GM4OS: An Evolutionary Oversampling Approach for Imbalanced Binary Classification Tasks

Davide Farinati(✉) and Leonardo Vanneschi

NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisboa, Portugal
dfarinati@novaims.unl.pt

Abstract. Imbalanced datasets pose a significant and longstanding challenge to machine learning algorithms, particularly in binary classification tasks. Over the past few years, various solutions have emerged, with a substantial focus on the automated generation of synthetic observations for the minority class, a technique known as oversampling. Among the various oversampling approaches, the Synthetic Minority Oversampling Technique (SMOTE) has recently garnered considerable attention as a highly promising method. SMOTE achieves this by generating new observations through the creation of points along the line segment connecting two existing minority class observations. Nevertheless, the performance of SMOTE frequently hinges upon the specific selection of these observation pairs for resampling. This research introduces the Genetic Methods for OverSampling (GM4OS), a novel oversampling technique that addresses this challenge. In GM4OS, individuals are represented as pairs of objects. The first object assumes the form of a GP-like function, operating on vectors, while the second object adopts a GA-like genome structure containing pairs of minority class observations. By co-evolving these two elements, GM4OS conducts a simultaneous search for the most suitable resampling pair and the most effective oversampling function. Experimental results, obtained on ten imbalanced binary classification problems, demonstrate that GM4OS consistently outperforms or yields results that are at least comparable to those achieved through linear regression and linear regression when combined with SMOTE.

Keywords: Oversampling · Imbalanced Data · Binary Classification · Genetic Programming · Genetic Algorithms

1 Introduction

In real-world classification tasks, it is common to encounter datasets where the proportion of labels is not homogeneous among the different classes. This problem is commonly referred to as imbalance. Classification models tend to exhibit higher accuracy when handling observations from the class that is more prevalent, often termed as the majority class, as opposed to the less frequently

S. Smith et al. (Eds.): EvoApplications 2024, LNCS 14634, pp. 68–82, 2024.
https://doi.org/10.1007/978-3-031-56852-7_5

occurring class, referred to as the minority class. A common strategy is to add new observations, typically created artificially, to the minority class, to improve the balancing of the dataset. The approach is called oversampling [1,2]. One of the most used oversampling approaches is the Synthetic Minority Oversampling Technique (SMOTE) [3]. This oversampling algorithm works by selecting a random pair of neighboring observations, drawing a straight line segment between the two, and randomly sampling a new observation along that segment. One of its notable drawbacks lies in the sensitivity of its performance to the choice of the set of data points used for resampling. The quality and relevance of the selected points play a crucial role in determining the effectiveness of SMOTE in generating synthetic samples that accurately represent the minority class. 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) [4] and Genetic Programming (GP) [5], to overcome some of the disadvantages of SMOTE. In GM4OS, individuals are represented as pairs of objects. The first one is a function, represented as a standard GP individual. The other one is a string, as a traditional GA individual. The GA part controls which existing observations from the minority class will belong to the resampling set. The GP part evolves an oversampling function that will combine points that belong to the resampling set, and create the new synthetic points needed to balance the dataset. The fitness of a GM4OS individual is given by the performance of a model trained by a previously chosen target machine learning algorithm on the newly balanced training set. The objective of the evolutionary process is to look for the best resampling set and the most effective oversampling function at the same time.

The paper is organized as follows: in Sect. 2 we revise previous and related work. In Sect. 3 we present GM4OS. Section 4 delves into the employed experimental framework, including the set of parameters, the datasets chosen as test cases, the models utilized as baseline for comparison with GM4OS and the used metrics. Section 5 presents and discusses the obtained experimental results. Finally, Sect. 6 concludes the work and suggests ideas for future research.

2 Literature Review

Imbalanced datasets represent a recurrent challenge in real-world classification tasks. In the literature, both internal and external approaches have been used in an attempt to obviate the impact of imbalanced data on the model's performance. External approaches (data level) rebalance the dataset by either removing observations from the majority class (undersampling) or by adding observations to the minority class (oversampling) [1,2]. In internal approaches (algorithm level), the algorithm is adapted to handle automatically imbalanced observations. This can be done by either assigning a different weight to each class [6] or by using multiple classifiers simultaneously, also referred to as ensemble learning [7]. A combination of both internal and external approaches can be implemented as well. The Synthetic Minority Oversampling

Technique (SMOTE) is an external approach that was designed by Chawla *et al.* in 2002 [3]. SMOTE generates new observations belonging to the minority class as a combination of existing points of the same class. One of the drawbacks of SMOTE is the over generalization of the minority class, leading to a possible overlap between classes [8]. Borderline observations are more important for classification task, being the ones more prone to misclassification. For this reason a variant of SMOTE, named borderline-SMOTE [9], uses as resampling set only the borderline and nearby observations. Another known oversampling technique is the Adaptive Synthetic Sampling Approach (Adasyn) [10]. In Adasyn, minority data samples are generated according to their distribution. To mitigate the learning bias present in the initial dataset, this approach employs a strategy where a greater number of synthetic observations is generated from those minority observations that happen to be more challenging to learn. This is done by assigning weights to the various minority class samples.

The GM4OS method presented in this study is an evolutionary algorithm that joins the representation power of GP and GAs. For this reason, particularly interesting for this literature review are some existing oversampling methodologies grounded in the use of evolutionary algorithms. GP has been previously applied to imbalanced classification tasks. For instance, standard GP and some of its variants have been successfully applied to several classification tasks in [11]. Also, recent research contributions by Pei *et al.* [12] as well as Kumar [13] have introduced novel fitness functions tailored specifically for GP. These innovative fitness functions have been designed to enhance GP's performance when addressing the intricacies of imbalanced classification scenarios. The GenSample algorithm introduced by Karia *et al.* in 2019 [14], implements an oversampling technique based on GAs. GenSample iteratively learns which minority samples are best suited for resampling and the authors reported on promising results compared to a set of state-of-the-art models.

The oversampling function evolved by the GP component of GM4OS combines two vectors (existing observations) to create a single one (a new synthetic observation). This process resembles the vectorial GP approach that was presented by Azzali *et al.* in [15,16]. For instance, similarly to the GP component of a GM4OS individual, vectorial GP allows aggregate functions and vectorial operations in the primitive GP set and vectorial variables as terminals.

3 Genetic Methods for Oversampling

As stated above, the Genetic Methods for OverSampling (GM4OS) is a novel oversampling approach for binary classification problems, based on a combination of GP and GAs representations. GM4OS employs the conventional workflow of evolutionary algorithms. Its peculiarity resides in the representation of the evolving solutions. In GM4OS, in fact, individuals are represented as pairs of objects, one resembling a standard GP individual and the other one a string-like GA individual. The GP part is a function, that can be represented for instance as a tree, that is able to take as input two vectors (existing observations) and combine them to create a new observation (synthetic observation).

The GA part is a string of length $n = size\_majclass - size\_minclass$, where $size\_majclass$ and $size\_minclass$ are, respectively, the number of observations belonging to the majority and minority class in the training set. Each allele of the GA part of the individual contains a pair of observations of the training set belonging to the minority class. Figure 1 provides a visual representation of an arbitrary GM4OS individual.

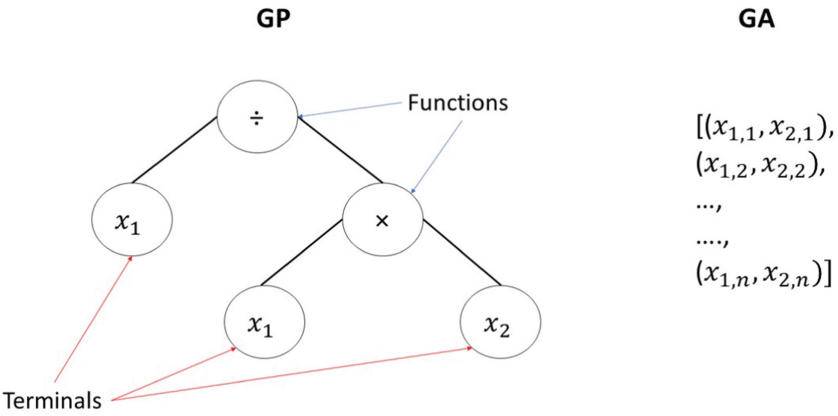


Fig. 1. Visual representation of a GM4OS individual. The GP function reported on the left side takes as input two vectors, $\mathbf{x_1}$ and $\mathbf{x_2}$ (existing observations) and combines them to create a new observation, given by its output vector (synthetic observation). The GA genome reported on the right side contains pairs of minority class observations. By giving each one of these pairs as input to the GP part, n synthetic observations can be created.

At the beginning of the evolution, the classification dataset is partitioned into 3 subsets: the training, validation and test sets. Then, by applying the function represented in the GP part to each one of the n pairs of observations of the GA part, n new synthetic observations are produced. The new observations are labeled as minority class and added to the training set, making it balanced.

Example. The GP part of the individual in Fig. 1, shown in the left side of the figure, represents the 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 $\cdot$ represent the element-by-element vectorial division and multiplication, respectively. Now, let us assume, for the sake of simplicity, that $n = 2$ (i.e., the majority class has only two observations more than the minority class). In such a case, the GA part would be represented as a vector of two pairs of observations, chosen randomly from the minority class. In the

simplistic hypothesis that the feature space has a dimension equal to three, and using arbitrary numbers as data, let us assume that the GA part is:

$$[([2,\ 1,\ 4], [1,\ 2,\ 0.5]), ([1,\ 3,\ 2], [2,\ 1,\ 0.5])]$$

where $[2,\ 1,\ 4], [1,\ 2,\ 0.5], [1,\ 3,\ 2]$ and $[2,\ 1,\ 0.5]$ are existing training observations, labeled as minority class. In such a simple example, GM4OS would create the two following synthetic observations:

- $\mathcal{P}([2,\ 1,\ 4], [1,\ 2,\ 0.5]) = [2/(2 \cdot 1),\ 1/(1 \cdot 2),\ 4/(4 \cdot 0.5)] = [1,\ 0.5,\ 2]$
- $\mathcal{P}([1,\ 3,\ 2], [2,\ 1,\ 0.5]) = [1/(1 \cdot 2),\ 3/(3 \cdot 1),\ 2/(2 \cdot 0.5)] = [0.5,\ 1,\ 2]$

These two newly created observations would now be inserted in the training set and labeled as minority class. In this way, the training set is now balanced.

At this point, a classification model is fitted on the newly balanced training set, and it is then used to make predictions on the validation set. The loss between expected and calculated outputs on the validation set is finally used as fitness for the GM4OS individual. In this work, the classification model chosen is Logistic Regression [17], given its simplicity and training efficiency. Figure 2 shows a flowchart representing the fitness evaluation process of a GM4OS individual.

During the evolution, distinct genetic operators are applied independently to each one of the two parts of a GM4OS individual: when crossover or mutation need to be applied to a GM4OS individual (according to the probabilities presented in Sect. 4), the same operator type (mutation or crossover) is applied simultaneously to both its GP and GA parts. The genetic operators used in this work for the GP and GA parts are also specified in Sect. 4. Traditionally, GA individuals are represented as vectors of scalar values. However, the fact that in GM4OS the GA part is a vector of pairs (the minority class observations that will be used by the GP part) does not represent an issue: it is still possible to use standard GA operators, treating each pair as a single and indivisible piece of information. So, for instance, one-point crossover will exchange substrings of pairs between parents, while one-point mutation will replace an existing pair with a new pair of minority class observations, generated at random. Note that when one-point crossover is applied the crossover point is restricted to fall in the boundaries between observations within a pair.

4 Experimental Settings and Test Problems

Table 1 reports all the GM4OS parameters employed in this experimental study. To assess the performance of GM4OS, we have employed two baseline methods for comparative evaluation. The first one is a simple Logistic Regression (denoted as LR in the continuation) [17,18] fitted on the imbalanced training dataset. The second one is still a Logistic Regression, but this time fitted on the training dataset re-balanced using the traditional SMOTE algorithm (denoted as SMOTE+LR in the continuation) [3,19]. Table 2 presents the binary classification datasets used as test problems in our experimental study. The imbalance

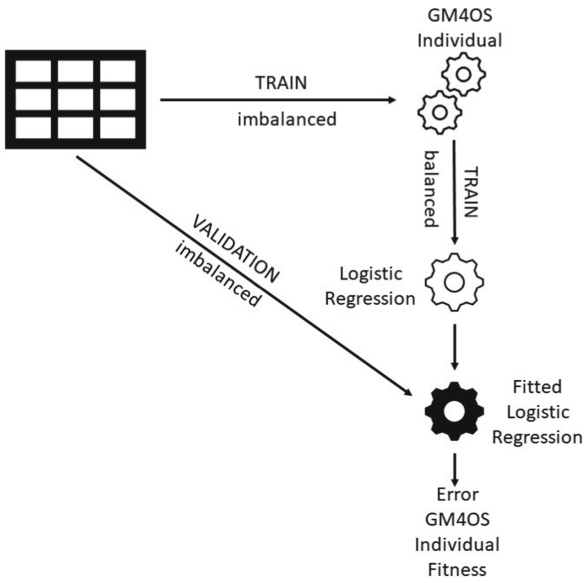


Fig. 2. Flowchart of the evaluation of a GM4OS individual. The initial dataset is split into training, test and validation. Then a GM4OS individual is used to produce enough new synthetic observations to balance the training set. A model, Logistic Regression in our experiments, is fitted on the balanced training set. Then, the fitted model is used to make predictions on the validation set. Finally, the loss between predictions and true labels on the validation set is used as fitness for the GM4OS individual.

ratio is calculated as the ratio between the number of observations of the majority class and the number of observations of the minority class. In this way, the imbalance ratio is, by definition, a positive number and, given that we are using imbalanced datasets, it is strictly larger than one in our test problems. All these datasets belong to the Penn Machine Learning Benchmarks (PMLB) library [20]. In general, several different performance measures that can be used to evaluate classification algorithms; among others, one may mention for instance precision, recall, accuracy, and F1-score. The choice of which metric to prioritize depends largely on the specific problem and its inherent characteristics. In our case, since we are addressing imbalanced classification problems, the F1-score of the minority class emerges as a particularly pertinent fitness metric for GM4OS. In fact, 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 [21].

Table 1. GM4OS parameters.

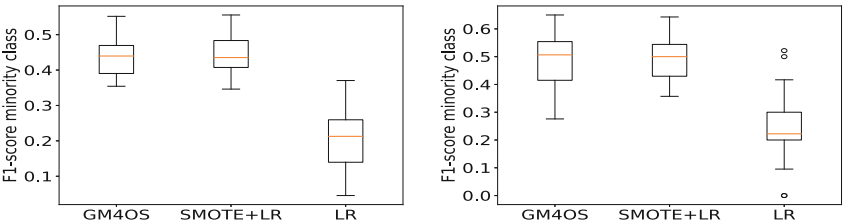
Parameter	Value
Population size	50
Generations	50
Mutation probability	0.2
Crossover probability	0.8
Elitism	True
Elite size	1
Selection algorithm	Tournament
Tournament size	2
GP initialization	Ramped-half-half
GA initialization	Random
GP constant set	$\{-1, 2, 3, 4, 5\}$
GP constant probability	0.3
GP function set	{add, sub, mul, div, mean}
GP crossover	Subtree single point swap crossover
GP mutation	Single node mutation
GA crossover	One point crossover
GA mutation	One point mutation
Maximum GP tree depth for initialization	8
Maximum GP tree depth for evolution	8

Table 2. 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

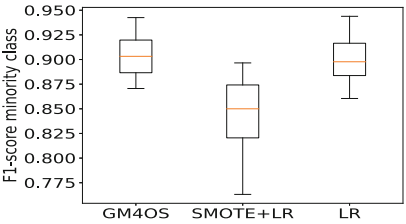
5 Experimental Results

All the results reported in this section are medians over 30 independent runs. At each run, the datasets are split randomly with uniform distribution into training, validation and test partitions, composed of 60%, 20% and 20% of the data observations, respectively. The proportion between majority and minority class observations is kept constant across the different splits. The same training/validation/test partition has been used for all the studied methods at each particular run. For LR and SMOTE+LR, the training and validation sets were joined and both were used as training set. The obtained experimental results are reported in Fig. 3. More particularly, Plot 3a (3b, 3c, 3d, 3e, 3f, 3g, 3h, 3i and 3j, respectively) reports the results for the flare (haberman, spect, spectf, ionosphere, hungarian, diabetes, hepatitis, appendicitis and analcatdata, respectively) test case. Each one of these plots represents a comparison between GM4OS, SMOTE+LR and LR. The comparison is done using box-plots of the F1-score of the minority class on the 30 different test sets. Table 3 reports the p-values of the Mann-Whitney U test for pairwise comparison of the methods, for five different metrics. The p-values are in bold when they indicate a statistically significant difference, using a significance level $\alpha = 0.05$, with the Bonferroni correction [22]. Also, the presence of the symbol ^{g} indicates that GM4OS outperforms the baseline, while the symbol ^{b} indicates the opposite. The last two rows of the table are a summary of the results for each measure(column). The first number, always positive, it indicates how many times GM4OS significantly outperforms LR+SMOTE or LR. While the second number, always negative, it indicates how many times GM4OS is significantly outperformed by LR+SMOTE or LR. For GM4OS, in both Fig. 3 and Table 3, the performance of the best individual at the last generation is used. Observing the F1-score metric for the minority class, we can notice that GM4OS is never significantly outperformed by the baselines. This outcome is corroborated by the box-plots presented in Fig. 3 and the p-values of the third column of Table 3. It is also possible to see that in four test cases out of ten GM4OS significantly outperforms LR, and in three test cases out of ten GM4OS significantly outperforms SMOTE+LR. The F1-score is the measure that we have used as fitness to guide the evolution of GM4OS. However, it is also interesting to show how GM4OS performs in terms of other metrics. Looking at the p-values of the accuracy, shown in Table 3, it is possible to observe that LR significantly outperforms GM4OS in four test cases out of ten, while having comparable performance in the remaining six cases. This is probably due to the fact that accuracy is a measure that can be misleading when working with imbalanced datasets. A model that predicts the majority class for every instance can be a very poor quality model, but still achieve a high accuracy when data is imbalanced. For the other studied metrics, it is difficult to identify a specific pattern. Depending on the test case, GM4OS can significantly outperform, be significantly outperformed or have comparable performance to the baseline models, as shown by the p-values presented in Table 3. Finally, Fig. 4 presents the evolution of the test fitness of the best individual of GM4OS along generations, compared to the one of the two baseline algorithms, namely LR and

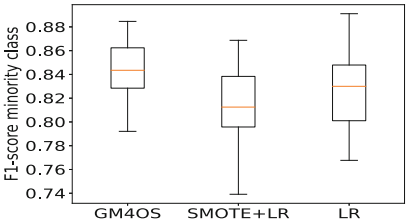


(a) flare

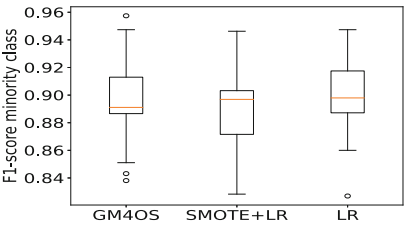
(b) haberman



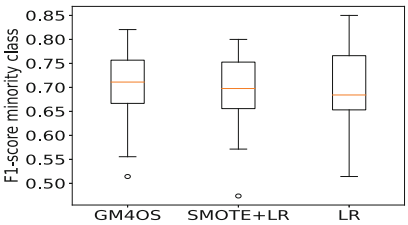
(c) spect



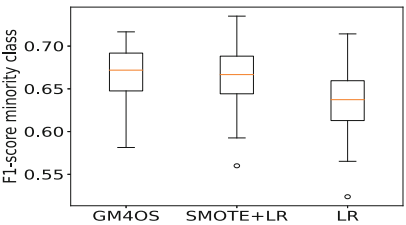
(d) spectf



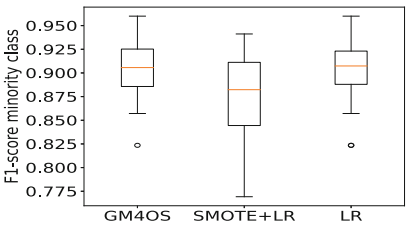
(e) ionosphere



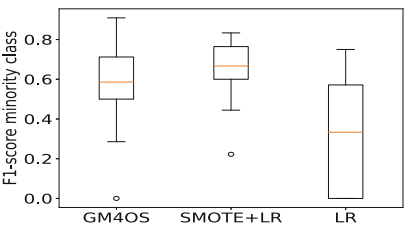
(f) hungarian



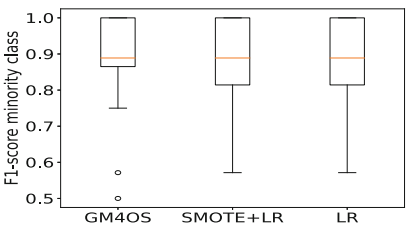
(g) diabetes



(h) hepatitis



(i) appendicitis



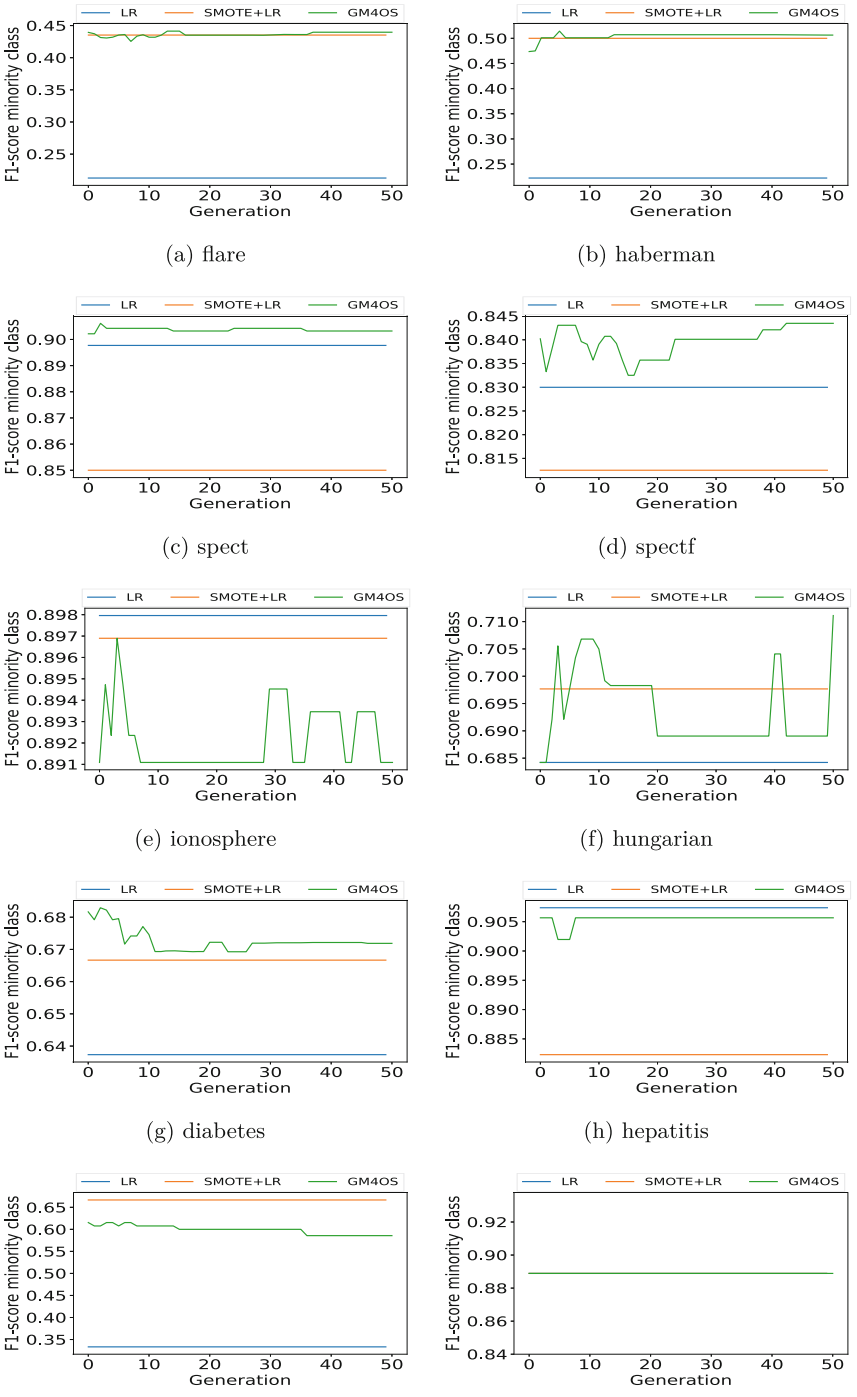
(j) analcatdata

Fig. 3. Box-plots of the F1-score of the minority class of GM4OS against the baselines for all the datasets.

Table 3. p-values of the Mann-Whitney U test [23] for different evaluation metrics of GM4OS against the baselines. In bold when the p-value indicates a statistically significant difference. Symbol ^{g} indicates that GM4OS outperforms the baseline, while symbol ^{b} indicates the opposite. The last two rows are a summary of the results for each measure(column). The first number, always positive, it indicates how many times GM4OS significantly outperforms LR+SMOTE or LR. While the second number, always negative, it indicates how many times GM4OS is significantly outperformed by LR+SMOTE or LR.

Problem	Baseline	F1-score minority class	recall minority class	precision minority class	recall majority class	precision majority class	accuracy
flare	LR+ SMOTE	0.695	0.21	0.137	**0.002**^{b}	0.663	0.041
	LR	**3.29e − 11**^{g}	**4.65e − 6**^{g}	**2.88e − 04**^{b}	**1.55e − 8**^{b}	**4.05e − 5**^{g}	**5.99e − 7**^{b}
haberman	LR+ SMOTE	0.888	0.243	0.025	**0.005**^{b}	0.739	0.023
	LR	**8e − 9**^{g}	**1.11e − 5**^{g}	**0.001**^{b}	**5.62e − 08**^{b}	**8.14e − 4**^{g}	**1.84e − 4**^{b}
spect	LR+ SMOTE	**1.61e − 8**^{g}	**5.72e − 18**^{b}	**4.77e − 7**^{g}	**2.97e − 18**^{g}	**4.89e − 12**^{b}	**1.67e − 4**^{g}
	LR	0.437	**1.24e − 12**^{b}	**1.46e − 4**^{g}	**5.01e − 12**^{g}	**8.89e − 8**^{b}	**0.011**^{b}
spectf	LR+ SMOTE	**8.99e − 4**^{g}	**9.88e − 9**^{b}	**0.015**^{g}	**1.9e − 9**^{g}	**4.03**^{b}	0.222
	LR	0.45	**6.52e − 7**^{b}	**0.015**^{g}	**1.57e − 8**^{g}	**2.93e − 05**^{b}	0.128
ionosphere	LR+ SMOTE	0.662	**0.002**^{b}	**2.63e − 6**^{g}	**3.4e − 7**^{g}	**0.006**^{b}	0.599
	LR	0.558	**0.006**^{b}	**1.79e − 4**^{g}	**3.22e − 5**^{g}	**0.1**^{b}	0.208
hungarian	LR+ SMOTE	0.721	0.92	0.894	0.815	0.947	0.699
	LR	0.693	0.811	0.524	0.55	0.781	0.234
diabetes	LR+ SMOTE	0.673	0.191	**0.008**^{b}	0.192	**0.008**^{b}	0.15
	LR	**0.001**^{g}	**2.63e − 7**^{b}	0.041	**2.63e − 7**^{b}	**0.041**^{b}	**1.16e − 5**^{b}
hepatitis	LR+ SMOTE	**0.012**^{g}	**1.47e − 5**^{b}	**9.22e − 4**^{b}	**1.1e − 6**^{g}	**3.72e − 4**^{b}	**0.016**^{g}
	LR	0.898	**0.001**^{b}	**0.015**^{b}	**1.5e − 4**^{g}	**0.008**^{b}	0.092
appendicitis	LR+ SMOTE	0.039	**0.01**^{b}	0.344	0.036	0.022	0.634
	LR	**1.11e − 4**^{g}	0.2	0.838	0.112	0.313	0.586
analcatdata	LR+ SMOTE	0.895	0.322	0.443	0.402	0.34	0.856
	LR	0.315	0.787	0.676	0.779	0.767	0.99
summary	LR+ SMOTE	+3, 0	+2, −5	+3, −2	+4, −2	0, −5	+2, 0
	LR	+4, 0	0, −5	+3, −3	+4, −3	+2, −5	0, −4

SMOTE+LR, which are presented as horizontal straight lines in the figure. More specifically, Plot 4a (4b, 4c, 4d, 4e, 4f, 4g, 4h, 4i and 4j, respectively) reports the results for the flare (haberman, spect, spectf, ionosphere, hungarian, diabetes, hepatitis, appendicitis and analcatdata, respectively) test case.
From these plots, it is possible to observe that GM4OS outperforms LR in the entire evolution process in six out of ten total test cases. Similarly GM4OS



(a) flare (b) haberman

(c) spect (d) spectf

(e) ionosphere (f) hungarian

(g) diabetes (h) hepatitis

(i) appendicitis (j) analcatdata

Fig. 4. Evolution of the test fitness of GM4OS, compared to the results returned by LR and SMOTE+LR, over the studied datasets.

outperforms SMOTE+LR over all the generations in four out of ten total test cases. In the case of Plot 4j, reporting results on the analcatdata test problem, all three models have the same performance, around 0.89, and GM4OS has it starting from the beginning of the evolution (and this is why the figure looks like a unique horizontal line). This result is coherent with the box-plot shown in Fig. 3j, where the median of all the models is around 0.89 as well.

6 Conclusions and Future Work

This paper introduced the Genetic Methods for OverSampling (GM4OS), an evolutionary oversampling approach for imbalanced binary classification problems. In real-world binary classification tasks, imbalancing between the classes represents a recurrent issue. In those cases, in fact, classification models typically struggle to classify correctly the observations belonging to the minority class. A popular approach to tackle this issue is to add synthetic observations to the minority class, commonly referred to as oversampling. One of the most known oversampling approaches is the Synthetic Minority Oversampling Technique (SMOTE) [3]. SMOTE creates new observations by selecting two existing minority class observations, and sampling a new synthetic point from the straight line segment that connects them. However, this approach relies heavily on the set of points that are used for resampling. 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 individuals, in fact, are represented as pairs of objects, one of which is a GP-like function, while the other one is a GA-like string. The GP part strongly resembles the recently introduced vectorial GP approach [15,16] and combines two vectors (existing observations) to generate a new single vector (synthetic observation), while the GA part controls which existing minority class observations will constitute the resampling set. GM4OS was experimentally compared with a simple Logistic Regression (LR) [17] and SMOTE combined with LR, on ten imbalanced binary classification test problems, taken from the Penn Machine Learning Benchmarks library [20]. The experimental results show that, on all the studied test problems, GM4OS is able to find models that have an F1-score on the minority class that is better, or at least comparable, to the baseline models.

Despite the positive experimental outcomes achieved, a significant scope remains open to future research. One such avenue involves the adaptation of the GM4OS framework to address multi-class classification problems. Another area of investigation pertains to the exploration of alternative fitness metrics for GM4OS, distinct from the F1-score utilized in this study. Specifically, the pursuit of metrics that have demonstrated efficacy for GP applied to imbalanced classification tasks, as evidenced for instance in [12,13], holds significant promise. Multi-objective optimization, using several fitness functions at the same time, also deserves investigation. An additional idea for future research is inspired by the observation that the GP component within GM4OS is susceptible to the issue of bloat, a common occurrence in GP. While the present

work implemented a strategy to restrict tree depth to a maximum of 8, a myriad of other strategies have been advanced to mitigate bloat. Notably, existing references suggest that the introduction of a dynamic GP population size effectively alleviates bloat, thereby reducing computational overhead while maintaining excellent performance levels, as documented for instance in [24–26]. In light of these findings, the incorporation of dynamic population sizing into GM4OS presents itself as an intriguing avenue for future development. Furthermore, as a forthcoming research endeavor, it is imperative to acknowledge that the present study employed a Logistic Regression model to evaluate the oversampling function/resampling set pair, due to its computational efficiency [17]. Nevertheless, the framework remains adaptable to experimentation with alternative models, such as for instance Decision Tree classifiers [27] or many others. In this study, we conducted a comparative analysis to assess the performance of GM4OS against an alternative oversampling technique, namely SMOTE (Synthetic Minority Over-sampling Technique). Our investigation aimed to elucidate the effectiveness of these oversampling approaches in addressing the challenge of class imbalance within classification tasks. Additionally, our future research endeavors will encompass further comparisons with alternative oversampling methodologies, including Adasyn [10] and borderline-SMOTE [9]. Lastly, an interesting prospect for future research entails the extension of GM4OS to encompass synthetic data generation. This extension can be realized through the modification of the resampling set governed by the GA component of GM4OS, extending its influence to the entire dataset.

Acknowledgments. This work was supported by national funds through FCT (Fundação para a Ciência e a Tecnologia), under the project - UIDB/04152/2020 - Centro de Investigação em Gestão de Informação (MagIC)/NOVA IMS.

References

1. Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor. **6**, 20–29 (2004)
2. Chawla, N., Japkowicz, N., Kołcz, A.: Editorial: special issue on learning from imbalanced data sets. SIGKDD Explor. **6** , 1–6 (2004). https://doi.org/10.1145/1007730.1007733
3. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. J. Artif. Int. Res. **16**(1), 321–357 (2002)
4. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press (1996)
5. Poli, R., Langdon, W.B., McPhee, N.F.: A Field Guide to Genetic Programming. Lulu Enterprises, UK Ltd (2008)
6. Ali, A., Shamsuddin, S.M., Ralescu, A.: Classification with class imbalance problem: a review **7**, 176–204 (2015)
7. Huang, J., Ling, C.: Using AUC and accuracy in evaluating learning algorithms. IEEE Trans. Knowl. Data Eng. **17**(3), 299–310 (2005). https://doi.org/10.1109/TKDE.2005.50

8. Gosain, A., Sardana, S.: Handling class imbalance problem using oversampling techniques: a review. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 79–85 (2017). https://doi.org/10.1109/ICACCI.2017.8125820
9. 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 2005), Springer, Heidelberg (2005), pp. 878–887. https://doi.org/10.1007/11538059_91
10. He, H., Bai, Y., Garcia, E.A., 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), pp. 1322–1328 (2008). https://api.semanticscholar.org/CorpusID:1438164
11. Frank, F., Bacao, F.: Advanced genetic programming vs. state-of-the-art automl in imbalanced binary classification. Emerg. Sci. J. **7**(4), 1349–1363 (2023). https://doi.org/10.28991/ESJ-2023-07-04-021
12. 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), pp. 2779–2786 (2019). https://doi.org/10.1109/CEC.2019.8789974
13. Kumar, A.: A new fitness function in genetic programming for classification of imbalanced data. J. Exp. Theor. Artif. Intell. 1–13 (2022). https://doi.org/10.1080/0952813X.2022.2120087
14. Karia, V., Zhang, W., Naeim, A., Ramezani, R., Gensample: a genetic algorithm for oversampling in imbalanced datasets. arXiv preprint arXiv:1910.10806 (2019)
15. 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. (eds.) Genetic Programming: 22nd European Conference, EuroGP 2019, Held as Part of EvoStar 2019, Leipzig 24–26 April 2019, Proceedings, pp. 213–227. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-16670-0_14
16. 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. **89**, 106097 (2020). https://doi.org/10.1016/j.asoc.2020.106097
17. Cox, D.R.: The regression analysis of binary sequences. J. Roy. Stat. Soc.: Ser. B (Methodol.) **20**(2), 215–232 (1958)
18. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
19. Lemaître, G., Nogueira, F., Aridas, C. K.: Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. J. Mach. Learn. Res. **18**(17), 1–5 (2017)
20. Romano, J.D., et al.: Pmlb v1.0: an open source dataset collection for benchmarking machine learning methods. arXiv preprint arXiv:2012.00058v2 (2021)
21. Ferrer, L.: Analysis and comparison of classification metrics. arXiv preprint arXiv:2209.05355 (2023)
22. Bonferroni, C.: Teoria statistica delle classi e calcolo delle probabilità, Pubblicazioni del R. Istituto superiore di scienze economiche e commerciali di Firenze, Seeber (1936)
23. Mann, H.B., Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other. Annal. Math. Statist. **18**(1), 50–60 (1947). https://doi.org/10.1214/aoms/1177730491

24. Fernandez, F., Vanneschi, L., Tomassini, M.: The effect of plagues in genetic programming: a study of variable-size populations. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R., Costa, E. (eds.) Genetic Programming, pp. 317–326. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36599-0_29
25. Rochat, D., Tomassini, M., Vanneschi, L.: Dynamic size populations in distributed genetic programming. In: Keijzer, M., Tettamanzi, A., Collet, P., van Hemert, J., Tomassini, M. (eds.) Genetic Programming: 8th European Conference, EuroGP 2005, pp. 50–61. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31989-4_5
26. Farinati, D., Bakurov, I., Vanneschi, L.: A study of dynamic populations in geometric semantic genetic programming. Inf. Sci. **648**, 119513 (2023). https://doi.org/10.1016/j.ins.2023.119513
27. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and regression trees. Biometrics **40**, 874 (1984). https://api.semanticscholar.org/CorpusID:29458883