Highlights

Geometric SMOTE for Imbalanced Datasets with Nominal and Continuous Features

Joao Fonseca, Fernando Bacao

- We propose Geometric-SMOTENC, a new oversampling algorithm for datasets with nominal and continuous features;
- We test the oversampler's performance over 20 datasets and compare it to 4 other relevant oversampling methods;
- Geometric-SMOTENC consistently outperforms the remaining methods and significantly improves classification performance;

Geometric SMOTE for Imbalanced Datasets with Nominal and Continuous Features

Joao Fonseca[a], Fernando Bacao[a]

[a]*NOVA Information Management School, Universidade Nova de Lisboa, Campus de Campolide, Lisboa, 1070–312, Lisboa, Portugal*

Abstract

There are different approaches to address imbalanced learning. Artificial data generation, however, is a more general approach when opposed to algorithmic modifications or cost-sensitive solutions. Since the proposal of the Synthetic Minority Oversampling TEchniqu, howevere (SMOTE), various SMOTE variants and neural network-based oversampling methods have been developed. However, the options to oversample datasets with nominal and continuous features are limited. We propose Geometric SMOTE for Nominal and Continuous features (G-SMOTENC), based on a combination of G-SMOTE and SMOTENC. Our method uses SMOTENC's encoding and generation mechanism for nominal features while using G-SMOTE's data selection mechanism to determine the center observation and k-nearest neighbors and generation mechanism for continuous features. G-SMOTENC's performance is compared against SMOTENC's along with two other baseline methods, a State-of-the-art oversampling method and no oversampling. The experiment was performed over 20 datasets with varying imbalance ratios, number of metric and non-metric features and target classes. We found a significant improvement in the quality of the generated data when using G-SMOTENC as the oversampling method. An open-source implementation of G-SMOTENC is made available in the Python programming language.

Keywords: Imbalanced Learning, Oversampling, SMOTE, Data Generation, Nominal Data

1. Introduction

Various Machine Learning (ML) tasks deal with highly imbalanced datasets, such as fraud transactions detection, fault detection and medical diagnosis [36]. In these situations, predicting false positives is often a more acceptable error, since the class of interest is usually the minority class [37]. However, using standard ML classifiers on imbalanced datasets induce a bias in favor of the classes with the highest frequency, while limiting the predictive power on lower frequency classes [27, 7]. This effect is known in the ML community as the Imbalanced Learning problem.

Imbalanced learning involves a dataset with two or more target classes with varying class frequencies. The minority class is defined as the class with the least amount of observations

and the majority class is the one with the highest amount of observations [23]. There are three main approaches to address imbalanced learning [13]:

1. Cost-sensitive solutions attribute a higher misclassification cost to the minority class observations to minimize higher cost errors;
2. Algorithmic level solutions modify ML classifiers to improve the learning of the minority class;
3. Resampling solutions generate synthetic minority class observations and/or remove majority class observations to balance the training dataset;

Since it is an external approach to imbalanced learning, the latter method becomes particularly useful. It dismisses the required domain knowledge to build a cost matrix and the technical complexity/knowledge of applying an imbalanced learning-specific classifier. Resampling can be done via undersampling, oversampling, or hybrid approaches [35]. In this paper, we will focus on oversampling approaches.

The presence of nominal features in imbalanced learning tasks limits the options available to deal with class imbalance. Even though it is possible to use encoding methods such as one-hot or ordinal encoding to convert nominal features into numerical, applying a distance metric on mixed-type datasets is questionable since the nominal feature values are unordered [28]. In this case, one possible approach is to use models that can handle different scales (*e.g.*, Decision Tree). However, this assumption may be limiting since there are few ML algorithms where this condition is verified. Another possible approach is transforming the variables to meet scale assumptions [28]. This method was explored in the algorithm Synthetic Minority Oversampling Technique for Nominal and Continuous features (SMOTENC) [6] (explained in Section 2).

In the presence of datasets with mixed data types, using most of the well-known resampling algorithms becomes unfeasible. This happens because these methods consider exclusively continuous data; they were not adapted to also use nominal features. Specifically, since the proposal of SMOTE, various other SMOTE-variants have been developed to address some of its limitations. Although, there was not a significant development in research to oversample datasets with both nominal and continuous features.

In this paper, we propose Geometric SMOTE for Nominal and Continuous features (G-SMOTENC). It generates the continuous feature values of a synthetic observation within a truncated hyper-spheroid with its nominal feature values using the most common value of its nearest neighbors. In addition, G-SMOTENC uses G-SMOTE's data selection strategy and SMOTENC's approach to find the center observation's nearest neighbors. G-SMOTENC is a generalization of both SMOTENC and G-SMOTE [9]. With the correct hyperparameters, our G-SMOTENC implementation can mimic the behavior of SMOTE, SMOTENC, or G-SMOTE. It is available in the open-source Python library "ML-Research" and is fully compatible with the Scikit-Learn ecosystem. These contributions can be summarized as follows:

1. We propose G-SMOTENC, an oversampling algorithm based on G-SMOTE for datasets with nominal and continuous features;

2. We test the proposed oversampler using 20 datasets and compare its performance to SMOTENC, Random Oversampling, Random Undersampling and a State-of-the-art oversampler;
3. We provide an implementation of G-SMOTENC in the Python programming language;

The rest of this paper is structured as follows: Section 2 describes the related work and its limitations, Section 3 describes the proposed method (G-SMOTENC), Section 4 lays out the methodology used to test G-SMOTENC, Section 5 shows and discusses the results obtained in the experiment and Section 6 presents the conclusions drawn from this study.

2. Related Work

A classification problem contains n classes, having C_{maj} as the set of majority class observations (*i.e.*, observations belonging to the most common target class) and C_{min} as the set of minority class observations (*i.e.*, observations belonging to the least common target class). Typically, an oversampling algorithm will generate synthetic data in order to ensure $|C'_{min}| = |C_{maj}| = |C_i|, i \in \{1, \ldots, n\}$.

Since the proposal of SMOTE, other methods modified or extended SMOTE to improve the quality of the data generated. The process of generating synthetic data using SMOTE-based algorithms can be divided into two distinct phases [12]:

1. Data selection. A synthetic observation, x^{gen}, is generated based on two existing observations. A SMOTE-based algorithm employs a given heuristic to select a non-majority class observation as the center observation, x^c, and one of its nearest neighbors, x^{nn}, selected randomly. For the case of SMOTE, x^c is randomly selected from each non-majority class.
2. Data generation. Once x^c and x^{nn} have been selected, x^{gen} is generated based on a transformation between the two selected observations. In the case of SMOTE, this transformation is a linear interpolation between the two observations: $x^{gen} = \alpha x^c + (1 - \alpha)x^{nn}, \alpha \sim \mathcal{U}(0, 1)$.

Modifications to the SMOTE algorithm can be distinguished according to the phase where they were applied. This distinction is especially relevant for the case of oversampling on datasets with mixed data types since it raises the challenge of calculating meaningful distances and k-nearest neighbors among observations. For example, State-of-the-art oversampling methods, such as Borderline-SMOTE [18], ADASYN [19], K-means SMOTE [11] and LR-SMOTE [26] modify the data selection mechanism and show promising results in imbalanced learning [14]. However, these algorithms select x^c using procedures that include calculating each observation's k-nearest neighbors or clustering methods, which are not prepared to handle categorical data.

Modifications to SMOTE's generation mechanism are uncommon. A few oversampling methods, such as Safe-level SMOTE [5] and Geometric-SMOTE [9] proposed this type of modification and have shown promising results [10]. However, these methods are also

unable to handle datasets with categorical data. Other methods attempt to replace the SMOTE data generation mechanism altogether using different Generative Adversarial Networks (GAN) architectures [33, 24, 22]. Network-based architectures, however, are computationally expensive to train and sensitive to the training initialization. It is also difficult to ensure a balanced training of the two networks involved and tuning their hyperparameters is often challenging or unfeasible [17].

As discussed in Section 1, research on resampling methods with mixed data types is scarce. The original paper proposing SMOTE also proposed SMOTE for Nominal and Continuous (SMOTENC), an adaptation of SMOTE to handle datasets with nominal and continuous features [6]. To determine the k-nearest neighbors of x^c, the Euclidean distance is modified to include the median of the standard deviations of the continuous features for every categorical feature with different values. Once x^c and x^{nn} are defined, the continuous feature values in x^{gen} are generated using the SMOTE generation mechanism. The categorical features are given the most common values occurring in the k-nearest neighbors.

Recently, a new SMOTE-based oversampling method for datasets with mixed data types, SMOTE-ENC [29], was proposed. This method modifies the encoding mechanism for categorical features used in the SMOTENC algorithm to account for categorical features' change of association with minority classes. The Multivariate Normal Distribution-based Oversampling for Numerical and Categorical features (MNDO-NC) [2] uses the original MNDO method [1] along with the SMOTENC encoding mechanism to find the values of the categorical features for the synthetic observation. However, the results reported in the paper showed that MNDO-NC was consistently outperformed by SMOTENC, which led us to discard this approach from further consideration.

Alternatively to SMOTE-based methods, it is possible to use non-informed over and undersampling methods for datasets with nominal and continuous features, specifically Random Oversampling (ROS) and Random Undersampling (RUS). These methods consist of randomly duplicating minority class observations (in the case of ROS), which can lead to overfitting [30, 4], or randomly removing majority class observations (in the case of RUS), which may lead to underfitting [3].

3. Proposed Method

We propose G-SMOTENC to handle both nominal and continuous features. This extension of the original G-SMOTE oversampler its selection and generation mechanisms. Due to the novelty of the work, these modifications are based on the SMOTENC mechanism. However, this method can be extended with further modifications to the categorical data encoding and selection mechanisms in future work.

Similar to G-SMOTE being an extension of SMOTE, G-SMOTENC is also an extension of SMOTENC since any method or ML pipeline using the SMOTENC generation mechanism can replace it with G-SMOTENC without any further modifications. The proposed method is described in pseudo-code in Algorithm 1. The functions $SelectionMechanism$ and $GenerationMechanism$ are described in Algorithms 2 and 3, respectively.

Algorithm 1: G-SMOTENC.

Given: Dataset with binary target classes C_{min} and C_{maj}
Input: $C_{maj}, C_{min}, \alpha_{sel}, \alpha_{trunc}, \alpha_{def}$
Output: C^{gen}
begin
  $N \leftarrow |C_{maj}| - |C_{min}|$
  $C^{gen} \leftarrow \emptyset$
  while $|C^{gen}| < N$ **do**
    $x^c, x^{nn}, X^{nn} \leftarrow SelectionMechanism(C_{maj}, C_{min}, \alpha_{sel})$
    $x^{gen} \leftarrow GenerationMechanism(x^c, x^{nn}, X^{nn}, \alpha_{trunc}, \alpha_{def})$
    $C^{gen} \leftarrow C^{gen} \cup \{x^{gen}\}$

G-SMOTENC's implementation involves additional considerations regarding the management of the nominal features. During the selection mechanism (identified as the function *SelectionMechanism*), the categorical features are encoded using the one-hot encoding technique, while the non-zero constant assumes the value of the median of the standard deviations of the continuous features in C_{min}, divided by two. This encoding mechanism varies from the one in SMOTENC in order to attribute less weight to the categorical features relative to the continuous features.

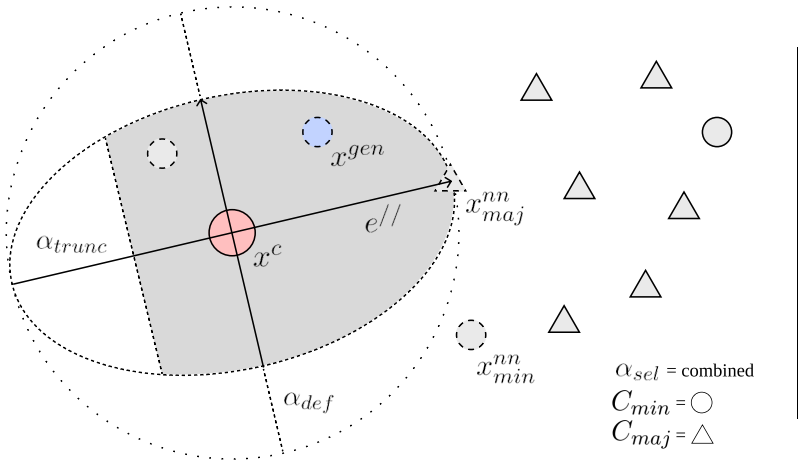
The selection strategy, α_{sel}, as well as C_{min} and C_{maj} are used to determine a central observation, x^c, its nearest neighbors, X^{nn}, and one of its nearest neighbors, $x^{nn} \in X^{nn}$. X^{nn} is calculated using the euclidean distance and both the continuous and encoded nominal features. The outcome of this step is dependent on the choice of α_{sel}:

1. If $\alpha_{sel} = minority$, X^{nn} will consist of x^c's k-nearest neighbors within C_{min};
2. If $\alpha_{sel} = majority$, X^{nn} will consist of x^c's nearest neighbor within C_{maj};
3. If $\alpha_{sel} = combined$, X^{nn} will consist of the union between x^c's k-nearest neighbors within C_{min} and x^c's nearest neighbor within C_{maj}, *i.e.*, $C_{min,k} \cup C_{maj,1}$. In this case, x^{nn} is selected using the majority class observation within X^{nn} as well as another randomly selected nearest neighbor, such that $x^{nn} = argmin(||x^{nn}_{min} - x^c||, ||x^{nn}_{maj} - x^c||)$;

Unlike in the original G-SMOTE mechanism, X^{nn} is used in the generation mechanism to determine the categorical feature values of x^{gen} based on the mode of these features within X^{nn}. The selection strategy uses two additional hyperparameters to generate the continuous features in x^{gen}: the truncation factor, α_{trunc}, and the deformation factor, α_{def}. They are generated by forming a hyper-sphere with center x^c and is modified according to the parameters:

1. α_{trunc} partitions the hyper-sphere
2. α_{def}

Figure 1 depicts the effect of those hyperparameters in the data selection and generation phases. For an in-depth explanation of these hyperparameters, the reader is referred to [9].



X^{nn}		f_1^{nom}	f_2^{nom}	f_3^{nom}
x^{nn}_{maj}	△	A	C	B
x^{nn}_{min}	○	B	B	B
n/a	○	B	C	A
x^{gen}_{nom}	●	B	C	B

Figure 1: A visual depiction of G-SMOTENC. In this example, α_{trunc} is approximately 0.5 and α_{def} is approximately 0.4.

3.1. Selection Mechanism

The data selection mechanism is preceded by the numerical encoding of the categorical features. It mixes the selection mechanisms of SMOTENC and G-SMOTENC, as shown in Algorithm 2. The selection mechanism uses the minority, majority, and combined mechanisms. The nominal features in the minority and majority class observations, C_{maj} and C_{min} are first encoded using a one-hot encoding approach and replacing the constant 1 with the median of the standard deviations of the continuous features in C_{min} divided by 2. The nearest-neighbors (X^{nn}) of x^c are determined based on α_{sel}, which are passed on to the generation mechanism to determine the nominal features' values of x^{gen} in the generation mechanism. Simultaneously, x^{nn} is randomly selected from X^{nn} and will be used to generate x^{gen}'s continuous features' values.

3.2. Generation Mechanism

G-SMOTENC's generation mechanism is shown in Algorithm 3. It divides the generation of x^{gen} into two parts: (1) generation of continuous feature values and (2) generation of nominal feature values. Next, the nominal features from x^c and x^{nn} are discarded. Afterwards, the continuous features are generated using G-SMOTE's generation mechanism; within a hyper-spheroid formed using α_{trunc} and α_{def}. The nominal feature values are generated by the mode of each feature within the observations in X^{nn}.

4. Methodology

This section describes how the evaluation of G-SMOTENC was performed. We describe the datasets used in the experiment, their source and preprocessing steps executed in

Algorithm 2: G-SMOTENC's selection mechanism.

Input: $C_{maj}, C_{min}, \alpha_{sel}$
Output: x^c, x^{nn}, X^{nn}
Function *CatEncoder(C_{maj}, C_{min})*:
    $S \leftarrow$ Standard deviations of the continuous features in C_{min}
    $\sigma_{med} \leftarrow median(S)$
    forall $i \in \{maj, min\}$ **do**
        forall $f \in C_i^T$ **do**
            if f *is categorical* **then**
                $f' \leftarrow OneHotEncode(f) \times \sigma_{med}/2$
                $C'_i \leftarrow (C_i^T \setminus f)^T$
                $C'_i \leftarrow (C_i'^T \cup f')^T$
    return C'_{maj}, C'_{min}
Function *Surface(α_{sel}, x^c, C_{maj}, C_{min})*:
    if $\alpha_{sel} = minority$ **then**
        $x^{nn} \in C_{min,k}$ `// One of the` k`-nearest neighbors of` x^c `from` C_{min}
        $X^{nn} \leftarrow C_{min,k}$
    if $\alpha_{sel} = majority$ **then**
        $x^{nn} \in C_{maj,1}$ `// Nearest neighbor of` x^c `from` C_{maj}
        $X^{nn} \leftarrow C_{maj,1}$
    if $\alpha_{sel} = combined$ **then**
        $x^{nn}_{min} \in C_{min,k}$
        $x^{nn}_{maj} \in C_{maj,1}$
        $x^{nn} \leftarrow argmin(||x^{nn}_{min} - x^c||, ||x^{nn}_{maj} - x^c||)$
        $X^{nn} \leftarrow C_{min,k} \cup C_{maj,1}$
    return x^{nn}, X^{nn} `//` X^{nn} `is the set of` k`-nearest neighbors`
begin
    $C'_{maj}, C'_{min} \leftarrow CatEncoder(C_{maj}, C_{min})$
    $x^c \in C'_{min}$ `// Randomly select` x^c `from` C'_{min}
    $x^{nn}, X^{nn} \leftarrow Surface(\alpha_{sel}, x^c, C'_{maj}, C'_{min})$
    Reverse encoding of nominal features in x^c, x^{nn} and X^{nn}

Algorithm 3: G-SMOTENC's generation mechanism.

Input: $x^c, x^{nn}, X^{nn}, \alpha_{trunc}, \alpha_{def}$
Output: x^{gen}
Function *Hyperball()*:
  $v_i \sim \mathcal{N}(0,1)$
  $r \sim \mathcal{U}(0,1)$
  $x^{gen} \leftarrow r^{1/p} \frac{(v_1,\ldots,v_p)}{||(v_1,\ldots,v_p)||}$
  return x^{gen}
Function *Vectors(x^c, x^{nn}, x^{gen})*:
  $e^{//} \leftarrow \frac{x^{nn}-x^c}{||x^{nn}-x^c||}$
  $x^{//} \leftarrow (x^{gen} \cdot e^{//})e^{//}$
  $x^{\perp} \leftarrow x^{gen} - x^{//}$
  return $x^{//}, x^{\perp}$
Function *Truncate($x^c, x^{nn}, x^{gen}, x^{//}, \alpha_{trunc}$)*:
  if $|\alpha_{trunc} - x^{//}| > 1$ **then**
    $x^{gen} \leftarrow x^{gen} - 2x^{//}$
  return x^{gen}
Function *Deform($x^{gen}, x^{\perp}, \alpha_{def}$)*:
  return $x^{gen} - \alpha_{def} x^{\perp}$
Function *Translate(x^c, x^{gen}, R)*:
  return $x^c + Rx^{gen}$
Function *GenNominal(X^{nn})*:
  $x^{gen}_{nom} = \emptyset$
  forall $f \in (X^{nn})^T$ **do**
    if *f is categorical* **then**
      $x^{gen}_{nom} \cup \{mode(f)\}$ `// Ties are decided with random selection`
  return x^{gen}_{nom}
begin
  Discard nominal features from x^c and x^{nn}
  $x^{gen} \leftarrow Hyperball()$
  $x^{//}, x^{\perp} \leftarrow Vectors(x^c, x^{nn}, x^{gen})$
  $x^{gen} \leftarrow Truncate(x^c, x^{nn}, x^{gen}, x^{//}, \alpha_{trunc})$
  $x^{gen} \leftarrow Deform(x^{gen}, x^{\perp}, \alpha_{def})$
  $x^{gen} \leftarrow Translate(x^c, x^{gen}, ||x^{nn}_{cont} - x^c||)$
  $x^{gen}_{nom} \leftarrow GenNominal(X^{nn})$
  $x^{gen} \leftarrow x^{gen} \cup x^{gen}_{nom}$

Section 4.1. The resampling and classification methods used to analyze G-SMOTE's performance are listed in Section 4.2. The performance metrics used are defined in Section 4.3. Finally, the experimental procedure is described in Section 4.4.

4.1. Experimental Data

The datasets used in this experiment were extracted from the UC Irvine Machine Learning Repository. All of the datasets are publicly available and cover a range of different domains. The criteria to select the datasets ensured that all datasets are imbalanced and contained non-metric features (*i.e.*, ordinal, nominal or binary). These datasets are used to show how the performance of different classifiers varies across over/undersamplers.

Initially, all datasets were preprocessed manually with minimal manipulations to avoid applying preprocessing methods beyond this paper's scope. This step intends to remove features and/or observations with missing values and to identify the non-metric features. The second stage of our preprocessing was done systematically. The description of the resulting datasets is shown in Table 1.

Table 1: Description of the datasets collected after data preprocessing. The sampling strategy is similar across datasets. Legend: (IR) Imbalance Ratio

Dataset	Metric	Non-Metric	Obs.	Min. Obs.	Maj. Obs.	IR	Classes
Abalone	1	7	4139	15	689	45.93	18
Adult	8	6	5000	1268	3732	2.94	2
Adult (10)	8	6	5000	451	4549	10.09	2
Annealing	4	6	790	34	608	17.88	4
Census	24	7	5000	337	4663	13.84	2
Contraceptive	4	5	1473	333	629	1.89	3
Contraceptive (10)	4	5	1036	62	629	10.15	3
Contraceptive (20)	4	5	990	31	629	20.29	3
Contraceptive (31)	4	5	973	20	629	31.45	3
Contraceptive (41)	4	5	966	15	629	41.93	3
Covertype	2	10	5000	20	2449	122.45	7
Credit Approval	9	6	653	296	357	1.21	2
German Credit	13	7	1000	300	700	2.33	2
German Credit (10)	13	7	770	70	700	10.00	2
German Credit (20)	13	7	735	35	700	20.00	2
German Credit (30)	13	7	723	23	700	30.43	2
German Credit (41)	13	7	717	17	700	41.18	2
Heart Disease	5	5	740	22	357	16.23	5
Heart Disease (21)	5	5	735	17	357	21.00	5

The second part of the data preprocessing pipeline starts with the generation of artificially imbalanced datasets with different Imbalance Ratios ($IR = \frac{|C_{maj}|}{|C_{min}|}$). For each original dataset, we create its more imbalanced versions at intervals of 10, while ensuring that $|C_{min}| \geq 15$. The sampling strategy was determined for class $n \in \{1, \ldots, n, \ldots, m\}$ as a

linear interpolation using $|C_{maj}|$ and $|C'_{min}| = \frac{|C_{maj}|}{IR_{new}}$, as shown in equation 1.

$$|C_i|^{imb} = \min(\frac{|C'_{min}| - |C_{maj}|}{n-1}.|C_i| + |C_{max}|, |C_i|) \quad (1)$$

The new, artificially imbalanced dataset, is formed by sampling observations without replacement from each C_i such that $C'_i \subseteq C_i, |C'_i| = |C_i|^{imb}$. The artificially imbalanced datasets are marked with its imbalance ratio as a suffix in Table 1.

The datasets (both original and artificially imbalanced versions) are then filtered to ensure all datasets have a minimum of 500 observations. The remaining datasets with a number of observations larger than 5000 are randomly sampled to match this number of observations. Afterward, we remove target classes with a frequency lower than 15 observations for each remaining dataset. Finally, the continuous and discrete features are scaled to the range $[0, 1]$ to ensure a common range between all features.

4.2. Machine Learning Algorithms

The choice of classifiers used in the experimental procedure was based on their type (tree-based, nearest neighbors-based, linear model and ensemble-based), popularity and consistency in performance. We used Decision Tree (DT), a K-Nearest Neighbors (KNN) classifier, a Logistic Regression (LR) and a Random Forest (RF).

Given the lack of existing oversamplers that address imbalanced learning problems with mixed data types, the amount of benchmark methods used is also limited. We used three appropriate, well-known methods and one state-of-the-art oversampling method: SMOTENC, RUS, ROS and SMOTE-ENC. Table 2 shows the hyperparameters used for the parameter search described in Section 4.4.

4.3. Performance Metrics

The choice of the performance metric plays a critical role in assessing the effect on classification tasks. The typical performance metrics, *e.g.*, Overall Accuracy (OA), are intuitive to interpret but are often inappropriate to measure a classifier's performance in an imbalanced learning context [34]. For example, to estimate an event that occurs in 1% of the dataset, a constant classifier would obtain an OA of 0.99 and still be unusable. However, this metric is still reported in some of our results to maintain a metric that is easier to interpret.

Recent surveys found the Geometric-mean (G-mean), F1-score (F-score), $Sensitivity = \frac{TP}{FN+TP}$ and $Specificity = \frac{TN}{TN+FP}$ to be appropriate and common performance metrics in imbalanced learning contexts [32, 21, 20]. G-mean and F-score are defined equations 2 and 3, respectively.

$$G\text{-}mean = \sqrt{\overline{Sensitivity} \times \overline{Specificity}} \quad (2)$$

Table 2: Hyperparameter definition for the classifiers and resamplers used in the experiment.

Classifier	Hyperparameter	Values
DT	min. samples split	2
	criterion	gini
	max depth	3, 6
LR	maximum iterations	10000
	multi-class	One-vs-All
	solver	saga
	penalty	None, L1, L2
KNN	# neighbors	3, 5
	weights	uniform
	metric	euclidean
RF	min. samples split	2
	# estimators	50, 100
	Max depth	3, 6
	criterion	gini
Resampler		
SMOTENC	# neighbors	3, 5
SMOTE-ENC	# neighbors	3, 5
G-SMOTENC	# neighbors	3, 5
	deformation factor	0.0, 0.25, 0.5, 0.75, 1.0
	truncation factor	-1.0, -0.5, 0.0, 0.5, 1.0
	selection strategy	"combined", "minority", "majority"
RUS	replacement	False
ROS	(no applicable parameters)	

$$F\text{-}score = 2 \times \frac{\overline{Precision} \times \overline{Recall}}{\overline{Precision} + \overline{Recall}} \tag{3}$$

They are calculated as a function of the number of False/True Positives (FP and TP) and False/True Negatives (FN and TN), having $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$. This led us to adopt, along with OA, both F-score and G-mean as the main performance metrics for this study.

4.4. Experimental Procedure

The experimental procedure was applied similarly to all combinations of resamplers, classifiers and hyperparameter combinations across all datasets. The evaluation of the models' performance was tested using a 5-fold Cross-Validation (CV) approach. The mean performance in the test set is calculated over the five folds and three different runs of the

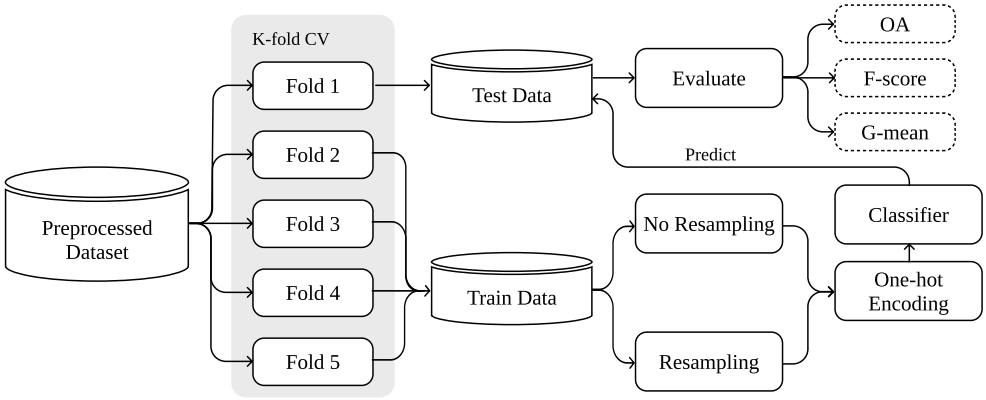


Figure 2: Experimental procedure used in this study.

experimental procedure for each combination of resampling/classifier hyperparameters. For each dataset, we select the results of the hyperparameters that optimize the performance of a resampler/classifier. Figure 2 shows a diagram of the experimental procedure described.

A CV run consists of a stratified partitioning (*i.e.*, each partition contains the same relative frequencies of target labels) of the dataset into five parts. A given resampler/classifier combination with a specific set of hyperparameters is fit and tested five times, using one of the partitions as a test set and the remaining ones as the training set. In the ML pipeline defined for each run, the nominal features are one-hot encoded after oversampling and before passing the data to the classifier. The estimated performance consists of the average classification performance across the five tests and three runs (*i.e.*, a total of 15 tests).

4.5. Software Implementation

The algorithmic implementation of G-SMOTENC was written using the Python programming language and is available in the open-source package ML-Research [15], along with other utilities used to produce the experiment and outputs used in Section 5. In addition, the packages Scikit-Learn [31], Imbalanced-Learn [25] and Research-Learn were also used in the experimental procedure to get the implementations of the classifiers, benchmark over/undersamplers and run the experimental procedure. The original SMOTE-ENC implementation was retrieved from the authors' GitHub repository. The Latex code, Python scripts (including data pulling and preprocessing, experiment setup and analysis of results), as well as the datasets used, are available in this GitHub repository.

5. Results and Discussion

In this section, we present the experimental results. We focus on the classification performance comparison using oversamplers whose generation mechanism is compatible with

datasets containing both continuous and categorical features. The experimental results were analyzed in two stages: (1) analysis of mean ranking and absolute performance and (2) statistical analysis. Section 5.3 discusses the main insights extracted by analyzing the results reported in Sections 5.1 and 5.2.

5.1. Results

Table 3 presents the mean rankings of CV scores across the different combinations of oversamplers, metrics and classifiers. These results were calculated by assigning a ranking score for each oversampler from 1 (best) to 4 (worst) for each dataset, metric and classifier.

Table 3: Mean rankings over the different datasets, folds and runs used in the experiment.

Classifier	Metric	G-SMOTENC	NONE	SMOTENC	ROS	RUS	SMOTE-ENC
DT	OA	1.66 ± 0.13	**1.61 ± 0.27**	3.58 ± 0.20	4.68 ± 0.15	5.42 ± 0.27	4.05 ± 0.23
DT	F-Score	**1.32 ± 0.11**	3.84 ± 0.40	3.13 ± 0.20	4.32 ± 0.19	5.47 ± 0.23	2.92 ± 0.34
DT	G-Mean	**1.68 ± 0.24**	5.84 ± 0.09	2.82 ± 0.21	2.95 ± 0.32	4.26 ± 0.32	3.45 ± 0.30
KNN	OA	2.50 ± 0.17	**1.37 ± 0.28**	4.21 ± 0.25	3.34 ± 0.35	5.68 ± 0.22	3.89 ± 0.15
KNN	F-Score	**1.37 ± 0.16**	3.95 ± 0.35	3.11 ± 0.29	3.47 ± 0.36	5.53 ± 0.23	3.58 ± 0.23
KNN	G-Mean	**1.74 ± 0.17**	5.84 ± 0.12	2.89 ± 0.23	3.76 ± 0.33	3.00 ± 0.45	3.76 ± 0.23
LR	OA	2.74 ± 0.19	**1.37 ± 0.28**	3.08 ± 0.21	4.34 ± 0.30	5.74 ± 0.17	3.74 ± 0.28
LR	F-Score	**2.11 ± 0.24**	4.53 ± 0.35	2.37 ± 0.28	3.47 ± 0.32	5.21 ± 0.27	3.32 ± 0.38
LR	G-Mean	2.13 ± 0.26	6.00 ± 0.00	3.61 ± 0.21	**2.11 ± 0.23**	3.32 ± 0.40	3.84 ± 0.28
RF	OA	1.82 ± 0.11	**1.24 ± 0.09**	3.97 ± 0.16	4.32 ± 0.21	5.92 ± 0.06	3.74 ± 0.22
RF	F-Score	**1.32 ± 0.13**	5.05 ± 0.31	3.16 ± 0.22	3.05 ± 0.31	5.37 ± 0.14	3.05 ± 0.27
RF	G-Mean	**1.68 ± 0.22**	5.79 ± 0.21	3.26 ± 0.28	2.47 ± 0.30	3.89 ± 0.35	3.89 ± 0.19

Table 4 presents the mean CV scores. Except for the OA metric, G-SMOTENC either outperformed or matched the remaining oversamplers.

Table 4: Mean scores over the different datasets, folds and runs used in the experiment

Classifier	Metric	G-SMOTENC	NONE	SMOTENC	ROS	RUS	SMOTE-ENC
DT	OA	0.74 ± 0.05	**0.75 ± 0.04**	0.68 ± 0.04	0.66 ± 0.04	0.58 ± 0.04	0.65 ± 0.04
DT	F-Score	**0.56 ± 0.04**	0.52 ± 0.04	0.54 ± 0.04	0.52 ± 0.04	0.48 ± 0.04	0.51 ± 0.04
DT	G-Mean	**0.69 ± 0.03**	0.60 ± 0.02	0.68 ± 0.03	0.67 ± 0.03	0.65 ± 0.03	0.66 ± 0.03
KNN	OA	0.69 ± 0.04	**0.73 ± 0.05**	0.67 ± 0.04	0.69 ± 0.05	0.57 ± 0.04	0.68 ± 0.05
KNN	F-Score	**0.53 ± 0.04**	0.50 ± 0.04	0.52 ± 0.04	0.52 ± 0.04	0.46 ± 0.04	0.51 ± 0.04
KNN	G-Mean	**0.66 ± 0.03**	0.58 ± 0.03	0.64 ± 0.03	0.62 ± 0.03	0.65 ± 0.03	0.63 ± 0.03
LR	OA	0.68 ± 0.05	**0.75 ± 0.04**	0.68 ± 0.05	0.66 ± 0.05	0.58 ± 0.04	0.67 ± 0.04
LR	F-Score	**0.54 ± 0.04**	0.52 ± 0.04	**0.54 ± 0.04**	0.53 ± 0.04	0.48 ± 0.04	0.52 ± 0.04
LR	G-Mean	**0.69 ± 0.02**	0.60 ± 0.03	0.68 ± 0.02	**0.69 ± 0.03**	0.67 ± 0.03	0.67 ± 0.03
RF	OA	0.74 ± 0.04	**0.76 ± 0.04**	0.69 ± 0.04	0.69 ± 0.04	0.59 ± 0.04	0.68 ± 0.05
RF	F-Score	**0.57 ± 0.04**	0.48 ± 0.04	0.55 ± 0.04	0.55 ± 0.04	0.49 ± 0.04	0.53 ± 0.04

Continued on next page

Table 4: Mean scores over the different datasets, folds and runs used in the experiment

Classifier	Metric	G-SMOTENC	NONE	SMOTENC	ROS	RUS	SMOTE-ENC
RF	G-Mean	**0.70 ± 0.02**	0.57 ± 0.02	0.68 ± 0.03	0.69 ± 0.03	0.68 ± 0.03	0.68 ± 0.02

5.2. Statistical Analysis

It is necessary to use methods that account for the multiple comparison problem to conduct an appropriate statistical analysis in an experiment with multiple datasets. Based on the recommendations found in [8], we applied a Friedman test followed by a Holm-Bonferroni test for post-hoc analysis.

In Section 4.3 we explained that OA, although easily interpretable, is not an appropriate performance metric for imbalanced learning problems. Therefore, the statistical analysis was developed using the two imbalance-appropriate metrics used in the study: F-Score and G-Mean. Based on the Friedman test [16], there is a statistically significant difference in performance across resampling methods. The results of this test are shown in Table 5. The null hypothesis is rejected in all cases.

Table 5: Results for the Friedman test. Statistical significance is tested at a level of $\alpha = 0.05$. The null hypothesis is that there is no difference in the classification outcome across resamplers.

Classifier	Metric	p-value	Significance
DT	F-Score	2.2e-10	True
DT	G-Mean	1.2e-10	True
KNN	F-Score	2.3e-09	True
KNN	G-Mean	9.4e-10	True
LR	F-Score	2.1e-07	True
LR	G-Mean	9.7e-11	True
RF	F-Score	8.5e-12	True
RF	G-Mean	2.0e-10	True

We performed a Holm-Bonferroni test to understand whether the difference in the performance of G-SMOTENC is statistically significant to the remaining resampling methods. The results of this test are shown in Table 6. The null hypothesis is rejected in 33 out of 40 trials.

Table 6: Adjusted p-values using the Holm-Bonferroni test. Statistical significance is tested at a level of $\alpha = 0.05$. The null hypothesis is that the benchmark methods perform similarly to the control method (G-SMOTENC).

Classifier	Metric	NONE	SMOTENC	ROS	RUS	SMOTE-ENC
DT	F-Score	**1.5e-04**	**1.5e-04**	**7.3e-06**	**1.2e-06**	1.0e-01
DT	G-Mean	**5.6e-07**	**2.7e-03**	**2.8e-02**	**3.9e-04**	**2.3e-02**
KNN	F-Score	**6.4e-04**	**2.2e-04**	**7.2e-04**	**6.4e-04**	**5.9e-06**
KNN	G-Mean	**1.6e-05**	**9.6e-03**	**6.5e-03**	2.0e-01	**3.5e-03**
LR	F-Score	**4.0e-03**	6.1e-01	**9.2e-03**	**3.6e-04**	5.6e-02
LR	G-Mean	**1.6e-07**	**4.0e-04**	8.6e-01	2.4e-01	**4.7e-03**
RF	F-Score	**1.7e-06**	**2.4e-04**	**8.0e-03**	**1.7e-06**	**8.0e-03**
RF	G-Mean	**3.8e-06**	**8.8e-03**	2.5e-01	**2.3e-02**	**1.7e-03**

5.3. Discussion

The results reported in Section 5.1 show that G-SMOTENC consistently outperforms the remaining well-known oversampling approaches. Considering the results for the two imbalanced learning appropriate metrics in Table 3, G-Mean and F-Score, G-SMOTENC was only (on average) outperformed once by a neglectable margin. Unlike the results reported in [29], SMOTE-ENC's performance was rarely superior to SMOTENC's.

The relative difference in the classifiers' performance is better visible in Table 4. Using an RF classifier, for example, the impact of using G-SMOTENC compared to no oversampling improves, on average, 13 percentual points on G-mean and nine percentual points using F-Score.

The difference in performance among the different oversamplers was found to be statistically significant across the different classifiers and relevant performance metrics by performing a Friedman test. The p-values of this test are reported in Table 5. The superiority of G-SMOTENC was confirmed by the p-values obtained with the Holm-Bonferroni test shown in Table 6. This test showed that G-SMOTENC outperformed with statistical significance the remaining resamplers in 82.5% of the comparisons done.

The results from this experiment expose some well-known limitations of SMOTE, which become particularly evident with SMOTENC. Specifically, the lack of diversity in the generated data and, on some occasions, the near-duplication of observations discussed in [9] may be a possible explanation for the performance of SMOTENC being comparable to ROS' performance, visible in Figure 3. In this figure, three groups of resampling methods with comparable performance are visible: (1) G-SMOTENC, the top-performing method, (2) SMOTENC, ROS and SMOTE-ENC, where SMOTE-ENC has the most inconsistent behavior and (3) RUS and no oversampling, the worst-performing approaches. In addition, G-SMOTENC's superiority seems invariable to the dataset's characteristics, with little overlap with the remaining benchmark methods.

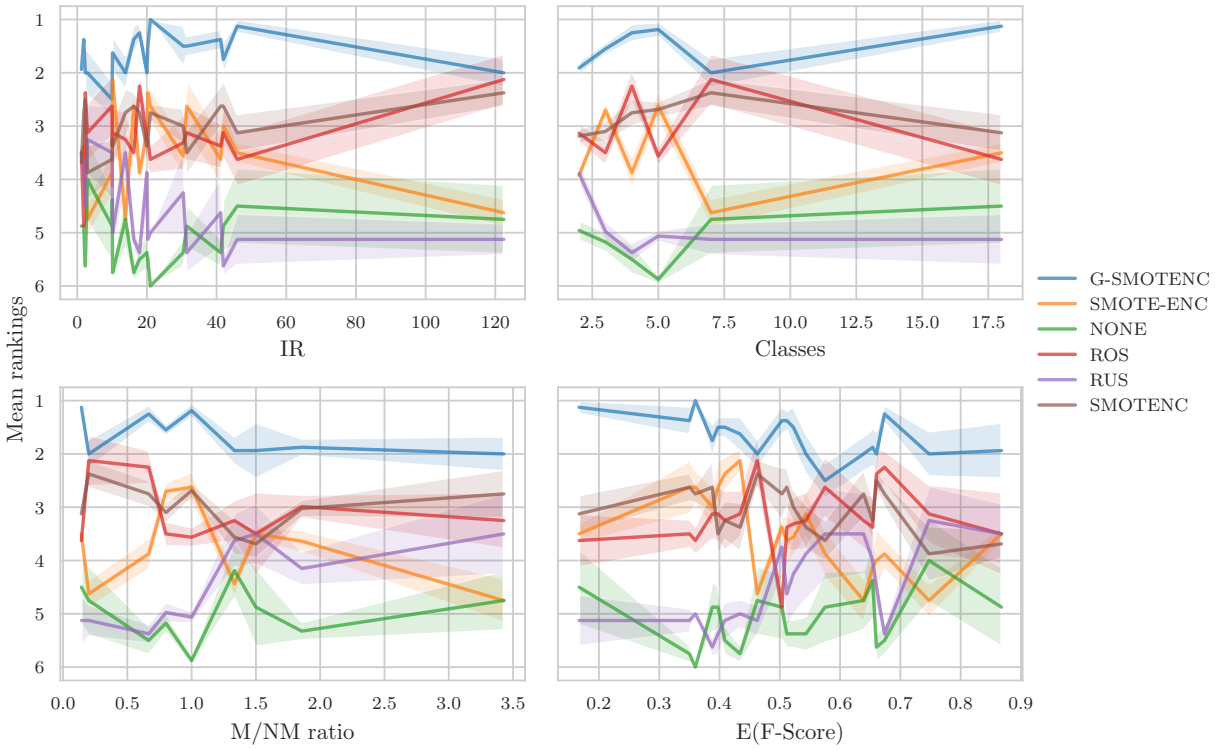


Figure 3: Average ranking of oversamplers over different characteristics of the datasets used in the experiment. Legend: IR — Imbalance Ratio, Classes — Number of classes in the dataset, M/NM ratio — ratio between the number of metric and non-metric features, E(F-Score) — Mean F-Score of dataset across all combinations of classifiers and oversamplers.

6. Conclusion

This paper presented G-SMOTENC, a new oversampling algorithm that combines G-SMOTE and SMOTENC. This oversampling algorithm leverages G-SMOTE's data selection and generation mechanisms into datasets with mixed data types. This was achieved by encoding and generating nominal feature values using SMOTENC's approach. The quality of the data generated with G-SMOTENC was tested over 20 datasets with different imbalance ratios, metric to non-metric feature ratios and number of classes. These results were compared to no oversampling, SMOTENC, Random Oversampling, Random Undersampling and SMOTE-ENC using a Decision Tree, K-Nearest Neighbors, Logistic Regression and Random Forest as classifiers.

G-SMOTENC can be seen as a drop-in replacement of SMOTENC, since when $\alpha_{trunc} = 1$, $\alpha_{def} = 1$ and $\alpha_{sel} = minority$, SMOTENC is reproduced. G-SMOTENC has three additional hyperparameters that allow for greater customization of the selection and generation mechanisms. However, determining the optimal parameters a priori (*i.e.*, with reduced parameter tuning) is a topic for future work.

The results show that G-SMOTENC performs significantly better when compared to its more popular counterparts (SMOTENC, Random Oversampling and Random Undersampling), as well as a recently proposed oversampling algorithm for mixed data types (SMOTE-ENC). This performance improvement is related to G-SMOTENC's selection mechanism, which finds a safer region for data generation, along with its generation mechanism which increases the diversity of the generated observations compared to SMOTENC. The G-SMOTENC implementation used in this study is available in the open-source Python library "ML-Research" and is fully compatible with the Scikit-Learn ecosystem.

Funding: This research was supported by a research grant of the Portuguese Foundation for Science and Technology ("Fundação para a Ciência e a Tecnologia"), reference SFRH/BD/151473/2021.

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

References

[1] Ambai, K., Fujita, H., 2018. Mndo: Multivariate normal distribution based oversampling for binary classification., in: Advancing Technology Industrialization Through Intelligent Software Methodologies, Tools and Techniques: Proceedings of the 17th International Conference on New Trends in Intelligent Software Methodologies, Tools and Techniques (SoMeT), pp. 425–438.

[2] Ambai, K., Fujita, H., 2019. Multivariate normal distribution based over-sampling for numerical and categorical features, in: Advancing Technology Industrialization Through Intelligent Software Methodologies, Tools and Techniques: Proceedings of the 18th International Conference on New Trends in Intelligent Software Methodologies, Tools and Techniques (SoMeT), p. 107.

[3] Bansal, A., Jain, A., 2021. Analysis of focussed under-sampling techniques with machine learning classifiers, in: 2021 IEEE/ACIS 19th International Conference on Software Engineering Research, Management and Applications (SERA), IEEE. pp. 91–96.

[4] Batista, G.E., Prati, R.C., Monard, M.C., 2004. A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD explorations newsletter 6, 20–29.

[5] Bunkhumpornpat, C., Sinapiromsaran, K., Lursinsap, C., 2009. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem, in: Pacific-Asia conference on knowledge discovery and data mining, Springer. pp. 475–482.

[6] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research 16, 321–357.

[7] Das, S., Datta, S., Chaudhuri, B.B., 2018. Handling data irregularities in classification: Foundations, trends, and future challenges. Pattern Recognition 81, 674–693.

[8] Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30.

[9] Douzas, G., Bacao, F., 2019. Geometric smote a geometrically enhanced drop-in replacement for smote. Information Sciences 501, 118–135.

[10] Douzas, G., Bacao, F., Fonseca, J., Khudinyan, M., 2019. Imbalanced learning in land cover classification: Improving minority classes' prediction accuracy using the geometric smote algorithm. Remote Sensing 11, 3040.

[11] Douzas, G., Bacao, F., Last, F., 2018. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. Information Sciences 465, 1–20.

[12] Fernández, A., Garcia, S., Herrera, F., Chawla, N.V., 2018. Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. Journal of artificial intelligence research 61, 863–905.

[13] Fernández, A., LóPez, V., Galar, M., Del Jesus, M.J., Herrera, F., 2013. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. Knowledge-based systems 42, 97–110.

[14] Fonseca, J., Douzas, G., Bacao, F., 2021a. Improving imbalanced land cover classification with k-means smote: Detecting and oversampling distinctive minority spectral signatures. Information 12, 266.

[15] Fonseca, J., Douzas, G., Bacao, F., 2021b. Increasing the effectiveness of active learning: Introducing artificial data generation in active learning for land use/land cover classification. Remote Sensing 13, 2619.

[16] Friedman, M., 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the american statistical association 32, 675–701.

[17] Gonog, L., Zhou, Y., 2019. A review: generative adversarial networks, in: 2019 14th IEEE conference on industrial electronics and applications (ICIEA), IEEE. pp. 505–510.

[18] Han, H., Wang, W.Y., Mao, B.H., 2005. Borderline-smote: a new over-sampling method in imbalanced data sets learning, in: International conference on intelligent computing, Springer. pp. 878–887.

[19] He, H., Bai, Y., Garcia, E.A., Li, S., 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning, in: 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), IEEE. pp. 1322–1328.

[20] Japkowicz, N., 2013. Assessment metrics for imbalanced learning. Imbalanced learning: Foundations, algorithms, and applications , 187–206.

[21] Jeni, L.A., Cohn, J.F., De La Torre, F., 2013. Facing imbalanced data–recommendations for the use of performance metrics, in: 2013 Humaine association conference on affective computing and intelligent interaction, IEEE. pp. 245–251.

[22] Jo, W., Kim, D., 2022. Obgan: Minority oversampling near borderline with generative adversarial networks. Expert Systems with Applications 197, 116694.

[23] Kaur, H., Pannu, H.S., Malhi, A.K., 2019. A systematic review on imbalanced data challenges in machine learning: Applications and solutions. ACM Computing Surveys (CSUR) 52, 1–36.

[24] Koivu, A., Sairanen, M., Airola, A., Pahikkala, T., 2020. Synthetic minority oversampling of vital statistics data with generative adversarial networks. Journal of the American Medical Informatics Association 27, 1667–1674.

[25] Lemaître, G., Nogueira, F., Aridas, C.K., 2017. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. Journal of Machine Learning Research 18, 1–5.

[26] Liang, X., Jiang, A., Li, T., Xue, Y., Wang, G., 2020. Lr-smote—an improved unbalanced data set oversampling based on k-means and svm. Knowledge-Based Systems 196, 105845.

[27] López, V., Fernández, A., García, S., Palade, V., Herrera, F., 2013. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. Information sciences 250, 113–141.

[28] Lumijärvi, J., Laurikkala, J., Juhola, M., 2004. A comparison of different heterogeneous proximity functions and euclidean distance, in: MEDINFO 2004, IOS Press. pp. 1362–1366.

[29] Mukherjee, M., Khushi, M., 2021. Smote-enc: A novel smote-based method to generate synthetic data for nominal and continuous features. Applied System Innovation 4, 18.

[30] Park, S., Park, H., 2021. Combined oversampling and undersampling method based on slow-start algorithm for imbalanced network traffic. Computing 103, 401–424.

[31] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

[32] Rout, N., Mishra, D., Mallick, M.K., 2018. Handling imbalanced data: a survey, in: International proceedings on advances in soft computing, intelligent systems and applications. Springer, pp. 431–443.

[33] Salazar, A., Vergara, L., Safont, G., 2021. Generative adversarial networks and markov random fields for oversampling very small training sets. Expert Systems with Applications 163, 113819.

[34] Sun, Y., Wong, A.K., Kamel, M.S., 2009. Classification of imbalanced data: A review. International journal of pattern recognition and artificial intelligence 23, 687–719.

[35] Tarekegn, A.N., Giacobini, M., Michalak, K., 2021. A review of methods for imbalanced multi-label classification. Pattern Recognition 118, 107965.

[36] Tyagi, S., Mittal, S., 2020. Sampling approaches for imbalanced data classification problem in machine learning, in: Proceedings of ICRIC 2019. Springer, pp. 209–221.

[37] Vuttipittayamongkol, P., Elyan, E., Petrovski, A., 2021. On the class overlap problem in imbalanced data classification. Knowledge-based systems 212, 106631.