Comparing the performance of oversampling techniques in combination with a clustering algorithm for imbalanced learning

Georgios Douzas*[1], Fernando Bacao[1]

[1]NOVA Information Management School, Universidade Nova de Lisboa

*Corresponding Author

Postal Address: NOVA Information Management School, Campus de Campolide, 1070-312 Lisboa, Portugal

Telephone: +351 21 382 8610

Imbalanced learning constitutes a recurring problem in machine learning. Frequently, practitioners and researchers have access to large amounts of data but its imbalanced nature hampers the possibility of building accurate robust predictive models. Many problems are characterized by the rare nature of the cases of interest, in domains such as health, security, quality control business applications, it is frequent to have huge populations of negative cases and just a very limited number of positive ones. The prevalence of the imbalanced learning problem explains why it continues to be a very active research topic, which in fact has grown in recent years. It is known that imbalance can exist both between-classes (the imbalance occurring between the two classes) and also within-classes (the imbalance occurring between sub-clusters of each class). In this later case traditional oversamplers will tend to perform poorly. In this paper we focus on testing the relevancy of using clustering procedures in combination with oversamplers to improve the quality of the results. We perform a series of extensive tests using the most popular oversamplers, with and without a preceding clustering procedure. The tests confirm that the use of a clustering procedure prior to the application of an oversampler will improve the results.

1 Introduction

Imbalanced learning can be described as the condition in which there is a significant difference between examples of different classes, in other words, when the classes are significantly skewed. Imbalanced learning is a widespread problem in the application of supervised learning in many different domains and applications [Fernández et al., 2013, Haixiang et al., 2017] and can be seen as a particular type of data scarcity.

Imbalanced learning affects the performance of classifiers due to two main reasons. The first, relates with the fact that, by design, standard learning methods are biased towards the majority class. This happens because, during the training phase, most learning algorithms assign similar costs to misclassification errors on the majority and minority classes. Consequently, the most abundant class tends to dominate the process, as the minority classes contribute less to the maximisation of the objective function (i.e. accuracy). The second, relates with the fact that to learn a classifier the learner needs to be able to

discriminate the classes; thus, having a small number of examples from one of the classes will make the learning task much more challenging.

Another relevant issue in imbalanced learning is related with misclassification costs. While learning a classifier in training both errors (i.e. misclassification of the majority and minority classes) are treated as similar, this is seldom the case in when the classifier is deployed for use. The norm, in most real world applications, is that the misclassification of positive cases (i.e. minority class) is much more expensive than the misclassifications of the negative cases (i.e. the majority class). This is the rule rather than the exception, in medical diagnoses or fraud detection, for instance, misclassifying positive cases as negative ones is much more expensive than the inverse situation (REF NEEDED).

We can divide the options to handle the negative effects of imbalance learning into three main approaches [Fernández et al., 2013]. The first one is by means of changing the cost function of the algorithm [Wu and Chang, 2005], so that it severely penalizes false negatives. Changing the cost function can be a painstaking process, especially if the user is considering making use of several different algorithms, a common practice nowadays. A second approach relies on designing new or changing existing algorithms to take into account the relevance of the minority class [Chawla et al., 2008]. Again, this can be a difficult and lengthy procedure, restricting the algorithm options available to the user. Finally, the last option is to modify the data, by including a preprocessing step to correct the skewness of the class distribution (REF). This can be done by generating new synthetic data instances of the minority classes, a process usually referred to as oversampling, or by removing instances from the majority class, known as undersampling. The advantage of this last option is that, by dealing with the problem at the data level, once the imbalance is corrected, the user can use any (and as many) algorithm, without need for any further changes. In this paper we focus on the oversampling strategy because it allows for the use of any algorithm without any need change it and, contrary to undersampling, it doesn't discard any available information.

For the purposes of oversampling it is important to understand that the imbalance learning problem encompasses two different sub-problems: between-class imbalance and within-class imbalance [Jo and Japkowicz, 2004]. Between-class refers to the skewness in the distribution between majority and minority classes. Within-class imbalance is a less obvious, but equally important, problem and refers to the possible existence of several sub-clusters of minority or majority instances. Closely related with within-class imbalance is the "small disjuncts problem". Small disjuncts [Galar et al., 2012]; [Weiss and Provost, 2003] that occurs when the minority concept is represented by subconcepts, usually small subclusters scattered through the input space. This adds complexity to the imbalance learning problem as it is necessary to guaranty that all the subconcepts are represented in the data such that they can be learned by the algorithm.

Many oversampling methods have been proposed and have proven to be effective in real-word domains. Particularly relevant is SMOTE [Chawla et al., 2002], which was the first algorithm proposed and which considerably improved upon random oversampling. Since the initial SMOTE paper many variations have been proposed in order to improve some of the weaknesses of the original algorithm. However, many of these approaches suffer either from being too complex and not readily available to practitioners and researchers or being too focused on a single vulnerability of SMOTE.

In order to mitigate the within-class imbalance problem, some algorithms proposed the use of clustering techniques before the oversampling procedure. The use of clustering should enable the oversampling algorithms to identify and target areas of the input space where the generation of artificial data is most effective and needed. This way, it is expected that the algorithms are able to address, both, between-class and within-class imbalances, while at the same time avoiding the generation of noisy samples. The goal of this paper is to assess the impact and effectiveness of adding a clustering procedure, prior to the oversampling step, in the imbalanced learning problem. The results of this extra clustering step and its value are explored by analyzing how each individual oversampling algorithm, with and without the

clustering step, fares against each other.

The remainder of this work is organized as follows. In section 2, we summarize the related literature and introduce some of the currently available oversampling methods, focusing on oversamplers that employ a clustering procedure. In section 3, the experimental framework is presented, while in section 4 the experimental results are presented. Finally, in section 5 we present the main conclusions of the study.

2 State of the art

As noted in the introduction the approaches to deal with imbalanced learning tasks are broadly divided into three main groups: changing the cost function of the algorithm [Wu and Chang, 2005]; designing new or changing existing algorithms to take into account the relevance of the minority class [Chawla et al., 2008]; modify the data, by including a preprocessing stage to correct the skewness of the class distribution (REF). Therefore, there are two options that operate at the level of the algorithm and the other one operates ate the data level. Clearly, this last option is more general, in the sense that once the data imbalance issues are solved any "off-the shelf" algorithm can be used, without the need for any additional modification.

The aforementioned data level solutions include different forms of resampling. Resampling is the process of manipulating the distribution of the training examples in an effort to improve the performance of classifiers [Jo and Japkowicz, 2004]. Generally, resampling methods can be categorized as undersampling and oversampling. Undersampling reduces the number of majority class samples by removing samples from the training set. On the other hand, oversampling works by generating synthetic examples for the minority class and adding them to the training set. It has been shown that both oversampling and undersampling can be effective depending on the specific problem that is being addressed [Chawla et al., 2002]. Both of these methods can be further categorized into random and heuristic approaches. It might be argued that while oversampling adds information to the problem, undersampling excludes information from the learning process which may negatively affect the performance of the classifier in cases where the data set is small [He et al., 2008]. On the other hand, depending on the quality of the generation procedure, oversampling, by generating synthetic examples, can lead to overfitting.

Several oversampling techniques have been proposed and studied in the past (MANY REFS). The simplest approach, which has been proven to perform well, is as Random Oversampling. This method works uninformed and aims to balance class distribution through the random replication of minority class examples. Given that the examples are merely replicated, the likelihood of overfitting occurring increases significantly [Batista et al., 2004]. In 2002, as an attempt to add information to the training data, [Chawla et al., 2002] proposed an alternative oversampling method called SMOTE (Synthetic Minority Oversampling Technique). Instead of replicating existing observations, synthetic samples are generated. This is achieved by linear interpolation between a randomly selected sample of the minority class and one of its minority neighboring observations [Douzas et al., 2018] [Fernandez et al., 2018] [Liu et al., 2007]. SMOTE was the first method proposed to improve random oversampling and still is the most popular oversampling method.

However, it is important to note that the skewed class distribution (i.e. between-class imbalance) is not the only difficulty posed by imbalanced datasets to deteriorate performance in supervised learning algorithms. The distribution of the data within each class (i.e. within-class imbalance) is also relevant.

The fact that SMOTE randomly chooses a minority instance to oversample with uniform probability allows for an effective solution combating between-class imbalance, leaving other issues such as within-class and small disjuncts unsolved. Input areas containing a large number of minority samples have a high probability of being populated further, while there can be underrepresented concepts located in

small areas of the data that are likely to remain sparse [Fernandez et al., 2018]. Another concern is the fact that the method is susceptible to noise generation because it doesn't distinguish overlapping class regions from so-called safe areas (REF).

Despite its weaknesses, the SMOTE is still considered the standard in the framework of learning from imbalanced data. In order to mitigate its disadvantages and improve its performance under the different possible situations, several modifications and extensions have been proposed throughout the years. They usually address a specific weakness from the original method, such as emphasizing certain minority class regions, combating within-class imbalance, or even attempting to avoid noise generation [Douzas et al., 2018].

The most frequent properties exploited by the techniques are the initial selection and adaptive generation of synthetic examples. Filtering is becoming more common in recent years, as well as the use of kernel functions. Regarding the interpolation procedure, it is also usual to replace the original method with other more complex ones, such as clustering-based or derived from a probabilistic function [Fernandez et al., 2018]. Safe-Level SMOTE modifies the SMOTE algorithm by applying a weight degree, the safe level, in the data generation process. The safe level provides a scale to differentiate between noisy and safe instances [Bunkhumpornpat et al., 2009].

Similarly, there are two other enhancements of SMOTE called Borderline SMOTE1 and Borderline SMOTE2, in which only the minority examples near the borderline are oversampled. For the minority class, experiments show that our approaches achieve better TP rate and F-value than SMOTE and random over-sampling methods [Han et al., 2005]. Along with this variation, MWMOTE (Majority Weighted Minority Oversampling Technique for Imbalanced Data Set Learning) [Barua et al., 2014], and its variation KernelADASYN [Tang and He, 2015] aim to achieve the same result. G-SMOTE [Douzas and Bacao, 2019] improves the diversity of generated samples by linearly interpolating generated samples between two minority class instances. G-SMOTE extends the linear interpolation mechanism by introducing a geometric region where the data generation process occurs. The methods above described deal with the between-class imbalance problem and as previously mentioned, there can be two kinds of imbalance present in a data set. To solve this, there are clustering-based methods proposed to effectively reduce not only the between-class imbalance but also the within-class imbalance and to oversample the data set by rectifying these two types of imbalances simultaneously. Firstly, they divide the input space into clusters and in a posterior phase use sampling methods to adjust the size of the newly built clusters [Douzas and Bacao, 2017] [Jo and Japkowicz, 2004]. Among these clustering-based approaches there is cluster-based oversampling. The algorithm applies random oversampling, after clustering the training examples in the minority and the majority classes, so that the majority and minority classes are of the same size [Jo and Japkowicz, 2004]. Nonetheless, since the approach does not generate new data and it merely replicates already existing samples, it is prone to overfitting. Cluster-SMOTE [Cieslak et al., 2006] initially applies the k-means algorithm to the minority class and then SMOTE is used in the clusters in order to generate artificial data. Similarly, DBSMOTE [Bunkhumpornpat et al., 2012] uses the DB-SCAN algorithm to discover arbitrarily shaped clusters and then generates synthetic instances along a shortest path from each minority class instance to a pseudo-centroid of the cluster. As a more sophisticated approach, there is A-SUWO [Nekooeimehr and Lai-Yuen, 2016] which creates clusters of the minority class instances with a size, which is determined using cross-validation and generates synthetic instances based on a proposed weighting system. SOMO (Self-Organizing Map Oversampling) [Douzas and Bacao, 2017] creates a two- dimensional representation of the input space and based on it, applies the SMOTE procedure to generate intra-cluster and inter-cluster synthetic data, preserving the underlying manifold structure. Another clustering-based approach called CURE-SMOTE [Ma and Fan, 2017] uses the hierarchical clustering algorithm CURE to cluster the samples of the minor class and remove the noise and outliers before applying SMOTE. The goal of the method is to eliminate the noise points at the end of the process and reduce the complexity because there is no need to eliminate the farthest generated artificial samples after the SMOTE algorithm runs. While it avoids noise generation, possible imbalances within the minority class are ignored. Consequently,

another clustering-based approach that was introduced named K-Means SMOTE [Douzas et al., 2018] employs the popular k-means clustering algorithm in conjunction with SMOTE oversampling in order to avoid the generation of noise by oversampling only in safe area and shifting its focus not only to fix between-class imbalance but also within-class imbalance. The method attempts to deal with the small disjuncts problem by inflating sparse minority areas and is easily implemented due to its simplicity and the widespread availability of both k-means and SMOTE.

3 Experimental Setup

In this section we describe the experimental setup used to test the relevance of using a clustering procedure before the oversampling task. In order to make sure the results are robust and generalizable we used a variety of classifiers, datasets and metrics, to assess the relevance of this strategy. The general design of the experiment is provided in Figure 1. A detailed description of the experimental data, the performance metrics, the classification algorithms and the experimental procedure is provided next.

FIGURE 1 about here

3.1 Datasets

In order to test the performance of the different oversamplers, 13 imbalanced datasets from Machine Learning Repository UCI were used. Furthermore, to generate additional datasets with higher imbalance ratios, each of the aforementioned datasets was randomly undersampled to generate various other datasets. Table 2 shows a summary of the datasets and their characteristics. NOTE: Undersampled versions of the original datasets are omitted from the table we need to add them!

Table 1 - Description of the datasets ABOUT HERE

3.2 Evaluation Metrics

The most popular evaluation metrics used to assess classifier performance are not suitable in the presence of a skewed class distribution. The most commonly used metric is classification problems is accuracy:

ACURACCY FORMULA HERE

Where TP represents the true positives, TN the true negatives, P the total number of positives and N the total number of negatives. The problem of using accuracy to assess the performance of a classifier in imbalanced datasets can be better explained with a simple example. Let's say we have a highly imbalanced dataset, with 99 percent negative instances and 1 percent positive instances, in this case a trivial classifier, that always classifies examples as negative, will yield 99 percent accuracy. Although extremely high this accuracy value is meaningless, as the classifier will never identify a positive case.

In the case of imbalanced datasets there are better alternatives to accuracy. The most common measures to assess the performance of classifiers when dealing with imbalanced problems are: F1 score, G-Mean and ROC-AUC [He and Garcia, 2009]. The F1 score, or F-Measure metric, combines precision and recall as a measure of the effectiveness of classification in terms of a ratio of the weighted importance on either recall or precision as determined by the β coefficient set by the user.

F1 FORMULA HERE

The G-Mean metric can be defined as the geometric mean of sensitivity and specificity [He and Garcia, 2009].

G-MEAN FORMULA HERE

Finally, the last metric to be considered is the ROC-AUC (Area Under the ROC Curve). The ROC curve is obtained by plotting the False Positive Rate (FPR), defined as the proportion of misclassified negative examples relative to the total number of negative class observations, represented in the X-axis versus the Y-axis, the True Positive Rate. Varying the classification threshold of the classifier identifies different points of the ROC curve. Since the ROC curve depends on the classification threshold, the AUC (Area Under the ROC Curve) is a useful metric for the performance of the classifier as it is independent of the decision criterion [He and Garcia, 2009].

3.3 Oversamplers

In the tests we use 4 different oversamplers, with and without a clustering procedure, before the oversampling process. The oversamplers used were: Random Oversampling, SMOTE, Borderline SMOTE and G-SMOTE. For SMOTE and its two variations, more specifically Borderline SMOTE and G-SMOTE, the optimal value of k-nearest neighbors was set to $k \in \{3, 5\}$.

Furthermore, a hyper-parameter grid was generated for G-SMOTE including the three different selection strategies, the number of nearest neighbors $k \in \{3, 5\}$, the truncation factor $\alpha_{trunc} \in \{-1.0, 0.0, 1.0\}$ and the deformation factor $\alpha_{def} \in \{0.0, 0.5, 1.0\}$.

Regarding the clustering procedure we used to most popular clustering algorithm (i.e. k-means). The choice of the k-means algorithm was mainly due to its simplicity and popularity. To use a clustering procedure before the oversampling procedure requires the definition of 3 main parameters. First, the number of clusters to consider, as in any clustering procedure using k-means. The second paremeter, caled here $distances\_exponent$, relates with . Finally, the last paremeter, the $filtering\_threshold$, defines the . These parameters were tested with the following grid:

$$n\_clusters \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$$

$$distances\_exponent \in \{0, 1, 2\}$$

$$filtering\_threshold \in \{0.5, 1.0\}$$

In summary, the four oversamplers presented above are tested with and without a clustering procedure based on k-means.

3.4 Classifiers

In order to understand if the effect of using a clustering procedure preceding the application of an oversample would be different, and to make the results as general as possible we decided to use 4 different classifiers: Logistic Regression (LR), Gradient Boosting Classifier (GBC), K-Nearest Neighbors (KNN) and Decision Tree (DT).

LR is a generalization of linear regression, which can be used for binary classification. Fitting the model is an optimization problem which can be solved using simple optimizers requiring no hyperparameters to be tuned [McCullagh and Nelder, 1989]. Because of this, results produced by LR are conveniently reproducible, and therefore, appropriate to be used as a benchmark for more sophisticated approaches.

The KNN algorithm is a classical classifier and assigns an observation to the most represented class of its nearest neighbors. The number of neighbors that are considered is determined by the method's hyperparameter k [Fix and Hodges, 1989].

DT's [Breiman et al., 1984] are also among the most popular and simple classifiers used in machine learning, nevertheless they continue to be frequently used, especially when the interpretation of the results needs to be simple and readily available.

GBC is an ensemble technique used for classification, which as become very popular in recent years due to the quality of its results. In the case of binary classification, one shallow decision tree is induced at each stage of the algorithm. Each tree is fitted to observations which could not be correctly classified by decision trees of previous stages. Predictions of GBC are made by majority vote of all trees. In this way, the algorithm combines several simple models (referred to as weak learners) to create one effective classifier. The number of decision trees to generate, which in binary classification is equal to the number of stages, is a hyperparameter of the algorithm [Friedman, 2001].

Different combinations of hyper-parameters were used and tested for each of the classifiers mentioned above. The classifiers are used with the default parameters unless stated otherwise. More specifically, the GBC hyper-parameter grid included the combinations resulting from $max\_depth \in \{3, 6\}$ and $number\_estimators \in \{50, 100\}$. For the DT Classifier, the combinations resulting from $max\_depth \in \{3, 6\}$ are tested. Additionally, for KNN, the $k \in \{3, 5\}$.

3.5 Experimental Procedure

In order to evaluate the performance of the algorithms the results are obtained by applying k-fold cross-validation with k = 3. For each each dataset, every metric is computed by averaging their values across the different runs. In addition to the arithmetic mean, the standard deviation is calculated. Furthermore, in order to achieve optimal results for all classifiers and oversamplers, a grid search procedure is used. All possible combinations of an algorithm's hyperparameters are generated and the algorithms are executed once for each combination. All metrics are used to score all resulting classifications, and the best value obtained for each metric is saved. Summing up, the experimental procedure was repeated three times and the implementation of the classifiers and standard oversampling algorithms was based on the Python library Scikit-Learn [Pedregosa et al., 2011].

4 Experimental Results

Since the main objective of the analysis is to understand the effects of clustering preceding the application of the different oversampling techniques, results are shown for each individual technique without oversampling, with oversampling and with clustering and oversampling. Taking into consideration the recommendations for evaluating classifier performance across multiple datasets [Demšar, 2006], the scores obtained are not compared directly, but instead ordered so as to derive a ranking. Having said that, the goal is to compare oversamplers, hence the method being adapted to rank the oversampling techniques, instead of classification algorithms. To derive the rank order we use the cross-validated scores, assigning a ranking score to each oversampling method for every combination of the datasets, 3 metrics and the

4 classifiers. The result is a different ranking for each of the experiment repetitions, divided by dataset, metric and classifier.

Keeping in mind that there are four methods being compared, rank one will be attributed to the best performing method and rank four to the worst performing one. Moreover, each method's rank is averaged across all datasets and experiment repetition. A method's rank is a real number from the interval [1;4]. The mean ranking results for each combination and classifier, regarding each individual oversampling technique are shown in Figures 1-4.

By testing the null hypothesis that the classifiers perform similarly in the mean rankings across the oversampling methods and evaluation metrics, the Friedman test [Friedman, 1937] is used to determine the statistical significance of the derived mean ranking. The Friedman test was chosen because it does not assume normality of the obtained scores [Demšar, 2006]. At a significance level of a = 0.05, the null hypothesis is rejected for all evaluated classifiers and evaluation metrics. Therefore, all of the rankings are assumed to be significant. The results of the application of the Friedman test for each of the oversampling methods are shown in annex in Table 4.

4.1 Random Oversampling

Considering the mean ranking results when random oversampling is applied it is possible to observe that the random oversampler coupled with the clustering algorithm, which can be referred to as K-Means Random Oversampling, outperforms the other two methods (no oversampling and random oversampling) in all metrics and classifiers. In all possible twelve combinations, K-Means Random Oversampling achieves a superior mean rank (XXX). Furthermore, it is the only method with a mean ranking better than two in two particular metrics, F1 score and AUC. Additionally, in three out of nine cases, it can be seen that K-Random Oversampling boosts classification results when Random Oversampling accomplishes a rank worse than no oversampling. However, it is clear that when no oversampling is applied, the results tend to be the worst amongst the three methods.

GRAPHICS HERE

4.2 SMOTE

Similarly, when the clustering procedure is added before applying SMOTE, called here K-Means SMOTE, the method outperforms the remaining two (no oversampling and SMOTE). Significantly, this can be observed regardless of the metric or classifier since the improvement can be seen in eleven out of twelve cases, consistently achieving a better mean rank. The only case in which the improvement does not happen is in the combination of G-Mean and the KNN classifier where it accomplishes a slightly inferior rank.

GRAPHICS HERE

4.3 Borderline SMOTE

In the case of Borderline SMOTE, the evidence shows that the K-Means Borderline SMOTE algorithm outperforms the other two methods (no oversampling and Borderline SMOTE) in all metrics and regardless of the choice in classifiers. Similarly, it is possible to note that no oversampling provides the worst results.

GRAPHICS HERE

4.4 G-SMOTE

In the case of G-SMOTE using the k-means procedure previous to the oversampling still improves the results in the majority of the tests, but the results are not so impressive. In eight, out of the twelve experiments, achieves the best results, but in four of the experiments G-SMOTE without any previous clustering procedure presents the best score. These four cases relate with the use of KNN and GBC classifier and the AUC and G-mean metrics. One possible explanation is that the quality of the G-SMOTE oversampling strategy mitigates, by itself, the problem of having subclusters of minority instances in the dataset. This means that when using G-SMOTE the negative impact of having minority subclusters has a smaller impact in the quality of the results. In any case, using k-means before applying G-SMOTE improves the results in the majority of the cases.

GRAPHICS HERE

Overall it is clear that the use of k-means clustering procedure before the application of an oversampling algorithm significantly improves the quality of the results. The existence of minority subclusters in the input space weakens the ability of the oversampler to produce robust results. Starting with a procedure that divides the input space into several subspaces to guide the oversampling process seems to be of great relevancy and usefulness. From these extensive tests we can conclude that using clustering before oversampling adds value to the classification process. Generally, it can be observed that the combination frequently achieves the best results in comparison with the other methods (no oversampling and the oversampling technique with no prior clustering step). This benefit can be seen across all classifiers and metrics used, which adds strength to the conclusion. To summarize, the mean ranking results of the different methods across the datasets for each combination of a classifier and evaluation metric is presented in Table 3.

References

[Barua et al., 2014] Barua, S., Islam, M. M., Yao, X., and Murase, K. (2014). MWMOTE - Majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):405–425.

[Batista et al., 2004] Batista, G. E. A. P. A., Prati, R. C., and Monard, M. C. (2004). A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *ACM SIGKDD Explorations Newsletter - Special issue on learning from imbalanced datasets*, 6(1):20–29.

[Breiman et al., 1984] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. (1984). *Classification and Regression Trees*. Taylor & Francis, Boca Raton, United States.

[Bunkhumpornpat et al., 2009] Bunkhumpornpat, C., Sinapiromsaran, K., and Lursinsap, C. (2009). Safe-level-SMOTE: Safe-level-synthetic minority over-sampling TEchnique for handling the class imbalanced problem. In *Advances in Knowledge Discovery and Data Mining*, pages 475–482. Springer Berlin Heidelberg.

[Bunkhumpornpat et al., 2012] Bunkhumpornpat, C., Sinapiromsaran, K., and Lursinsap, C. (2012). DBSMOTE: Density-based synthetic minority over-sampling technique. *Applied Intelligence*, 36(3):664–684.

[Chawla et al., 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

[Chawla et al., 2008] Chawla, N. V., Cieslak, D. A., Hall, L. O., and Joshi, A. (2008). Automatically countering imbalance and its empirical relationship to cost. *Data Mining and Knowledge Discovery*, 17(2):225–252.

[Cieslak et al., 2006] Cieslak, D. A., Chawla, N. V., and Striegel, A. (2006). Combating imbalance in network intrusion datasets. In *2006 IEEE International Conference on Granular Computing*, pages 732–737.

[Demšar, 2006] Demšar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30.

[Douzas and Bacao, 2017] Douzas, G. and Bacao, F. (2017). Self-organizing map oversampling (somo) for imbalanced data set learning. *Expert Systems with Applications*, 82:40 – 52.

[Douzas and Bacao, 2019] Douzas, G. and Bacao, F. (2019). Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE. *Information Sciences*, 501:118–135.

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

[Fernandez et al., 2018] Fernandez, A., Garcia, S., Herrera, F., and V. Chawla, N. (2018). Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. 61:863–905.

[Fernández et al., 2013] Fernández, A., López, V., Galar, M., del Jesus, M. J., and 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.

[Fix and Hodges, 1989] Fix, E. and Hodges, J. L. (1989). Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3):238.

[Friedman, 2001] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

[Friedman, 1937] 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(200):675–701.

[Galar et al., 2012] Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. (2012). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484.

[Haixiang et al., 2017] Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., and Bing, G. (2017). Learning from class-imbalanced data. *Expert Syst. Appl.*, 73(C):220–239.

[Han et al., 2005] Han, H., Wang, W.-Y., and Mao, B.-H. (2005). Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In *Lecture Notes in Computer Science*, pages 878–887. Springer Berlin Heidelberg.

[He et al., 2008] He, H., Bai, Y., Garcia, E. A., and Li, S. (2008). ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1322–1328.

[He and Garcia, 2009] He, H. and Garcia, E. A. (2009). Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.

[Jo and Japkowicz, 2004] Jo, T. and Japkowicz, N. (2004). Class imbalances versus small disjuncts. *ACM SIGKDD Explorations Newsletter*, 6(1):40.

[Liu et al., 2007] Liu, A., Ghosh, J., and Martin, C. E. (2007). Generative oversampling for mining imbalanced datasets. In *DMIN*, pages 66–72.

[Ma and Fan, 2017] Ma, L. and Fan, S. (2017). CURE-SMOTE algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests. *BMC Bioinformatics*, 18(1).

[McCullagh and Nelder, 1989] McCullagh, P. and Nelder, J. (1989). *Generalized Linear Models*.

[Nekooeimehr and Lai-Yuen, 2016] Nekooeimehr, I. and Lai-Yuen, S. K. (2016). Adaptive semi-unsupervised weighted oversampling (A-SUWO) for imbalanced datasets. *Expert Systems with Applications*, 46:405–416.

[Pedregosa et al., 2011] 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., and Duchesnay, E. (2011). *Scikit-learn: Machine learning in Python*, volume 12.

[Tang and He, 2015] Tang, B. and He, H. (2015). KernelADASYN: Kernel based adaptive synthetic data generation for imbalanced learning. In *2015 IEEE Congress on Evolutionary Computation*. IEEE.

[Weiss and Provost, 2003] Weiss, G. M. and Provost, F. (2003). Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–354.

[Wu and Chang, 2005] Wu, G. and Chang, E. Y. (2005). Kba: Kernel boundary alignment considering imbalanced data distribution. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):786–795.