif Intell. 2024;36:1021–33. https://doi.org/10.1080/0952813X.2022.2120087.
20. Pei W, Xue B, Shang L, Zhang M. New fitness functions in genetic programming for classification with high-dimensional unbalanced data. In: 2019 IEEE Congress on evolutionary computation (CEC); 2019. p. 2779–2786. https://doi.org/10.1109/CEC.2019.8789974
21. Karia V, Zhang W, Naeim A, Ramezani R. GenSample: a genetic algorithm for oversampling in imbalanced datasets' 2019.
22. Azzali I, Vanneschi L, Silva S, Bakurov I, Giacobini M. A vectorial approach to genetic programming. In: Sekanina L, Hu T, Lourenço N, Richter H, García-Sánchez P, editors. Genetic program. Cham: Springer; 2019. p. 213–27.
23. Azzali I, Vanneschi L, Bakurov I, Silva S, Ivaldi M, Giacobini M. Towards the use of vector based GP to predict physiological time series. Appl Soft Comput. 2020;89:106097. https://doi.org/10.1016/j.asoc.2020.106097.
24. Quan W, Soule T. A study of the role of single node mutation in genetic programming. In: Genetic and evolutionary computation—GECCO 2004. Berlin: Springer; 2004. p. 717–718. https://doi.org/10.1007/978-3-540-24855-2_84
25. Crossover operators in genetic algorithms: a review. 2025. https://www.researchgate.net/publication/288749263_CROSSOVER_OPERATORS_IN_GENETIC_ALGORITHMS_A_REVIEW
26. GA one-point mutation of a numerical string; 2025. https://1library.net/article/ga-one-point-mutation-of-a-numerical-string.q0vvpl3z
27. Romano JD, Le TT, La Cava W, Gregg JT, Goldberg DJ, Chakraborty P, Ray NL, Himmelstein D, Fu W, Moore JH. Pmlb v1.0: an open source dataset collection for benchmarking machine learning methods. arXiv preprint arXiv:2012.00058v2; 2021.
28. Breiman L, Friedman JH, Olshen RA, Stone CJ. Classification and regression trees. Biometrics. 1984;40:874.
29. Cox DR. The regression analysis of binary sequences. J R Stat Soc Ser B (Methodol). 1958;20(2):215–32.
30. Elor Y, Averbuch-Elor H. To SMOTE, or Not to SMOTE?
31. Chen T, Guestrin C. Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. KDD '16. ACM; 2016.https://doi.org/10.1145/2939672.2939785.
32. Ferrer L. Analysis and comparison of classification metrics; 2023.
33. Anderson TW. Asymptotic theory for principal component analysis. Ann Math Stat. 1963;34(1):122–48. https://doi.org/10.1214/aoms/1177704248.