GBRF: A novel framework for encoding user-preferences in imbalanced data distributions via genetic optimization

Miguel Carvalho[1][0000−0001−9906−9128] (✉), Armando Pinho[1][0000−0002−9164−0016], and Susana Brás[1][;0000−0001−8650−9219]

IEETA, DETI, LASI, University of Aveiro, Aveiro, 3810-193, Portugal
{miguelacarvalho,ap,susana.bras}@ua.pt

Abstract. Resampling techniques are widely used by researchers and practitioners to address class imbalance due to their adaptability across diverse classification tasks. However, they inherently lack the ability to enforce user-defined preferences regarding model behavior after training, a feature typically exclusive to cost-sensitive learning frameworks or prediction post-processing techniques. This limitation is particularly critical in high-stakes applications, such as in the medical domain, where maximizing minority class accuracy while minimizing false negatives is essential. To overcome this constraint, we introduce the Genetic Beta Resampling Framework (GBRF), a novel, customizable and computationally efficient resampling framework that integrates user preferences into the process of synthetic data generation. GBRF leverages Genetic Algorithms to optimize two probability mass functions (PMFs) that govern the sampling probabilities of different instance groups, enabling synthetic data generation and/or instance removal. Consequently, GBRF can function as an hybrid sampling, oversampling or undersampling technique. User preferences are encoded through a parameter, β, which controls the trade-off between precision and recall. Comprehensive experiments on 60 OpenML datasets demonstrate that GBRF effectively embeds user preferences into data distributions, thus shaping model behavior accordingly. It consistently outperforms state-of-the-art resampling techniques, such as SMOTE-IPF and ProWSyn, as well as cost-sensitive classifiers, even when integrated with various classification models. Furthermore, by employing a non-instance-wise genetic optimization approach, GBRF significantly reduces the search space, achieving faster convergence to optimal solutions. Finally, since synthetic data generation is governed by two PMFs, GBRF provides an intuitive and transparent mechanism for understanding how data is generated. Code available at: https://github.com/MiguelCarvalhoPhD/GBRF.

Keywords: Class Imbalance · Genetic Optimization · Resampling.

Supplementary Information: The online version contains supplementary material available at `link_to_supplementary_materials`.

1 Introduction

Class imbalance refers to the unequal distribution of the target variable in supervised learning tasks, where the underrepresented class, often termed the minority class in binary classification problems, typically experiences reduced predictive performance [4]. This skewed data distribution has far-reaching consequences in applications where correctly classifying the minority class holds critical importance, such as fraud detection, rare medical condition diagnosis, forecasting natural disasters, and fields including chemical and biochemical engineering, IT security, agriculture, and emergency management [21]. This problem arises from the assumption embedded in most learning algorithms that all data points are equally important, leading to accuracy-oriented optimization strategies with uniform misclassification penalties across all classes [17]. Consequently, classifiers often prioritize majority class performance, as correctly classifying majority instances while disproportionately misclassifying minority samples typically incurs lower training loss [17]. However, it is essential to recognize that class imbalance alone does not affect model performance [20]. Note that in cases where the classes are easily linearly separable with large margins, optimal training loss can still be achieved without the need to bias the decision boundary in favor of the majority class [20]. As such, class imbalance becomes truly problematic when combined with other existing data adverse characteristics (also referred to as data irregularities), such as class overlap, existence of small disjuncts, high dimensionality, and noise [20].

Given the widespread incidence of class imbalance across diverse sectors and its significant impact on model performance, a wide range of methods has been developed to address this issue, which can be broadly divided into data-level and algorithmic-level approaches [10,4]. Data-level approaches involve modifying the data distribution to magnify the relative importance of the minority class during model training, typically by generating synthetic minority samples (oversampling) or extracting majority samples (undersampling) until identical class frequencies are attained. Contrarily, algorithmic level approaches focus on altering existing learning algorithms to mitigate the bias towards the majority class, without altering the training data [19,7]. Strategies encompass adjusting the decision threshold, training class-specific classifiers, weighting-based approaches, among others [7]. Data-level approaches are more widely utilized by researchers and practitioners given their ability to be utilized alongside any other learning algorithm or classification context, contrasting with algorithmic-level approaches which are constrained to models capable of being adapted to imbalanced domains and often rely on expert knowledge for fine-tuning [19,4,16]. However, a key limitation of resampling methods is their inability to directly encode user-specific objectives for model behavior. As Branco et al. highlighted in [4], determining the optimal degree of undersampling based on user preferences remains an open research problem. This issue is mainly addressed through cost-sensitive classifiers, which provide a structured framework for incorporating class-specific costs into the learning process [4].

To overcome this fundamental limitation of resampling approaches, we propose an explainable, customizable, and computationally efficient framework for synthetic dataset generation. This framework, designated Genetic Beta Resampling Framework (GBRF), ensures that models trained on the generated artificial data adhere to predefined user preferences, which are encoded in a parameter, β. This parameter regulates the trade-off between recall and precision: higher β values prioritize recall, favoring synthetic datasets that reduce false negatives at the expense of false positives, and vice versa. In practice, penalizing false negatives more heavily encourages synthetic sample generation within the decision boundary, effectively expanding underrepresented regions and reducing minority class misclassifications. However, selecting an optimal resampling approach is not solely dependent on user preferences but is also influenced by the aforementioned intrinsic dataset characteristics. As suggested by the No Free Lunch Theorem, no universally optimal resampling strategy exists [22], and the connection between data irregularities and effective resampling protocols remains an open research question [20]. To provide an adaptive solution, our framework employs Genetic Algorithms to optimize two probability mass functions (PMF) that govern the sampling probability of different instance groups, determining whether samples are synthesized or eliminated.

The main contributions of this work are manifold:

- We present a mathematical analysis, leveraging Bayes' Theorem, to demonstrate how selectively resampling minority class instances with different class-conditional densities can alter the decision boundary and subsequently influence model behavior. This analysis substantiates the validity of our resampling strategy.
- We propose a novel resampling framework designed to determine optimal resampling strategies for a wide range of datasets and user-specified preferences—an objective that, until now, has been achievable only through algorithmic changes of existing models. The framework's efficacy is empirically demonstrated across 60 OpenML benchmarks and further supported by a visual inspection of the generated synthetic datasets. The proposed framework is also fully customizable, enabling users to specify custom functions for sample grouping and generation, which are then optimized by GBRF.
- By deviating from conventional instance-wise optimization in genetic resampling, we ensure that the search space of the genetic algorithm scales with the number of generated sample groups rather than increasing exponentially with dataset size, enabling faster convergence to high-quality solutions.

2 Related Work

Resampling Tabular Datasets to Address Class Imbalance. The majority of recent research on resampling in tabular data has focused on addressing the limitations of the Synthetic Minority Oversampling TEchnique (SMOTE) to develop more robust oversampling methods [9]. SMOTE is an interpolation-based data generation technique that operates as follows [6]: Given a training

dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, where $\mathrm{NN}_j(\mathbf{x}_i)$ denotes the j-th minority-class nearest neighbor of $\mathbf{x}_i$, the generation of a synthetic instance, $\mathbf{x}_{\mathrm{new}}$, is formalized as:

$$x_{new} = x_i + \alpha(x_i - NN_j(x_i)), \quad \alpha \sim \mathcal{U}(0,1) \tag{1}$$

with $\mathcal{U}(0,1)$ denoting a uniform random variable from the interval $[0,1]$ This procedure is repeated until the desired class ratio is achieved. Importantly, SMOTE applies uniform sampling probabilities across all minority class instances, which introduces three major issues: (i) it fails to address within-class imbalance, as low-density regions remain underrepresented post-oversampling, (ii) it may oversample outliers, potentially propagating existing noise and degradating classifier performance, and (iii) it does not consider the structure of the majority class, which can lead to class overlap. To overcome these limitations, numerous SMOTE variants have been developed, each introducing specific modifications to the original algorithm [16]. As categorized in [9], these variations include (1) initial selection of samples for oversampling, (2) adaptive generation of synthetic instances, (3) integration with undersampling, among others. While these approaches introduce distinct methodological refinements, they typically share a commom principle: they aim to identify and quantify existing data irregularities and adapt the resampling process accordingly. However, fully characterizing these data irregularities remains an unresolved research challenge, particularly given the intricate and multifaceted nature of class overlap (see [20] for a detailed discussion). This limitation imposes fundamental constraints on conventional resampling methods, which rely on predefined metrics to adjust different aspects of the resampling algorithm. This is major factor dictating why we adopt a metaheuristic framework capable of optimizing resampling protocols dynamically, eliminating the dependence on handcrafted dataset characterization metrics and enabling a more adaptable and generalizable solution.

Genetic Algorithms in Resampling Genetic Algorithms. Genetic Algorithms are population based search algorithms that model complex systems by constructing a simulated environment that mimics natural evolution, allowing them to solve problems that do not have a well-defined efficient solution [5]. The fundamental components of a GA include chromosome encoding, genetic operators (selection, mutation, and crossover), and a fitness function that quantifies the suitability of each candidate solution. GA typically begins with a randomly initialized population, where each chromosome encodes a candidate solution. Fitness evaluation determines solution quality, guiding selection and crossover processes that generate offspring by combining traits of the fittest individuals [5]. Mutation introduces genetic diversity by altering genes, preventing premature convergence to local optima and enhancing exploration of the search space. Selection mechanisms determine which individuals advance to the next generation, and this iterative process continues until a stopping criterion or predefined number of generations is met [1].

In resampling applications, Genetic Algorithms are primarily used for undersampling [13,15,8]. Typically, each majority-class sample is represented by a

boolean gene indicating its inclusion or exclusion from the final dataset. Such representations lead to exponentially expanding search spaces as dataset size increases, making large-scale applications computationally demanding and reducing convergence speed to high-quality solutions. Common fitness functions include AUC, G-Mean, and AUC-ROC, evaluated on the training set [14,15]. However, assessing performance on the same dataset used for training often leads to unreliable generalization estimates. In [11], the authors propose stratified partitioning to create a validation dataset, though this approach risks overfitting to that specific partition.

3 Proposed Method

3.1 Overview

The proposed algorithm follows a four-stage process: (1) Framework instantiation, where sample grouping and sample generation algorithms are defined, a β parameter is introduced to control the trade-off between precision and recall according to user-defined preferences, and each PMF is assigned the function of either generating or removing samples (2) Genetic algorithm initialization encodes PMFs as chromosomal structures and generates all the required synthetic samples for subsequent optimization. (3) Genetic optimization identifies the optimal PMFs, the ratio of samples generated/removed by each PMF and the extent of outlier removal to be conducted, ensuring an adaptive resampling strategy. (4) Synthetic dataset generation constructs the final resampled dataset using the fittest solution identified through genetic optimization.

As the core premise of this resampling approach is the selective generation and removal of samples in specific regions of the feature space to encode user-defined preferences on the data distribution, it is essential to first demonstrate theoretically that this mechanism can meaningfully shift a model's decision boundary.

3.2 Theoretical considerations on encoding user-preferences in the data distribution

Setup and Notation. Let D be a one-dimensional dataset with two classes, $y \in \{-, +\}$, with a high imbalance degree, that is, $n_+ \gg n_-$, with n_+ and n_- representing the number of majority and minority samples, respectively. Let the minority samples follow $N(\mu_-, \sigma)$ and majority samples follow $N(\mu_+, \sigma)$, where $\mu_- < \mu_+$. Let the class priors be $\pi_- = p(y = -)$ and $\pi_+ = p(y = +)$ such that $\pi_- + \pi_+ = 1$ and $\pi_- \ll \pi_+$. The optimal Bayes decision boundary, denoted by k, is defined as the unique solution to:

$$p(x \mid y = -)\,\pi_- = p(x \mid y = +)\,\pi_+. \tag{2}$$

where $p(x \mid y = +)$ and $p(x \mid y = -)$ represent the class-conditional densities for the majority and minority class, respectively. The uniqueness of k follows from the fact that the likelihood ratio $\lambda(x) = \frac{p(x|y=-)}{p(x|y=+)}$ is strictly monotonic in x for these Gaussian densities.

Resampling the minority class. Let us define a nonnegative, measurable weighting function $w(x)$ that emphasizes regions of interest. For instance, one may choose $w(x)$ such that it is large for $x > \mu_-$, thereby upweighting the tail of the minority class. We assume that $0 < \int_{-\infty}^{\infty} p(t \mid y = -)\, w(t)\, dt < \infty$. The normalized re-weighted minority density is:

$$\widetilde{p}(x \mid y = -) = \frac{p(x \mid y = -)\, w(x)}{\displaystyle\int_{-\infty}^{\infty} p(t \mid y = -)\, w(t)\, dt} \tag{3}$$

This mimics the effect of adding synthetic samples to the region $x > \mu_-$. Moreover, assume that re-sampling adjusts the minority class prior to $\widetilde{\pi}_-$, matching the class prior of the majority class, that is, $\widetilde{\pi}_- = \widetilde{\pi}+$. Thus, the joint densities after re-sampling become:

$$\widetilde{p}(x, y = -) = \widetilde{p}(x \mid y = -)\, \widetilde{\pi}_- \tag{4}$$

$$\widetilde{p}(x, y = +) = p(x \mid y = +)\, \widetilde{\pi}_+ \tag{5}$$

New Decision Boundary. A classifier trained on the re-sampled data will have its decision boundary $\widetilde{k}$ defined by $\widetilde{p}(\widetilde{k} \mid y = -)\, \widetilde{\pi}_- = p(\widetilde{k} \mid y = +)\, \widetilde{\pi}_+$. Substitute the expression for $\widetilde{p}(x \mid y = -)$:

$$\widetilde{\pi}_- \frac{p(\widetilde{k} \mid y = -) w(\widetilde{k})}{Z} = p(\widetilde{k} \mid y = +) \widetilde{\pi}_+ \tag{6}$$

with $Z = \int_{-\infty}^{\infty} p(t \mid y = -) w(t)\, dt$. Rearranging gives:

$$\frac{p(\widetilde{k} \mid y = -)}{p(\widetilde{k} \mid y = +)} = \frac{Z\, \widetilde{\pi}_+}{w(\widetilde{k})\, \widetilde{\pi}_-} \tag{7}$$

Because the likelihood ratio $\lambda(x) = \frac{p(x|y=-)}{p(x|y+)}$ is strictly monotonic, equation 7 admits a unique solution for $\widetilde{k}$. Notice that if $w(\widetilde{k})$ increases (as is the case when $\widetilde{k}$ lies in a region where $w(x)$ is large), then the right-hand side decreases. To restore the equality, $\widetilde{k}$ must shift to a region where $\lambda(x)$ is naturally lower. For the Gaussian densities under consideration (with $\mu_- < \mu_+$), $\lambda(x)$ is decreasing in x, so the new decision boundary $\widetilde{k}$ shifts rightward relative to the original boundary k. This reasoning highlights how, depending on the applied $w(x)$, different shifts of the decision boundary will take place.

Impact on the False-Positive Probability. To finalize this analysis, let us consider the impact on the behavior of the resulting classifier. The false-positive probability under the original (true) distribution for class $y = +$ is:

$$\widetilde{P}(\mathrm{FP}) = \int_{-\infty}^{k} p(x \mid y = +)\, \pi_+\, dx \tag{8}$$

Since upweighting the tail causes k to shift to a higher value, $\widetilde{k}$, the integration region $[-\infty, \widetilde{k})$ becomes bigger. Therefore, more true "+" points fall into the region where the classifier predicts "−", and the false-positive probability of class $y = +$ increases. The same line of reasoning can be applied to the probability of false negatives, demonstrating how selectively resampling regions of the minority class can encode preferences on the data distribution.

3.3 GBRF: Algorithm Instantiation

As outlined in Section 3.1, initializing the algorithm requires specifying several parameters, notably: (i) functions that construct sample groups with similar characteristics (ii) functions that govern the synthetic sample generation process, and (iii) the β parameter dictating user-preferences. Lastly, it is important to discuss how synthetic dataset generation is mediated by the two distinct probability mass functions.

Sample Grouping Functions. Prior to introducing the two sample grouping functions tested in this work, let us formally describe them. Let $X \subset \mathbb{R}^d$ be the feature space and $Y = \{0, 1\}$ the class labels. A sample grouping function

$$g : X_{min} \to G, \quad G = \{G_1, \ldots, G_k\} \tag{9}$$

partitions X_{min}, with $X_{min} = \{x \in X : y(x) = 1\}$, into disjoint sets, i.e., $X_{min} = \bigcup_{i=1}^{k} G_i$ with $G_i \cap G_j = \emptyset$ for $i \neq j$. The first utilized sample grouping function, g_1, partitions samples by analyzing their local neighborhood, specifically by counting the number of majority-class samples among their five nearest neighbors, denoted by $N_5(x)$, thereby forming at most six distinct sample groups: $g_1(x) = G_{m_5(x)+1}$, where $m_5(x) = |\{x_j \in N_5(x) \mid y_j = 0\}|$. Note that, by Cover and Hart's theorem (1967), k-NN classification error is upper bounded by twice the error of a Bayes classifier, making nearest-neighbor analysis a reliable non-parametric estimate of class-conditional density [18]. As such, samples heavily surrounded by majority class samples tend to lie near the decision boundary, providing avenues from which user-preferences can be encoded in the distribution as suggested in Section 3.2.

For the second sample grouping function, g_2, we adopt the ProWsyn methodology, which partitions the minority class based on its global structure relative to the majority class, as opposed to the more localized analysis provided by g_1 [3]. The process iteratively constructs N_{sets} disjoint subsets of minority samples. Initialization starts with an empty set G_1, which is populated with all minority samples appearing among the five nearest neighbors of any majority-class sample. These selected samples are then removed from the dataset, and the process repeats for N_{sets} iterations [3]. Furthermore, it is important to highlight that algorithm instantiation requires specifying whether sample grouping functions are intended for instance removal or synthetic generation. When configured for removal, the method is applied to the majority class instead of the minority class, wherein majority samples are grouped into distinct subsets, enabling subsequent instance elimination.

Sample Generation Functions. We implement two procedures for sample generation. First, the standard SMOTE procedure is implemented, which is denoted by $f_{\text{SMOTE}}(G_k)$, where $f_{\text{SMOTE}} : G_k \rightarrow X^*$, with X^* representing the synthetic data space [6]. Here, a minority sample is randomly selected from an existing group, G_k, from which one of its five nearest minority-class neighbors is utilized for interpolation, as described in Equation 1 [6]. Secondly, the ProWsyn-based procedure is employed, which is represented as $f_{ProWsyn}(G_k)$ [3]. Instead of restricting the selection to the five nearest neighbors, all elements of the grouping set G_k are considered for interpolation. This broader neighborhood yields greater synthetic sample diversity and increases the framework's degrees of freedom to impose user-preferences. In summary, our implementation couples the grouping function g_1 with the SMOTE-based generation function f_{SMOTE} and the grouping function g_2 with the ProWsyn-based generation function f_{ProWsyn}. Note that, as previously outlined, if either of the sample grouping functions is designated for sample removal, the sample generation procedure is not conducted and random samples are removed from the generated groups, allowing the framework to act as an oversampling, hybrid sampling and undersampling method depending on the chosen configuration. Lastly, we stress the fact that, since sample generation functions are dependent on sample groups, g_1 can be applied in conjunction with $f_{ProWsyn}$ and g_2 with f_{SMOTE}, illustrating the framework's flexibility to incorporate any of the vast resampling methodologies described in the literature [9]. However, it should be noted that most SMOTE variants assign sample-wise instance sampling probabilities, as opposed to group-wise sampling probabilities [9], but this can be easily addressed by discretizing the sampling distribution.

Dual PMF-based Synthetic Dataset Generation. Within the proposed framework, we define two probability mass functions, p_1 and p_2, each mapping the set of sample groups G to the interval $[0, 1]$. That is, for each function $p_i : G \rightarrow [0, 1]$ (with $G = \{G_1, G_2, \ldots, G_k\}$ obtained via g_1 and g_2), we have

$$\forall j, \quad p_i(G_j) \geq 0 \quad \text{and} \quad \sum_j p_i(G_j) = 1 \tag{10}$$

These functions govern the sampling probabilities for each group. Within each group, samples are then randomly selected for either generation—using the corresponding sample generation function—or removal, based on predetermined parameters. Note that, p_1 is used alongside g_1 and f_{SMOTE}, whereas p_2 is utilized in conjunction with g_2 and $f_{ProWSyn}$.

Defining User-preferences. The encoding of user-preferences is conducted through the trade-off between precision, P, and recall, R, and is inspired by [2], which states that the relative importance a user attaches to precision and recall is the R/P ratio at which $\frac{dE}{dR} = \frac{dE}{dP}$, with E being a measure of effectiveness based on precision and recall, that is, $E = E(P, R)$. Practically, this premise

leads to the $F\beta$-score formulation:

$$F\beta - \text{score} = \frac{(1+\beta^2)PR}{(\beta^2 P) + R} \tag{11}$$

where β times more importance is attached to recall than precision. Naturally, by assigning β values higher than 1, higher importance is given to recall, which then means that minimizing false negatives will be prioritized as opposed to false positives, and vice-versa. This preference is then embedded into the genetic algorithm through the fitness function. Specifically, the fitness function, which directs the optimization process, is formulated as the average $F\beta$-score obtained via 5-fold cross-validation, where resampled data is used for training and a partition of the original data is reserved for testing. This approach is deliberately adopted to avoid evaluation bias associated with the inclusion of synthetic data in the test set, as is prevalent in many current methods [14,15].

3.4 Genetic Optimization.

Upon establishing the synthetic data generation mechanisms and the corresponding fitness function, the next step is to determine how the PMFs (i.e. p_1 and p_2) are encoded as chromosomes to enable genetic optimization, along with the selection of the appropriate genetic operators.

PMF Genetic Encoding. Within each chromosome, one gene is allocated for each sample group per probability mass function, in addition to a gene that specifies the proportion of samples generated by each PMF and an optional gene regulating the percentage of outlier removal. Collectively, these genes provide all the necessary information for synthetic data generation. For instance, if g_1 generates six groups in a dataset and the first six genes are assigned equal values, all minority samples will have an equal probability of being selected for oversampling, thereby replicating the baseline SMOTE approach. Conversely, if the last two of these six genes assume significantly higher values than the others, the resulting behavior will resemble that of ADASYN [12]. Note that, this example implies both p_1 and p_2 are set for sample generation.

Genetic Operators. All the adopted genetic operators and corresponding parametrization are inspired in existing literature regarding genetic resampling [13,14,15,11,8]. In the proposed framework, a fixed population size of 24 chromosomes is maintained. At each generation, tournament selection ($k = 4$) is applied with replacement to select parent pairs for reproduction. Single-point crossover is applied with a probability of 0.75, followed by random mutation with a probability of 0.25. A total of 20 offspring are generated per generation, and the top 4 individuals from the current population are preserved via elitism. The algorithm terminates either when the average $F\beta$-score reaches 1 or upon completion of 300 iterations.

Pre-Optimization Sample Generation. Prior to initializing the genetic optimization process, all required samples are pre-generated. Specifically, N samples, with N being the total number of samples required to achieve the desired class ratio (typically 1:1), are produced for each sample group and for each PMF, ensuring that every potential convergence solution is fully supported by the necessary data. This design choice is based on experimental testing, which demonstrated that dynamically generating samples during optimization significantly degrades algorithm runtime. Moreover, using a consistent set of samples throughout the optimization enhances convergence speed by obviating the stochastic component of interpolation-based sample generation (as outlined in Equation 1).

3.5 Genetic Optimization and Synthetic Data Generation

After establishing all necessary functions and parameters, the genetic optimization process begins by initializing all genes with one of 20 equally spaced values between zero and one. This approach restricts gene space complexity, promotes rapid convergence, and mitigates overfitting. As a result, the proposed method has a substantially reduced search space, which depends only on the number of sample groups generated by g_1 and g_2. Each candidate solution is then normalized to ensure that the probability mass function properties are satisfied (Equation 10). Subsequently, samples are retrieved from the pre-generated data matrices and concatenated to form the synthetic dataset. This process is conducted for 5 train/test splits, where a model is trained on the resampled train data partition and fitness is computed by assessing model performance on the test partition. Finally, when the genetic optimization process terminates, the final dataset is generated based on the fittest solution using the same procedure.

4 Experiments

To comprehensively evaluate the effectiveness of our framework, GBRF, we address the following research questions:

- **RQ1**: Is GBRF capable of encoding user preferences in data distributions while achieving superior performance compared to state-of-the-art methods? Furthermore, can GBRF be used effectively alongside different classifiers, as is the case with standard resampling techniques?
- **RQ2**: Do the optimal probability mass functions obtained by GBRF conform to theoretical expectations?
- **RQ3**: What is the impact of the group-wise genetic optimization approach on the convergence speed of the genetic algorithm, and how does this translate into improvements in overall performance?

4.1 Experimental Setup

Datasets. We randomly selected 60 datasets from OpenML, all with an imbalance ratio above 3. The set includes 30 binary datasets and 30 obtained through

binary decomposition of imbalanced multiclass datasets (via One-vs-All). The binarization of multiclass datasets was intentionally employed to amplify the existing class imbalance. Table 1 presents the primary statistical characteristics of the datasets analyzed in this study. The datasets exhibit considerable heterogeneity in class imbalance, sample sizes, and feature dimensionality, thereby encompassing a broad spectrum of classification problems. This diversity facilitates a rigorous assessment of GBRF's efficacy.

Table 1. Mean, standard deviation, median and range for the utilized 60 benchmark datasets of OpenML.

Metric	**Mean ± Std**	**Median**	**Range**
Imbalance Ratio	54.2 ± 117.5	9.0	[3.0, 695.2]
Number of Samples	2433.5 ± 1965.1	2407.0	[31, 5473]
Number of Features	18.2 ± 41.9	8.0	[1, 299]

The names and IDs of the utilized OpenML datasets are the following: **Binary**: sick (38), hepatitis (55), oil_spill (311), scene (312), yeast_ml8 (316), SPECT (336), SyskillWebertSheep (376), various analcatdata datasets (450, 463, 465, 479, 728, 747, 757, 760, 764, 765, 767, 852, 865, 867, 875), backache (463), balloon (914), socmob (934), water-treatment (940), various arsenic datasets (947, 949, 950, 951), spectrometer (954), braziltourism (957), segment (958), mfeatmorphological (962); **Multiclass**: page-blocks (30.0–30.4), abalone (183.1–183.10, 183.12–183.15, 183.21–183.27). IDs with "." refer to different binary decompositions of multiclass datasets, all of which exhibit different sample sizes and imbalance ratios.

Preprocessing involved mode imputation and ordinal encoding for categorical features, while missing numerical values were imputed using $k-$nearest neighbors (with $k = 5$).

Compared Baselines. Our method was compared against four resampling techniques: the widely used SMOTE and ADASYN, along with SMOTE-IPF and ProWSyn, two of the best-performing methods identified in a comprehensive study of 85 SMOTE variants [16]. Since resampling methods cannot encode user preferences in data generation, we also compared our approach with cost-sensitive classifiers, including cost-sensitive Random Forest, SVM, Extra Trees, and AdaBoost with cost-sensitive Decision Trees as the base classifier.

Evaluation Metrics. We assess user preferences using the $F\beta - score$ via 5-fold cross-validation, ensuring resampling is only applied to the training set. To compare performance across datasets, methods are ranked by their average $F\beta$-score per dataset (1 to 9, lower is better), and the mean rank is reported. We also compute the median percentage difference in $F\beta$-score between competing

approaches and our method to quantify relative performance increase/decrease (positive values indicate an increase of performance versus competing method and vice-versa). Various β values are tested to evaluate the method's ability to achieve specific trade-offs between precision and recall. Note that, to incorporate the same objective in cost-sensitive classifiers as in GBRF, for each tested β, the missclassification penalty of the minority class is set to β^2, whereas the misslcassification penalty of the majority class is always kept at 1, as stipulated by Equation 11.

Implementation Hyperparameters and Settings. $\mathbf{GBRF}_{hybrid}$ refers to the use of g_1 and f_{SMOTE} for sample generation, guided by p_1, while g_2 is employed to form sample groups from which instances are removed based on p_2. In this configuration, outlier removal is disabled. In contrast, $\mathbf{GBRF}_{over}$ denotes the use of both p_1 and p_2 for sample generation, with outlier removal enabled. When either of the framework variants is applied with a classifier, the same type of classifier is used within the fitness function.

4.2 Performance Comparisons (RQ1)

Tables 2 and 3 present the average rankings for the $\mathbf{GBRF}_{over}$ and $\mathbf{GBRF}_{hybrid}$ variants of the proposed framework, respectively, applied with the Naïve Bayes and Random Forest classifiers.

In Table 2, $\mathbf{GBRF}_{over}$ exhibits higher performance across all β values, except for $\beta = 10$, for both classifiers. In addition, the method's average performance is notably superior to all other approaches. This demonstrates its ability to maximize either recall or precision based on predefined user preferences by selectively resampling specific regions of the feature space. Notably, when paired with Gaussian Naïve Bayes—a typically less effective classifier due to its assumption of feature independence and normal distributed data—$\mathbf{GBRF}_{over}$ outperforms cost-sensitive ensemble methods, which are often the strongest performers for tabular data. This result underscores both the robustness of the proposed method and the limitations of cost-sensitive approaches. The latter impose a fixed β^2 misclassification penalty on the minority class, which effectively mimics replicating (i.e. oversampling) each sample β^2 times. In practice, this rigid penalization often results in highly nonlinear and overfitted decision boundaries that fail to generalize well (or at all), similar to the shortcomings of random oversampling [4]. This effect was particularly evident in more challenging datasets, where cost-sensitive methods showed substantial declines in performance compared to resampling-based techniques, thus justifying the large observed differences in median performance. In contrast, resampling-based methods can more effectively expand the minority class distribution through interpolation-based sample generation, which is less susceptible to overfitting when tailored to the dataset characteristics and aligned with user preferences. Furthermore, it is evident that conventional resampling methods lack the ability to incorporate user preferences into the data distribution, as evidenced by their reduced performance at extreme β values.

Table 2. Average rank obtained over the $F\beta$-scores across all datasets for different β values with $\mathbf{GRFB_{over}}$, excluding ties. Tests are conducted with Naives Bayes (top) and Random Forest (bottom). Average median % performance difference of GBRF relative to the competing methods is also presented. Best result per β is **bolded**.

Approach	Methods | β values	0.1	0.2	0.5	1	2	5	10	Average
Resampling + Gaussian NB	**$\mathbf{GRFB_{over}}$ (Ours)**	**3.91**	**3.96**	**4.08**	**3.77**	**3.88**	**4.12**	4.00	**3.96 (0.00%)**
	SMOTE-IPF	5.11	4.96	4.71	4.59	4.36	4.35	4.45	4.65 (1.55%)
	ProWSyn	4.90	4.75	4.20	4.40	4.38	4.49	4.75	4.55 (1.56%)
	SMOTE	4.79	4.84	4.76	4.62	4.70	4.49	4.80	4.71 (1.19%)
	ADASYN	5.00	4.97	5.19	4.92	4.61	4.29	4.58	4.79 (1.17%)
Cost-sensitive Classifiers	CS Random Forest	4.55	4.89	5.44	5.50	6.11	6.23	6.33	5.58 (25.36%)
	CS SVM	7.67	7.55	7.14	6.44	5.74	4.63	**3.68**	6.12 (58.64%)
	CS ExtraTrees	4.85	4.92	5.32	5.79	6.25	6.76	6.77	5.81 (28.46%)
	CS AdaBoost	4.22	4.15	4.15	4.96	4.98	5.64	5.64	4.82 (18.96%)
Resampling + Random Forest	**$\mathbf{GRFB_{over}}$ (Ours)**	**3.37**	**3.30**	**3.85**	**3.97**	**3.53**	**3.57**	3.80	**3.63 (0.00%)**
	SMOTE-IPF	4.79	4.81	4.09	4.20	4.79	4.99	5.08	4.68 (2.63%)
	ProWSyn	3.92	4.16	4.51	4.44	4.33	4.12	4.51	4.28 (0.70%)
	SMOTE	4.41	4.23	4.09	4.36	4.36	4.92	5.15	4.50 (2.24%)
	ADASYN	4.74	4.51	4.40	4.05	4.47	4.74	4.73	4.52 (2.03%)
Cost-sensitive Classifiers	CS Random Forest	5.15	5.41	5.65	5.68	6.34	6.21	6.22	5.81 (19.18%)
	CS SVM	8.01	7.99	7.67	6.72	5.38	4.19	**3.48**	6.21 (53.75%)
	CS ExtraTrees	5.35	5.60	6.12	6.15	6.50	6.70	6.55	6.14 (17.82%)
	CS AdaBoost	5.27	4.99	4.61	5.42	5.29	5.54	5.48	5.23 (7.20%)

CS: Cost sensitive; NB: Naive Bayes;

Regarding $\mathbf{GBRF}_{hybrid}$, it demonstrates even greater effectiveness than $\mathbf{GBRF}_{over}$, achieving up to 8% improvement in $F\beta$-score compared to all other resampling approaches. By introducing the ability to remove majority-class samples, the framework can refine the decision boundary by modifying both class distributions, thereby providing additional flexibility in achieving the desired trade-off between precision and recall. Notably, this improvement is particularly observed for $\beta = 1$ with the Random Forrest, highlighting GBRF's ability to outperform traditional resampling methods even at their designed precision-recall operating region. This is made possible by the method's ability to adapt its resampling strategy to the dataset's specific characteristics. Additionally, the framework performs optimally with multiple classifiers, maintaining a model-agnostic property similar to conventional resampling techniques (with results from additional classifiers available in Appendix A). To further validate the proposed framework's ability to encode user preferences in the data distribution, we present a graphical analysis of the differences in resampled datasets and their corresponding probability mass functions using $\mathbf{GBRF}_{hybrid}$ with varying β values. The contrast between $\beta = 0.1$ and $\beta = 10$ is immediately evident: at $\beta = 0.1$, sample generation occurs primarily in regions densely populated by minority class samples, thereby reducing false positives. In contrast, $\beta = 10$ results in a substantial removal of majority class samples, as indicated by the high values of p_2 in several sample groups, to ensure correct classification of all mi-

nority samples and minimize false negatives. This showcases how GBRF shapes sampling distributions to achieve the desired model behavior. Moreover, GBRF enhances transparency by providing explicit information on how the resampling was conducted, which is typically lacking in existing resampling techniques.

Table 3. Average rank obtained over the $F\beta$-scores across all datasets for different β values with $\mathbf{GRFB_{hybrid}}$, excluding ties. Tests are conducted with Naives Bayes (top) and Random Forest (bottom). Average median % performance of GBRF difference relative to the competing method is also presented. Best result per β is **bolded**.

Approach	Methods | β values	0.1	0.2	0.5	1	2	5	10	Average
Resampling + Gaussian NB	**$\mathbf{GRFB_{hybrid}}$ (Ours)**	**4.12**	**4.32**	**3.77**	**3.76**	**3.61**	**3.62**	3.77	**3.85 (0.00%)**
	SMOTE-IPF	5.08	5.08	5.01	4.62	4.57	4.81	4.64	4.83 (2.47%)
	ProWSyn	4.65	4.72	4.27	4.32	4.30	4.47	4.53	4.47 (1.84%)
	SMOTE	4.99	4.85	4.77	4.73	4.62	4.58	4.59	4.73 (1.96%)
	ADASYN	4.99	5.12	5.28	4.90	4.66	4.51	4.74	4.89 (2.60%)
Cost-sensitive Classifiers	CS Random Forest	4.45	4.51	5.13	5.59	6.18	6.29	6.58	5.53 (27.73%)
	CS SVM	7.65	7.59	7.17	6.50	5.74	4.65	**3.74**	6.15 (59.57%)
	CS ExtraTrees	4.73	4.74	5.50	5.83	6.22	6.70	6.74	5.78 (28.44%)
	CS AdaBoost	4.33	4.09	4.10	4.76	5.11	5.38	5.67	4.77 (17.27%)
Resampling + Random Forest	**$\mathbf{GRFB_{hybrid}}$ (Ours)**	**3.49**	**3.30**	**3.69**	**2.98**	**2.96**	**2.90**	**3.16**	**3.21 (0.00%)**
	SMOTE-IPF	4.64	4.81	4.30	4.23	4.49	5.01	5.14	4.66 (7.57%)
	ProWSyn	4.43	4.16	4.29	4.80	4.57	4.71	4.76	4.53 (4.94%)
	SMOTE	4.47	4.23	4.02	4.48	4.73	5.18	4.94	4.58 (7.96%)
	ADASYN	4.23	4.51	4.54	4.20	4.42	5.16	5.24	4.61 (7.69%)
Cost-sensitive Classifiers	CS Random Forest	5.44	5.41	5.87	5.87	6.37	5.92	6.16	5.86 (29.07%)
	CS SVM	7.85	7.99	7.60	6.82	5.61	4.20	3.53	6.23 (59.23%)
	CS ExtraTrees	5.10	5.60	6.06	6.40	6.55	6.63	6.48	6.12 (24.74%)
	CS AdaBoost	5.35	4.99	4.63	5.22	5.30	5.29	5.60	5.20 (15.62%)

CS: Cost sensitive; NB: Naive Bayes;

4.3 Probability Mass Function Analysis (RQ2)

Figure 2 presents the average probability mass functions (PMFs) obtained for datasets with five sample groups for both g_1 and g_2, thus simplifying the analysis. Initially, it is evident that for g_1, β values greater than one tend to acquire significantly higher sampling probabilities in regions of low minority density, aggressively promoting the expansion of the majority class by generating samples in such areas. Conversely, β values lower than one preferentially generate samples in high minority density regions, reinforcing existing clusters in a more controlled manner. Moreover, in the case of g_2, since sample generation is based on a broader neighborhood, thus facilitating the expansion of the majority class, higher probabilities are assigned to these sample groups for larger β values, and vice-versa. These findings align with the theoretical discussion in Section 3.2, where sampling in regions of lower minority class likelihood is shown to shift the decision boundary towards the majority class, and vice-versa.

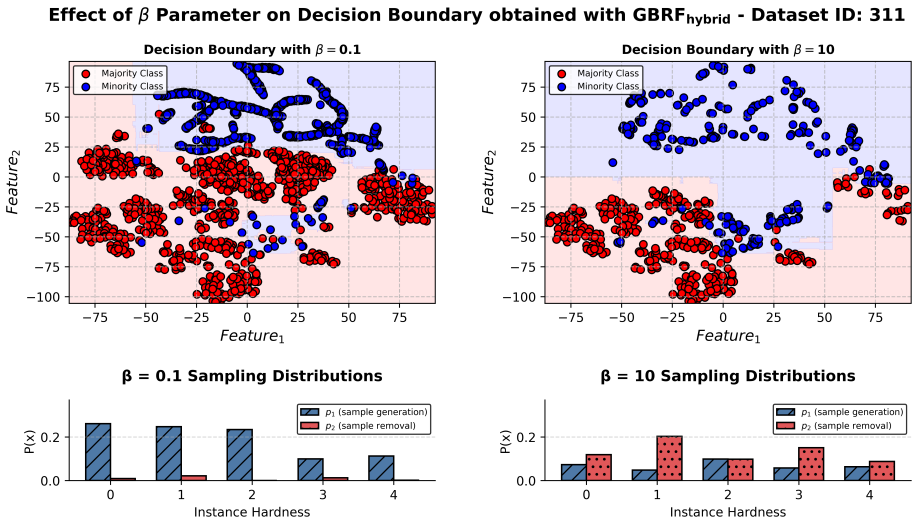

Fig. 1. Synthetic dataset obtained after applying $\mathbf{GBRF}_{hybrid}$ on Dataset ID 311 (randomly selected) from the OpenML repository with $\beta = 0.1$ (left) and $\beta = 10$ (right). The optimal PMF for both scenarios is shown below, where the x-axis represents the number of majority samples within the neighborhood of minority samples for g_1 and the proximity to the decision boundary for g_2, thereby reflecting instance hardness.

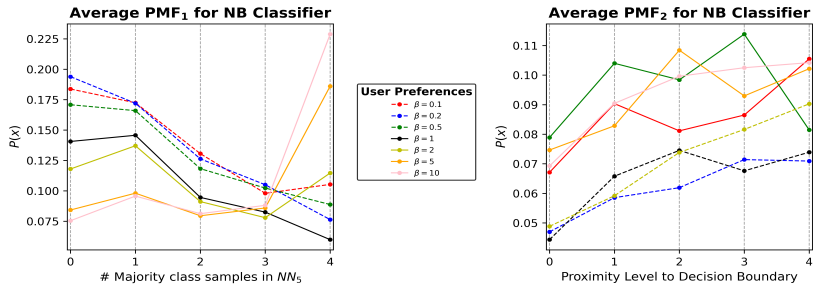

Fig. 2. Optimal average PMFs obtained for p_1 (left) and p_2 (right) with $\mathbf{GBRF}_{overs}$ on a set of 88 train splits (generated via 5-fold cross validation of the 60 OpenML datasets), where the same amount of sample groups were generated per PMF. These averages are computed for each β value individually, with $\beta \geq 1$ represented with a solid line, whereas $\beta < 1$ are represented with a dotted line.

4.4 Impact of Architectural Choices on Computational Efficiency (RQ3)

To evaluate the effect of expanding the search space on the method's ability to attain high-quality solutions, we compared the average ranks of $\mathbf{GRFB}_{\mathbf{over}}$ under two conditions: one where each $g_1(G_k)$ and $g_2(G_k)$ is restricted to 20

equally spaced values and another where this range is expanded by a factor of 100. Note that, even a 100× expansion remains negligible relative to the search spaces routinely encountered in instance-wise genetic resampling approaches.

Table 4. Average ranks for the $\mathbf{GRFB_{over}}$ methodology with Naives Bayes Classifier, when considering a 100-fold increase in possible values of p_1 and p_2 per sample group, compared to the baseline search space. Median runtime across all datasets and β values is also presented. The experiments were conducted on a machine with an AMD Ryzen 7 5800H and GeForce RTX 3070 Mobile/Max-Q running Ubuntu 22.04.4 LTS without any background processes running simultaneously. Best results are **bolded**.

Methodology | β values	**0.1**	**0.2**	**0.5**	**1**	**2**	**5**	**10**	**Average**
100x search space $\mathbf{GRFB_{over}}$ (Median Runtime : 9.696s)	4.74	4.99	4.84	5.20	5.03	4.51	4.72	4.86
Baseline $\mathbf{GRFB_{over}}$ (Median Runtime : **5.294s**)	**3.91**	**3.96**	**4.08**	**3.77**	**3.88**	**4.12**	**4.00**	**3.96**

Analysis of Table 4 reveals that a larger parameter grid incurs a significant drop in performance alongside increased runtime, thereby highlighting the advantage of the proposed architectural design. Our fixed-size, group-wise mechanism offsets the combinatorial explosion of search space in standard genetic-resampling methods, producing superior solutions. This framework also expedites convergence via more frequent activation of the early stopping criteria, as noted by the reduced runtime under identical genetic hyperparametrizations (except search space). A median runtime of approximately 5s was recorded, which is notably efficient for a genetic-based algorithm. Still, the predominant contributor to total runtime is the model training required for fitness estimation. These bottlenecks may be alleviated by optimizing key parameters dictating runtime (e.g., population size, generation count), exploiting in-context learning for tabular data to bypass the need for model training, or initializing the evolutionary search with a low time complexity classifier, as models trained with equivalent objectives (β parameters) generally converge to analogous data distributions.

5 Conclusion

The widespread adoption of resampling methods for addressing class imbalance originates from their compatibility with diverse learning algorithms. However, their inability to explicitly encode user preferences in the resulting models has driven the development of this work. We introduce GBRF, a novel framework that integrates user-defined preferences into the data distribution, shaping model behavior through a tunable parameter, β, which governs the precision-recall trade-off. GBRF leverages genetic algorithms to optimize two probability mass functions that control the generation and removal of samples, effectively functioning as an oversampling, undersampling, or hybrid resampling method. By

employing an evolutionary strategy, we ensure that resampling is conducted to account for both user-preferences and dataset characteristics. Theoretical analysis and empirical validation across 60 benchmark datasets confirm that GBRF effectively encodes user preferences, consistently surpassing eight state-of-the-art resampling and cost-sensitive approaches, particularly in hybrid settings. Additionally, the learned probability mass functions provide accurate and interpretable insights into the resampling process, offering a structured approach to understanding the inter-dependencies between data characteristics, user preferences, and resampling strategies. Future research will focus on extending this methodology to multiclass scenarios and improving scalability.

Acknowledgments. This work was supported by FCT - Fundação para a Ciência e Tecnologia, I.P. by project reference PRT/BD/154859/2023 and DOI identifier https://doi.org/10.54499/PRT/BD/154859/2023. Additionally, this work was also funded by FCT under unit 00127-IEETA.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Alam, T., Qamar, S., Dixit, A., Benaida, M.: Genetic algorithm: Reviews, implementations, and applications (2020), https://arxiv.org/abs/2007.12673
2. Alimohammadi, D., Bolin, M.: Mathematics for classical information retrieval: Roots and applications (12 2010)
3. Barua, S., Islam, M.M., Murase, K.: Prowsyn: Proximity weighted synthetic oversampling technique for imbalanced data set learning. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) Advances in Knowledge Discovery and Data Mining. pp. 317–328. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
4. Branco, P., Torgo, L., Ribeiro, R.P.: A survey of predictive modeling on imbalanced domains. ACM Comput. Surv. **49**(2) (Aug 2016). https://doi.org/10.1145/2907070, https://doi.org/10.1145/2907070
5. Chahar, V., Katoch, S., Chauhan, S.: A review on genetic algorithm: Past, present, and future. Multimedia Tools and Applications **80** (02 2021). https://doi.org/10.1007/s11042-020-10139-6
6. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research **16**, 321–357 (Jun 2002). https://doi.org/10.1613/jair.953, http://dx.doi.org/10.1613/jair.953
7. Douzas, G., Bação, F., Last, F.: Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. Information Sciences **465** (06 2018). https://doi.org/10.1016/j.ins.2018.06.056
8. Drown, D., Khoshgoftaar, T., Seliya, N.: Evolutionary sampling and software quality modeling of high-assurance systems. IEEE Transactions on Systems, Man, and Cybernetics, Part A **39**, 1097–1107 (09 2009). https://doi.org/10.1109/TSMCA.2009.2020804
9. Fernández, A., Garcia, S., Herrera, F., Chawla, N.: Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. Journal

of Artificial Intelligence Research **61**, 863–905 (04 2018). https://doi.org/10.1613/jair.1.11192
10. Fernández, A., García, S., Galar, M., Prati, R., Krawczyk, B., Herrera, F.: Learning from Imbalanced Data Sets (01 2018). https://doi.org/10.1007/978-3-319-98074-4
11. Ha, J., Lee, J.S.: A new under-sampling method using genetic algorithm for imbalanced data classification. In: Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication. IMCOM '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2857546.2857643, https://doi.org/10.1145/2857546.2857643
12. 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://doi.org/10.1109/IJCNN.2008.4633969
13. Jain, A., Ratnoo, S., Kumar, D.: Addressing class imbalance problem in medical diagnosis: A genetic algorithm approach. pp. 1–8 (08 2017). https://doi.org/10.1109/ICOMICON.2017.8279150
14. Jiang, K., Lu, J., Xia, K.: A novel algorithm for imbalance data classification based on genetic algorithm improved smote. Arabian Journal for Science and Engineering **41** (05 2016). https://doi.org/10.1007/s13369-016-2179-2
15. Kim, H.J., Jo, N.O., Shin, K.S.: Optimization of cluster-based evolutionary undersampling for the artificial neural networks in corporate bankruptcy prediction. Expert Syst. Appl. **59**(C), 226–234 (Oct 2016). https://doi.org/10.1016/j.eswa.2016.04.027, https://doi.org/10.1016/j.eswa.2016.04.027
16. Kovács, G.: An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. Applied Soft Computing (07 2019). https://doi.org/10.1016/j.asoc.2019.105662
17. Kulkarni, A., Chong, D., Batarseh, F.A.: 5 - foundations of data imbalance and solutions for a data democracy. In: Batarseh, F.A., Yang, R. (eds.) Data Democracy, pp. 83–106. Academic Press (2020). https://doi.org/https://doi.org/10.1016/B978-0-12-818366-3.00005-8, https://www.sciencedirect.com/science/article/pii/B9780128183663000058
18. Lu, Y., ming Cheung, Y., Tang, Y.Y.: Bayes imbalance impact index: A measure of class imbalanced dataset for classification problem (2019), https://arxiv.org/abs/1901.10173
19. Onan, A., García-Díaz, V.: Consensus clustering-based undersampling approach to imbalanced learning. Sci. Program. **2019** (Jan 2019). https://doi.org/10.1155/2019/5901087, https://doi.org/10.1155/2019/5901087
20. Santos, M.S., Abreu, P.H., Japkowicz, N., Fernández, A., Santos, J.: A unifying view of class overlap and imbalance: Key concepts, multi-view panorama, and open avenues for research. Information Fusion **89**, 228–253 (2023). https://doi.org/https://doi.org/10.1016/j.inffus.2022.08.017, https://www.sciencedirect.com/science/article/pii/S1566253522001099
21. Wernernbsp;denbsp;Vargas, V., Schneidernbsp;Aranda, J.A., dos Santosnbsp;Costa, R., da Silvanbsp;Pereira, P.R., Victórianbsp;Barbosa, J.L.: Imbalanced data preprocessing techniques for machine learning: a systematic mapping study. Knowl. Inf. Syst. **65**(1), 31–57 (Nov 2022). https://doi.org/10.1007/s10115-022-01772-8, https://doi.org/10.1007/s10115-022-01772-8
22. Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1), 67–82 (1997). https://doi.org/10.1109/4235.585893