In-Depth Performance Analysis of SMOTE-Based Oversampling Algorithms in Binary Classification

Review

Mario Dudjak
J. J. Strossmayer University of Osijek,
Faculty of Electrical Engineering, Computer Science and Information Technology
Osijek, Croatia
mario.dudjak@ferit.hr

Goran Martinović
J. J. Strossmayer University of Osijek,
Faculty of Electrical Engineering, Computer Science and Information Technology
Osijek, Croatia
goran.martinovic@ferit.hr

Abstract *– In the field of machine learning, the problem of class imbalance considerably impairs the performance of classification algorithms. Various techniques have been proposed that seek to mitigate classifier bias with respect to the majority class, with simple oversampling approaches being one of the most effective. Their main representative is the well-known SMOTE algorithm, which introduces a synthetic instances creation mechanism as an interpolation procedure between minority instances. To date, an abundance of SMOTE-based extensions that intend to improve the original algorithm has been proposed. This paper aims to compare the performance of several such extensions. In addition to comparing the overall performance, the impact of the selected oversamplers on the per-class performance is also evaluated. Finally, this paper tries to interpret the obtained performance results with respect to the internal procedures of oversampling algorithms. Some interesting findings have been made in this regard.*

Keywords *– classification, class imbalance, SMOTE, sensitivity, specificity*

1. INTRODUCTION

The main goal of a classification is to identify the class to which the new data will be assigned. Although many approaches have been developed to tackle classification problems, the characteristics and size of the available datasets have always had a significant impact on the quality of the predictions, if not greater. One data property that notably complicates classification problems is the concept of class imbalance, according to which the label of one class is represented to a lesser degree. Classifier training over imbalanced datasets results in models that are strongly biased towards the majority class [1]. On top of that, the class with the minority of instances is usually the class of interest, resulting in more false positive predictions [2]. Such a result can have dire consequences in detecting faults or intrusions, problems arising from medical domains and several others.

For that reason, it should come as no surprise that the problem of class imbalance has gathered considerable research interest. There are two main approaches to addressing the problem of imbalanced data in the literature [3]: algorithm-level approaches and data-level approaches. The former imply that the standard classification algorithms are modified to enhance the learning task with respect to the minority class, while the latter adjust the class imbalance ratio in order to achieve a balanced distribution between classes. Due to their adaptability and simplicity, data-level approaches that either conduct undersampling of majority instances or oversampling of minority ones are more commonly used [4]. To avoid the elimination of important majority instances, oversampling algorithms represent the preferable choice. Their main representative is the SMOTE (synthetic minority oversampling technique) algorithm, proposed by Chawla et al. [5]. In the reminiscence of the origins of SMOTE [6], Chawla highlights two classification results that he sought to improve when developing this algorithm: (1) performance on minority class and (2) generalization capacity. The novelty of SMOTE was in its synthetic instance creation mechanism that tried to cope with challenges arisen from overfitting the minority class when performing random oversampling, the state-of-the-art method at that time.

To date, more than 85 variants of the SMOTE algorithm have been proposed to improve the original in terms of various classification metrics [6]. The inner procedures of the original algorithm most often subject to change are the initial selection and adaptive generation of synthetic examples, where some extensions incorporate filtering, integration with undersampling, kernels and relabeling. Given the considerable number of SMOTE-based extensions, there are surprisingly few studies in the literature that attempt to provide their in-depth comparison. The latest such studies have begun to place more emphasis on experimental analysis than on plain review. In analysis performed by Bajer et al. [7], six SMOTE-based algorithms are statistically compared in terms of the obtained geometric mean across large number of diverse imbalanced datasets and three different classifiers. An interesting conclusion of this analysis is that even a slight change in the synthetic instance creation method can alter the performance of the overall algorithm. More recent research was conducted by Kovács [8], who carried out a comprehensive comparison and evaluation of 85 variants of minority oversampling techniques across 104 imbalanced datasets and four classifiers.

In both studies, algorithms were compared using typical metrics to show overall classification performance. The conclusions of similar analyzes performed in the literature are often limited by the small number of datasets and algorithms used. With the increase in the number of datasets, the mean performance values become more realistic and it becomes more difficult to fine-tune the algorithms for seemingly better results, which is the impression sometimes given by the new propositions of oversampling algorithms. However, the analysis of the overall classification performance does not provide insight into a per-class performance. While oversamplers improve performance on the minority class, the question is how they affect the prediction of the majority class. The trade-off between the number of false negative and false positive predictions can be a significant performance indicator of a classifier and can be ameliorated using an oversampling algorithm. In numerous classification problems, attaining good performance on both minority and majority classes is crucial, and it would be of great benefit to determine if there are oversampling algorithms that are dominant for both classes. This paper attempts to give certain insights in that regard. The objective of the paper is twofold: (1) the overall performance of several SMOTE-based oversampling approaches was evaluated and analyzed, and (2) a method for per-class performance evaluation that integrates traditional classification metrics and techniques, from the domain of multi-objective optimization, has been proposed and conducted on the same set of oversampling algorithms.

The rest of the paper is organized as follows. The overview of oversampling approaches is given in Section 2, with an emphasis on the SMOTE algorithm. Extensions to SMOTE are briefly described and categorized by the interpolation mechanisms they implement. Section 3 presents the settings of the experimental analysis in terms of the datasets, classifiers, and performance metrics used. The comparison of selected algorithms is demonstrated in Section 4, through both overall and per-class performance aspects. In addition, the same comparisons were made for the interpolation mechanisms represented by the averaged performance results of the associated oversampling algorithms. Finally, Section 5 summarizes the reached conclusions and provides possible directions for future work.

2. OVERSAMPLING ALGORITHMS

Early research on oversampling [9, 10] has discussed oversampling with replacement, where the positive (minority) examples were duplicated with replacement to match the number of negative (majority) examples. Additionally, if minority instances are selected at random, this method is known as a simple random oversampling. Its main drawback is that it does not introduce new information to the data and can lead to overfitting [11]. The first method that was proposed to alleviate this problem was the SMOTE algorithm.

2.1. SYNTHETIC MINORITY OVERSAMPLING TECHNIQUE

In addition to improving the performance of the classifier on the minority class, SMOTE seeks to improve its generalization capacity, potentially eliminating the occurrence of overfitting. For this reason, it includes synthetic instance creation mechanism, according to which new minority instances are not duplicates of existing ones but are located in their neighborhood. For each minority instance x in M, first its k-neighborhood is determined, whereby various metric functions may be employed (typically the Euclidean distance is used). This is followed by the creation of q synthetic instances as the convex combination

$$\boldsymbol{s}^i := \boldsymbol{x} + U_i(0,1) \cdot (\hat{\boldsymbol{x}}^{r(i)} - \boldsymbol{x}), \quad i = 1, \ldots, q, \qquad (1)$$

where the created instance is placed on a line between the minority instance $\boldsymbol{x}$ and its random neighbor from the k-neighborhood $\hat{\boldsymbol{x}}^{r(i)}$, the exact placement being defined by a random number generated using a uniform distribution U_i (0,1). The complete SMOTE algorithm is outlined in Fig. 1.

Require: Minority set: $\mathcal{M}$; amount of synthetic data: q; number of nearest neighbours: k;
1: $\mathcal{S} = \emptyset$ {Synthetic data set}
2: **for all** $\mathbf{x} \in \mathcal{M}$ **do**
3: determine the k-neighbourhood $\mathcal{N}_k(\mathbf{x})$ of $\mathbf{x}$ {Synthetic instances creation}
4: **for** $i = 1 : q$ **do**
5: randomly select $\hat{\mathbf{x}}^r \in \mathcal{N}_k(\mathbf{x})$
6: $\mathbf{s}^i := \mathbf{x} + U_i(0,1) \cdot (\hat{\mathbf{x}}^{r(i)} - \mathbf{x})$
7: $\mathcal{S} = \mathcal{S} \cup \mathbf{s}^i$
8: **end for**
9: **end for**
10: **return** $\mathcal{S}$

Fig. 1. Outline of SMOTE [7]

While developing a synthetic instance creation mechanism, the authors of SMOTE focused on feature space, rather than data space, to make the algorithm appropriate for a general imbalance problem [5]. Therefore, the synthetic instances are created along the line segments joining selected class neighbors. However, the pseudo-code of the algorithm presented in the same paper does not coincide with the described mechanism. While the pseudo-code incorporates the approach described by Eq. (1) for the synthetic instance creation, the approach compatible with the original description is

$$\boldsymbol{s}^i := \boldsymbol{x} + \boldsymbol{U}_i(0,1) \odot (\hat{\boldsymbol{x}}^{r(i)} - \boldsymbol{x}), \quad i = 1, \ldots, q, \quad (2)$$

where $U_i(0,1)$ is a n-dimensional vector of uniform random variables in (0,1), and $\odot$ denotes the Hadamard product. In the first case, new data is created on a line segment connecting the two minority instances. In the second case, new data is created within the bounding rectangle described by the two minority instances.

Furthermore, comparing the original paper [5] with the review paper of the same authors [6], a twofold interpretation of the procedure for the initial selection of minority instances is noticeable. While in the former interpretation, each minority instance is used for oversampling, in the latter, they are selected at random. Based on the observations mentioned above, a combination of different interpretations of instance selection and synthetic creation procedures can produce four different versions of the SMOTE algorithm. Accordingly, the performance evaluation of different SMOTE-based algorithms can be reinforced with respect to the type of these two procedures.

2.2. EXTENSIONS TO SMOTE

In practice, SMOTE is renowned as the benchmark in the learning from imbalance datasets [6]. Despite its simplicity and efficiency, its shortcomings are evident in certain problems, especially when class imbalance coincides with various data intrinsic characteristics. Considering only minority instances and neglecting the majority ones can lead to increased overlap between classes [12] or to overgeneralization [13]. Furthermore, its selection method can also extend noise regions, by oversampling noisy examples, which eventually increases the possibility of overfitting [14]. Various extensions to SMOTE have been proposed to tackle the aforementioned issues, by integrating miscellaneous filtering, interpolation, undersampling or dimensionality change techniques. A more extensive overview of SMOTE-based variants can be found in [6].

However, by incorporating these diverse techniques within a simple original algorithm, it becomes increasingly burdensome to justify the obtained performance and clarify the origin of newly created synthetic instances. Given that most of the algorithms consist of initial selection and interpolation procedures, the algorithms selected for the analysis consist solely of certain combinations of those, so as not to impair the simplicity and efficiency of the original SMOTE algorithm. Accordingly, representative algorithms were selected for three different interpolation mechanisms defined in [6]: SMOTE-like, range restricted and multiple.

Given the influence of the SMOTE algorithm in the overall problem of learning from imbalanced data, many of its extensions leave the original interpolation mechanism intact and alter other procedures. Such a group of algorithms can be said to perform SMOTE-like interpolation. MSMOTE [15] is one of the earliest algorithms in this group, which does not introduce additional mechanisms but merely changes the initial selection procedure. The MSMOTE algorithm categorizes the minority instances into three groups (security, border and latent noise), and performs different neighborhood selection strategy for each group. Unlike noise samples, an increase in the number of security samples can improve the performance of a classifier. Those samples that are difficult to classify are referred to as boundary samples. One of the more recent algorithms that has also preserved the interpolation mechanism of the original algorithm is Weighted-SMOTE [16], which calculates a weight matrix for determining the amount of synthetic data to be created for each minority instance. The weights are determined using the Euclidean distance of a given minority instance with respect to all remaining instances of the same class. The smaller the distance, the greater the amount of synthetic instances is generated for that minority data sample.

The interpolation mechanisms may be range restricted, with the structure of a close neighborhood affecting the location of a newly created synthetic instance. One of the earliest and well-established SMOTE-based extensions to include this type of interpolation is Borderline-SMOTE [17], which only strengthens the borderline minority examples. Examples at the borderline and those in the vicinity are more prone to misclassification than those far from the border and are therefore more important for classification. Such examples are found by analyzing the amount of minority and majority instances in the neighborhood of each minority example. The idea established in Borderline-SMOTE was broaden and refined for Safe-Level-SMOTE [13], which places synthetic instances closer to the largest safe level. The safe level is determined with a ratio between numbers of positive neighbors for the minority instance and its nearest neighbor. In contrast, SMOTE and Borderline-SMOTE can generate synthetic instances in unbefitting locations, such as overlapping regions and noise regions. One of the latest approaches that suggest a range restricted interpolation procedure is SMOTE-D (Deterministic version of SMOTE) [18], which generates synthetic examples by dividing the difference in attributes between two instances by the number of instances to be generated between them. Moreover, the greater the dispersion of distances between the minority instance and its class-neighbors, the more synthetic instances will be created.

When creating new synthetic examples, some SMOTE-based extensions perform multiple interpolations by calculating a combination of attributes from multiple neighbors. The earliest such approach is Distance-SMOTE [19], in which the new synthetic instances lie between minority instances and the mean example from their neighborhood. To create the new synthetic instance the difference between the minority example and the mean instance is multiplied by a random number between 0 and 1, to select a random point. Further on, the Random-SMOTE algorithm [20] creates synthetic instances inside the triangle derived from the minority examples and two random instances in the minority-class space. This idea was introduced to better address sparse regions among minority examples. A somewhat similar interpolation approach was utilized in the SNOCC [21] oversampling technique, but the synthetic instance is calculated as a convex combination of two randomly selected minority instances. This technique aims to generate instances that naturally model the distribution of original samples.

In the end, a multitude of SMOTE-based extensions can be found in the literature. When it comes to interpolation mechanisms, popular approaches that prevail either perform clustering of the minority examples, use Gaussian disturbances to generate the data from a single point or follow topologies based on geometric shapes, Voronoi diagrams, and graphs.

3. EXPERIMENTAL ANALYSIS

As stated at the outset, the two goals of this paper are to compare the overall performance of known oversampling algorithms and to determine their impact on the predictions of each class separately. Algorithms consisting only of initial selection and interpolation procedures were selected in order to maintain a simplicity of the original algorithm while investigating combinations of these mechanisms for better performance. The selected algorithms are shown in Table 1. In addition to the displayed SMOTE-based extensions, a simple random oversampling algorithm is also included in the analysis to serve as a point of reference. The algorithms are grouped by the interpolation mechanisms they implement.

The experiment setup is similar to that proposed in [7], in terms of used datasets and performance evaluation method, but different metrics are compared and different classifiers are considered. The datasets were obtained from the Keel [22] and the UCI [23] machine learning repositories and represent real-world problems with various imbalanced ratios to support the comparison. The characteristics of the selected datasets are summarized in Table 2. Oversampling techniques are evaluated in diverse classification settings because some classifiers are more equal than others. In order to observe which oversampling algorithms operate best with different types of classifiers, we selected the *k*-nearest neighbors with $k = 5$ (5-NN), multilayer perceptron (MLP), decision tree (DT) and support vector machine (SVM) algorithms. These classifiers are often used in the literature to compare oversampling techniques [7, 8]. The classifiers were implemented using Scikit-learn library [24], leaving default settings for each, to avoid the impact of manually selected parameters. Performance evaluation was conducted using stratified holdout testing. The datasets were split into the standard 0.75:0.25 ratio for training and testing, respectively. Each oversampler and classifier combination was evaluated simultaneously on the same split. This was repeated 30 times, each with a new split. The neighborhood size for all oversampling algorithms was set to the default and widely used value $k = 5$ [5, 17]. Regarding the number of synthetic instances created, a preliminary parameter study was conducted to find the value of q that yields the best performance for each combination of the dataset, classifier, and oversampling algorithm.

3.1. PERFORMANCE METRICS FOR IMBALANCED PROBLEMS

The notion of metrics is particularly important in class imbalance problems. Accuracy is a fundamental metric for evaluating classification performance, but it is not suitable for evaluating imbalanced datasets since it places more weight on the majority class, making it difficult to assess the classifier performance on the minority class [25]. In imbalanced problems, the performance evaluation metrics must consider class distribution [26]. Metrics such as the number of true positives (TP), the number of true positives (TN), the number of false positives (FP) and the number of false negatives (FN), can demonstrate the classification performance on each class separately. Considering the importance of achieving quality performance for all classes in imbalanced problems, these metrics are combined into more comprehensive metrics such as AUC [27], G-mean [28] and F-measure [29]. The latter has been widely used in the evaluation of oversampling algorithms in the literature [3, 7, 30], as a balanced F_β-score with a β value of 1

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \quad (3)$$

The balanced F_1-score demonstrates the trade-off between the precision and recall regarding the positive class, and it ranges between 0 and 1. The trade-off indicates whether a classifier obtains high recall by sacrificing precision or conversely by giving the classifier a low score. The F_1-score may be a good indicator of the algorithm's overall performance, but like other such measures, it manually weighs the importance of accuracy on a per-class basis. Performance metrics proposed specifically for imbalanced problems can be found in the literature, such as the Adjusted F-measure (AGF) [31], which places greater emphasis on the minority class, assuming that the classifier efficiency should be evaluated mostly on it.

Table 1. The oversampling algorithms being compared

Interpolation mechanism	Algorithm name	Algorithm label	Initial selection description
Oversampling with replacement	Random oversampling	RO	Minority instances are chosen at random [10].
SMOTE-like interpolation	Synthetic minority oversampling technique	SMOTE	Each minority instance is considered for synthetic creation procedure [5].
	Modified-SMOTE	MSMOTE	Minority instances for synthetic creation procedure are selected from three groups: latent, security, border [15].
	Weighted-SMOTE	WSMOTE	Each minority instance is assigned a weight that determines its contribution in synthetic creation procedure [16].
Range restricted	Borderline-SMOTE	BSMOTE	Only minority instances from danger zone are chosen [17].
	Safe-Level-SMOTE	SLSMOTE	Only minority instances with positive safe level are chosen [13].
	Deterministic Version of SMOTE	SMOTED	Minority instances are deterministically chosen based on their deviation of distances [18].
Multiple interpolations	Distance-SMOTE	DSMOTE	Each minority instance is considered for synthetic creation procedure [19].
	Random-SMOTE	RSMOTE	Each minority instance is considered for synthetic creation procedure [20].
	Sigma Nearest Oversampling based on Convex Combination	SNOCC	Minority instances are chosen at random [21].

To avoid dictating a more dominant class in a single metric, this paper proposes a per-class performance evaluation procedure using two metrics, specificity and sensitivity. The specificity (or true negative rate) metric

$$TNR = \frac{TN}{TN+FP} \ , \tag{4}$$

can be interpreted as the proportion of actual negatives that are correctly identified as such. On the other hand, the sensitivity (or true positive rate) metric

$$TPR = \frac{TP}{TP+FN} \ , \tag{5}$$

can be read as the proportion of actual positives that are correctly identified as such. In Eq. (4), FP denotes the number of false positives, while in Eq. (5), FN denotes the number of false negatives.

Table 2. Characteristics of the datasets used

Name	Label	Source	#inst.	#feat.	IR
Climate	D1	UCI	540	18	10.74
Ecoli3	D2	Keel	336	7	8.6
Ionosphere	D3	UCI	351	34	1.79
Cleveland0v4	D4	Keel	173	13	12.31
Vehicle1	D5	Keel	846	18	2.9
Yeast2v4	D6	Keel	514	8	9.08
Abalone9v21	D7	Keel	731	7	16.4
Dermatology6	D8	Keel	358	34	16.9
Glass6	D9	Keel	214	9	6.38
LEDdigitAllv1	D10	Keel	443	7	10.97
Poker9v7	D11	Keel	244	10	29.5
Shuttle6v23	D12	Keel	230	9	22
Vehicle3	D13	Keel	846	18	2.99
Vowel0	D14	Keel	988	13	9.98
WineQualityRed4	D15	Keel	1599	11	29.17
WineQualityRed8v6	D16	Keel	656	11	35.44
Yeast1v7	D17	Keel	459	7	14.3
Parkinsons	D18	UCI	195	22	3.06
Relax	D19	UCI	182	12	2.5
Transfusion	D20	UCI	748	4	3.2

4. RESULTS AND DISCUSSION

The experimental analysis is divided into three parts. The first part deals with the comparison of the selected oversamplers in terms of the overall performance represented by the obtained F_1-scores. The second part is concerned with the performance on the per-class basis. In the third part, both overall and per-class performance is evaluated at the level of interpolation mechanisms of the selected algorithms.

4.1. OVERALL CLASSIFICATION PERFORMANCE EVALUATION

To determine the best performing oversampling algorithms in terms of the overall performance, distances between attained F_1-scores and the perfect classifier score were calculated for each combination of oversampling and classifier algorithms. The performance of each such combination was constituted as a point in 20-dimensional space given that it is the number of datasets used, and each coordinate of the point is represented with the F_1-score obtained on a particular dataset. The Euclidean distance was used to order the constructed performances and the Chebyshev distance was applied to break ties, as in [7].

These distances are reported in Table 3 for each combination of oversampling and classifier algorithms. It should be noted that the label NO represents the case without oversampling and serves as a basis for comparison. Moreover, the smallest distances for each combination are highlighted.

At first glance, it can be noticed that random oversampling was twice the closest to the perfect classifier, for MLP and SVM classifiers. In addition, the performance achieved is considerably different with respect to the classifier used than with the oversampler used. Furthermore, the performance of DT classifier is practically unsusceptible to the oversampling preprocessing step, whereas other algorithms show significant gains in performance when oversampling is applied. This outcome is consistent with previous works [32], which show that oversampling does not affect the DT algorithm.

Table 3. Distance of obtained F_1-scores from the perfect classifier for selected oversamplers across different classifiers

Oversampler	5-NN	MLP	DT	SVM
NO	1.52 (0.55)	1.74 (0.58)	1.46 (0.54)	2.22 (0.58)
RO	1.38 (0.54)	**1.35 (0.57)**	1.44 (0.50)	**1.55 (0.58)**
SMOTE	1.35 (0.53)	1.40 (0.58)	1.45 (0.53)	1.61 (0.58)
MSMOTE	1.36 (0.53)	1.40 (0.59)	1.45 (0.54)	1.62 (0.58)
WSMOTE	1.35 (0.54)	1.40 (0.57)	1.44 (0.53)	1.62 (0.58)
BSMOTE	1.35 (0.53)	1.38 (0.57)	1.45 (0.52)	1.75 (0.58)
SLSMOTE	1.36 (0.55)	1.40 (0.58)	1.46 (0.52)	1.66 (0.58)
SMOTED	**1.34 (0.52)**	1.43 (0.59)	1.45 (0.54)	1.82 (0.58)
DSMOTE	1.37 (0.55)	1.40 (0.57)	1.46 (0.54)	1.66 (0.58)
RSMOTE	1.35 (0.54)	1.41 (0.57)	**1.43 (0.52)**	1.66 (0.58)
SNOCC	1.37 (0.54)	1.45 (0.57)	1.46 (0.55)	1.63 (0.58)

For the sake of better validation and reliability, statistical tests were conducted to further validate the results. To gain insight into the statistical differences between oversampling algorithms within these three classifiers, average rankings of the Friedman test for multiple comparisons were determined for each such combination. The rankings are shown in Table 4, where smaller values indicate better performing algorithms. It is noticeable that all oversamplers considerably outperform NO. Random oversampling manifested as the weakest oversampler for the KNN classifier, but its total ranking is surprisingly ahead of the ranks of some SMOTE-based extensions. SNOCC and SLSMOTE stand out as the worst oversamplers, independent of the classifier. The best ranked oversampler in total is WSMOTE, although it was not the best for any particular classifier but was consistently among the best. In fact, WSMOTE has already manifested as preeminent in terms of overall performance in similar comparisons [7]. DSMOTE has twice been the best-performing oversampler, for MLP and SVM classifiers, and follows WSMOTE in ranking as well as SMOTE and RSMOTE. It is interesting to note that the original SMOTE algorithm outperforms as many as six of its extensions in overall performance, i.e. all but WSMOTE and DSMOTE.

Table 4. Average rankings of the Friedman test for obtained F_1-scores of different oversamplers across different classifiers

Oversampler	5-NN	MLP	SVM	Total
NO	9.475	9.85	9.475	28.800
RO	7.400	4.975	5.275	17.650
SMOTE	5.275	5.200	5.400	15.875
MSMOTE	6.100	6.025	5.800	17.925
WSMOTE	4.425	5.100	5.400	**14.925**
BSMOTE	5.725	5.900	5.725	17.350
SLSMOTE	6.525	5.875	6.800	19.200
SMOTED	**4.175**	5.725	6.650	16.550
DSMOTE	5.425	**4.825**	**5.250**	15.500
RSMOTE	4.875	5.775	5.275	15.925
SNOCC	6.600	6.750	6.025	19.375

4.2. PER-CLASS PERFORMANCE EVALUATION

While it is of substantial importance to compare the overall performance of the selected oversampling algorithms, it is also interesting to study how they affect the prediction of each class individually. To this end, Euclidean distances from the perfect classifier were again obtained for two metrics: specificity and sensitivity. Table 5 shows these distances, with the first number representing the distance of the specificity values, and the number in parentheses the distance of the sensitivity values. It can be concluded that the prediction for the majority class is most accurate when no oversampling is performed. The DT classifier again proves immune to oversampling, as there are no major differences in either of these two metrics, so it is excluded from further analysis. On the other hand, the use of oversampling with the SVM classifier alters both metrics the most.

Table 5. Distance of specificity and sensitivity scores from the perfect classifier for selected oversamplers across different classifiers

Oversampler	5-NN	MLP	DT	SVM
NO	**0.27** (2.97)	**0.11** (3.41)	0.55 (2.46)	0.01 (4.17)
RO	0.69 (2.02)	0.51 **(2.13)**	**0.53** (2.44)	0.45 **(2.76)**
SMOTE	0.69 (2.05)	0.51 (2.37)	0.61 (2.30)	0.50 (2.83)
MSMOTE	0.68 (2.21)	0.52 (2.41)	0.61 (2.33)	0.51 (2.88)
WSMOTE	0.71 (2.01)	0.53 (2.33)	0.60 (2.29)	0.57 (2.79)
BSMOTE	0.70 (1.99)	0.47 (2.35)	0.59 (2.36)	0.36 (3.18)
SLSMOTE	0.73 (2.01)	0.53 (2.28)	0.61 (2.32)	0.42 (3.09)
SMOTED	0.56 (2.15)	0.33 (2.54)	0.59 (2.33)	0.30 (3.40)
DSMOTE	0.78 (1.93)	0.51 (2.37)	0.60 (2.35)	0.55 (2.95)
RSMOTE	0.78 **(1.92)**	0.53 (2.38)	0.63 **(2.24)**	0.56 (2.96)
SNOCC	0.71 (2.14)	0.52 (2.58)	0.61 (2.34)	0.49 (2.95)

As in the overall performance comparison, average rankings of Friedman test were derived to rank the selected oversampling algorithms according to the attained specificity and sensitivity scores. These ranks are shown in Table 6, with the first number representing the rank in terms of attained specificity values and the number in parentheses rank in terms of attained sensitivity values. It is obvious that the employment of oversampling greatly impairs the prediction accuracy on the majority class, i.e. the specificity value. The SMOTED algorithm least disrupts the performance on the majority class, and according to the utilized Nemenyi's, Holm's and Shaffer's post-hoc procedures, most often statistically outperforms other algorithms. The performance of other oversamplers is similar when it comes to majority-class prediction. When it comes to ranks of sensitivity scores, it is evident that the use of oversampling considerably contributes to the success of minority-class prediction. WSMOTE, RSMOTE and DSMOTE stand out as oversampling algorithms that best contribute to the performance of a classifier on the minority class. However, these algorithms are ranked the worst by specificity scores, which suggests that their success on the minority class results from sacrificing performance on the majority class.

Table 6. Average rankings of the Friedman test for obtained specificity and sensitivity scores of different oversamplers across different classifiers

Oversampler	5-NN	MLP	SVM	Total
NO	**1.475** (10.750)	**1.675** (10.750)	**2.525** (9.800)	**5.675** (31.300)
RO	7.475 (6.225)	6.425 (4.900)	6.500 (5.150)	20.400 (16.305)
SMOTE	5.900 (5.725)	6.900 (5.125)	6.350 (5.325)	19.150 (16.175)
MSMOTE	5.375 (6.450)	6.425 (5.575)	6.300 (5.875)	18.100 (17.900)
WSMOTE	6.350 (4.850)	6.800 **(4.050)**	8.000 **(4.200)**	21.150 **(13.100)**
BSMOTE	6.550 (5.175)	5.400 (6.200)	5.450 (6.225)	17.400 (17.600)
SLSMOTE	6.900 (5.400)	7.775 (4.900)	5.500 (7.325)	20.175 (17.625)
SMOTED	4.650 (6.825)	4.000 (7.775)	4.350 (7.775)	13.000 (22.375)
DSMOTE	7.425 (4.400)	7.125 (4.850)	7.425 (4.525)	21.975 (13.775)
RSMOTE	7.025 **(4.075)**	7.500 (5.325)	6.850 (4.375)	21.375 (13.775)
SNOCC	6.875 (6.125)	5.975 (6.550)	6.750 (5.425)	19.600 (18.100)

In various problems of class imbalance, the success of majority class prediction can be almost as important as the success of minority class prediction. Therefore, the question is whether some oversamplers achieve better performance on both classes on a particular problem. In order to discover such oversamplers on given datasets, non-dominated sorting for multi-objective problems [33] was applied. In this case, the specificity and sensitivity scores represented the objectives to be maximized and the solutions were oversampling algorithms being compared. For each combination of dataset and classifier, sorting determines non-dominated oversamplers, i.e. oversamplers that no other oversampling algorithm outperforms in both metrics. Given the extensiveness of displaying these results for each dataset, Table 7 shows how many times each oversampler appeared as a non-dominated solution for each classifier.

The oversampling algorithm that most often turned out to be a non-dominated solution is SMOTED and this applies for all three classifiers. This can be attributed to the fact that it achieved the best performance on the minority class, according to the Friedman rankings presented in Table 6. Surprisingly, the MSMOTE algorithm immediately follows SMOTED, though it ranks among the worse algorithms when comparing overall performance, and sensitivity and specificity scores. On the contrary, original SMOTE algorithm appeared least as a non-dominated solution, although it was among the better oversamplers in terms of overall performance. The best performing algorithms overall, WSMOTE, DSMOTE and RSMOTE did not outperform its competition for any classifier. Due to their poor performance on the majority class, they fail to dominate other algorithms, despite their superiority on the minority class. Finally, the SNOCC algorithm is one of the worst performing algorithms in this comparison as well, which raises a question of its improvement over the original SMOTE algorithm.

Table 7. The number of occurrences as non-dominated solution for each oversampler across different classifiers

Oversampler	5-NN	MLP	SVM	Total
RO	5	12	**15**	32
SMOTE	8	7	10	25
MSMOTE	12	9	11	39
WSMOTE	10	11	9	30
BSMOTE	10	13	13	36
SLSMOTE	11	11	11	33
SMOTED	**16**	**15**	**15**	**46**
DSMOTE	10	13	11	34
RSMOTE	12	7	14	33
SNOCC	6	8	12	26

4.3. PERFORMANCE EVALUATION OF INTERPOLATION MECHANISMS

The interpolation procedure is the main novelty introduced by the SMOTE algorithm, while extensions to the original algorithm are mainly concerned with the improvement of such procedures. In order to compare groups of different interpolation mechanisms, the results of individual algorithms are aggregated, and the performance of a group is represented by the average

performance of the belonging algorithms. Table 8 presents average rankings of the Friedman test according to the obtained F_1-scores of each combination of interpolation mechanism and classifier. In this way, the overall performance of the interpolation mechanisms can be compared.

The SMOTE-like interpolation, represented by the SMOTE, MSMOTE, and WSMOTE algorithms, achieved the best rank for KNN and MLP classifiers, as well as the best overall rank. On the other hand, range restricted interpolation is the worst-ranked interpolation mechanism because poor performance of BSMOTE and SLSMOTE algorithms impairs the respectable performance of the SMOTED algorithm. Furthermore, this interpolation mechanism is the worst ranked for both MLP and SVM classifiers but performed as second best for KNN classifier. The opposite is true for oversampling with replacement, which performed especially well for the SVM algorithm. The multiple interpolation mechanism maintains its rank almost independently of the classifier, being the second-worst mechanism in each case.

Table 8. Average ranking of the Friedman test for derived F_1-scores of different interpolation mechanisms across different classifiers

Interpolation mechanism	5-NN	MLP	SVM	Total
Oversampling with replacement	3.15	2.175	**2**	7.325
SMOTE-like interpolation	**2.05**	**2.15**	2.35	**6.55**
Range restricted	2.15	3.075	2.95	8.175
Multiple interpolations	2.65	2.6	2.7	7.95

However, the differences in ranks from Friedman test are unsubstantial, so the Wilcoxon test [34] was conducted to further explore the statistical differences. The obtained ranks computed by the Wilcoxon test are presented in Table 9. Statistically significant differences in ranks are denoted with ⊕ when the method in the row improves upon the one in the column, and with ⊙ when the method in the column improves upon one in the row. In this regard, the upper diagonal considers a significance level of 0.9, and the lower diagonal a level of 0.95. Table 9 shows that, for KNN classifier, both SMOTE-like and range restricted interpolation mechanisms outperform the oversampling with replacement in a statistically significant manner. For MLP and SVM classifiers, the range restricted mechanism was significantly outperformed by oversampling with replacement and SMOTE-like mechanisms, respectively. Multiple interpolations mechanism at no time outperformed the competition. Given that it is implemented by two algorithms that are among the best in the previous comparisons, DSMOTE and RSMOTE, this mediocre result is probably due to the ineffectiveness of the SNOCC algorithm.

As with the comparison of the per-class performance of individual algorithms, non-dominated sorting was again conducted to determine non-dominated solutions, which are represented by interpolation mechanisms. Specificity and sensitivity values are aggregated for each interpolation mechanism which is represented by their average scores. Table 10 shows how many times each interpolation mechanism appeared as a non-dominated solution for each classifier. These numbers are in line with the ranks shown previously. The SMOTE-like interpolation mechanism is most often the dominant solution regardless of the classifier, apart from oversampling with replacement. This only adds significance to the original algorithm and emphasizes the complexity of the task of improving upon. Although not far behind in the overall performance, other mechanisms fail to improve the original, which is the main goal of such ideas. Naturally, it should be noted that some algorithms from worse-performing interpolation mechanisms outperform those of other groups, but this can be attributed to their initial selection procedures.

Table 9. Ranks computed by the Wilcoxon test for the utilized classifiers using different interpolation mechanisms

5-NN				
Interpolation mechanism	Oversampling with replacement	SMOTE-like interpolation	Range restricted	Multiple interpolations
Oversampling with replacement	—	54.0 ⊙	55.0 ⊙	65.0
SMOTE-like interpolation	156.0	—	99.0	117.0
Range restricted	155.0	91.0	—	115.0
Multiple interpolations	145.0	73.0	75.0	—
MLP				
Interpolation mechanism	Oversampling with replacement	SMOTE-like interpolation	Range restricted	Multiple interpolations
Oversampling with replacement	—	111.0	145.0 ⊕	122.0
SMOTE-like interpolation	79.0	—	146.0	127.0
Range restricted	45.0 ⊙	64.0	—	95.0
Multiple interpolations	68.0	63.0	115.0	—
SVM				
Interpolation mechanism	Oversampling with replacement	SMOTE-like interpolation	Range restricted	Multiple interpolations
Oversampling with replacement	—	122.5	172.5 ⊕	130.5
SMOTE-like interpolation	87.5	—	140.5 ⊕	114.5
Range restricted	37.5 ⊙	49.5	—	69.5
Multiple interpolations	79.5	75.5	120.5	—

Table 10. The number of occurrences as a non-dominated solution for each interpolation mechanism across different classifiers

Interpolation mechanism	5-NN	MLP	SVM	Total
Oversampling with replacement	9	18	17	44
SMOTE-like interpolation	18	16	15	49
Range restricted	17	15	13	45
Multiple interpolations	15	14	13	42

5. CONCLUSION

The significance and impact of the SMOTE algorithm are clear in the literature on handling the problem of class imbalance. Despite its efficiency, the algorithm is straightforward and consists only of initial selection and interpolation procedures. The primary objective of this study was to conduct an in-depth comparison of several oversampling techniques that only consist of the two procedures of the original algorithm. In doing so, comparisons were made in terms of overall and per-class performances.

The experimental results indicate that the performance of these algorithms should not be evaluated through a single metric. The Weighted-SMOTE algorithm achieved the best overall performance, and is closely followed by Distance-SMOTE, original SMOTE, and Random-SMOTE. However, these oversamplers have been shown to most impair the accuracy of the majority class prediction. In addition, when looking at the overall performance alone, as many as six SMOTE-based extensions fail to outperform the original algorithm. By conducting a per-class comparison, it is possible for practitioners to select oversampling algorithms depending on the efficiency on more important class. The deterministic version of SMOTE dominated its competition by performance in both classes. Interestingly, the most dominant algorithms by the criteria of both specificity and sensitivity metrics are not the most successful in terms of overall performance. Lastly, the recently proposed oversampling approach, Sigma Nearest Oversampling based on Convex Combination (SNOCC), performed the worst in all observed aspects.

The performance evaluation per interpolation mechanisms confirmed the discoveries of some previous works in terms of the poor performance of well-established SMOTE-based variants, Borderline-SMOTE and Safe-Level-SMOTE. The interpolation mechanism of the original algorithm proved to be the best in both overall and per-class performance, which raises the question of whether the some SMOTE-based extensions introduce valid improvements or suggest modifications for the sake of publication. For future work, other mechanisms of SMOTE-based extensions can also be compared, starting with initial selection. In addition, it would be beneficial to compare other interpolation mechanisms that are not considered in this paper, such as clustering.

Finally, this performance study of oversampling algorithms has resulted in several interesting findings. In the future, it should be extended to more algorithms proposed for dealing with the problems of class imbalance. Also, it would be beneficial to include undersampling algorithms in the study for a more comprehensive evaluation of data-level approaches. In that regard, the issue of performance metrics for imbalanced problems needs to be addressed more profoundly. Hopefully, this paper clearly points to the need for more extensive comparisons of oversampling algorithms and their evaluation of their impact on both the minority and majority classes.

6. ACKNOWLEDGMENTS

The current archival periodical article is based on the conference presentation [7]. This research has been supported by the European Regional Development Fund under the grant KK.01.2.1.01.0127. The authors would like to thank Dražen Bajer and Bruno Zorić for constructive remarks on the manuscript.

7. REFERENCES:

[1] N. Japkowicz, M. Shah, "Evaluating Learning Algorithms: A Classification Perspective", 1st Ed., Cambridge University Press, 2011.

[2] S. Fotouhi, S. Asadi, M. W. Kattan, "A comprehensive data level analysis for cancer diagnosis on imbalanced data", Journal of Biomedical Information, Vol. 90, 2019, pp. 103089.

[3] A. Ali, S. M. Shamsuddin, A. L. Ralescu, "Classification with class imbalance problem: a review", International Journal of Advances in Soft Computing and its Applications, Vol. 7, No. 3, 2015, pp. 176-204.

[4] P. Skryjomski, B. Krawczyk, "Influence of minority class instance types on SMOTE imbalanced data oversampling", Proceedings of the 1st International Workshop on Learning with Imbalanced Domains: Theory and Applications, Skopje, Macedonia, 22 September 2017, Vol. 74, pp. 7-21.

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique", Journal of Artificial Intelligence Research, Vol. 16, 2002, pp. 321-357.

[6] A. Fernández, S. García, F. Herrera, N. V. Chawla, "SMOTE for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary", Journal of Artificial Intelligence Research, Vol. 61, 2018, pp. 863-905.

[7] D. Bajer, B. Zorić, M. Dudjak, G. Martinović, "Performance Analysis of SMOTE-based Oversampling Techniques When Dealing with Data Imbalance", Proceedings of the 26th International Conference on Systems, Signals and Image Processing, Osijek, Croatia, 5-7 June 2019, pp. 265-271.

[8] G. Kovács, "An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets", Applied Soft Computing, Vol. 83, 2019, pp. 105662.

[9] M. Kubat, S. Matwin, "Addressing the curse of imbalanced training sets: one-sided selection", Proceedings of the 14th International Conference on Machine Learning, Nashville, Tennessee, USA, 8-12 July 1997, pp. 179-186.

[10] C. X. Ling, C. Li, "Data mining for direct marketing: Problems and solutions", Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, New York, New York, USA, 27-31 August 1998, Vol. 98, pp. 73-379.

[11] G. E. Batista, R. C. Prati, M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data", ACM SIGKDD Explorations Newsletter, Vol. 6, No. 1, 2004, pp. 20-29.

[12] V. García, R. A. Mollineda, J. S. Sánchez, "On the k-NN performance in a challenging scenario of imbalance and overlapping", Pattern Analysis and Applications, Vol. 11, No. 3-4, 2008, pp. 269-280.

[13] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem", Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Bangkok, Thailand, 27-30 April 2009, pp. 475-482.

[14] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, A. Folleco, "An empirical study of the classification performance of learners on imbalanced and noisy software quality data", Information Sciences, Vol. 259, 2014, pp. 571-595.

[15] S. Hu, Y. Liang, L. Ma, Y. He, "MSMOTE: improving classification performance when training data is imbalanced", Proceedings of the 2nd International Workshop on Computer Science and Engineering, Qingdao, China, 28-30 October 2009, Vol. 2, pp. 13-17.

[16] M. R. Prusty, T. Jayanthi, K. Velusamy, "Weighted-SMOTE: A modification to SMOTE for event classification in sodium cooled fast reactors", Progress in Nuclear Energy, Vol. 100, 2017, pp. 355-364.

[17] H. Han, W. Y. Wang, B. H. Mao, "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning", Proceedings of the International Conference on Intelligent Computing, Hefei, China, 23-26 August 2005, pp. 878-887.

[18] F. R. Torres, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, "SMOTE-D a deterministic version of SMOTE", Proceedings of the 8th Mexican Conference on Pattern Recognition, Guanajuato, Mexico, 22-25 June 2016, pp. 177-188.

[19] J. De La Calleja, O. Fuentes, "A Distance-Based Over-Sampling Method for Learning from Imbalanced Data Sets", Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference, 7-9 May 2007, pp. 634-635.

[20] Y. Dong, X. Wang, "A new over-sampling approach: random-SMOTE for learning from imbalanced data sets", Proceedings of the International Conference on Knowledge Science, Engineering and Management, Dalian, China, 10-12 August 2011, pp. 343-352.

[21] Z. Zheng, Y. Cai, Y. Li, "Oversampling method for imbalanced classification", Computing and Informatics, Vol. 34, No. 5, 2016, pp. 1017-1037.

[22] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, "Kernel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework", Journal of Multiple-Valued Logic & Soft Computing, Vol. 17, No. 11, 2011, pp. 255-287.

[23] K. Bache, M. Lichman, "UCI machine learning repository", 2013, http://archive.ics.uci.edu/ml (accessed: 2019).

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, "Scikit-learn: Machine learning in Python", Journal of Machine Learning Research, Vol. 12, 2011, pp. 2825-2830.

[25] X. Guo, Y. Yin, C. Dong, G. Yang, G. Zhou, "On the class imbalance problem", Proceedings of the 4th

International Conference on Natural Computation, Jinan, China, 25-27 August 2007, Vol. 4, pp. 192-201.

[26] V. López., A. Fernández, S. García, V. Palade, F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics", Information Sciences, Vol. 250, 2013, pp. 113-141.

[27] J. Huang, C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms", IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 3, 2005, pp. 299-310.

[28] R. Barandela, J. S. Sánchez, V. Garca, E. Rangel, "Strategies for learning in class imbalance problems", Pattern Recognition, Vol. 36, No. 3, 2003, pp. 849-851.

[29] R. Baeza-Yates, B. Riberio-Neto, "Modern information retrieval", 1st Ed., Addison Wesley, 1999.

[30] M. S. Santos, J. P. Soares, P. H. Abreu, H. Araujo, J. Santos, "Cross-validation for imbalanced datasets: Avoiding overoptimistic and overfitting approaches", IEEE Computational Intelligence Magazine, Vol. 13, No. 4, 2018, pp. 59-76.

[31] A. Maratea, A. Petrosino, M. Manzo, "Adjusted F-measure and kernel scaling for imbalanced data learning", Information Sciences, Vol. 257, 2014, pp. 331-341.

[32] C. Drummond, R. C. Holte, "C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling", Proceedings of the Workshop on Learning from Imbalanced Data Sets, Washington, DC, USA, 21 August 2003, Vol. 11, pp. 1-8.

[33] M. Woodruff, J. Herman, "Nondominated sorting for multi-objective problems", 2018, https://github.com/matthewjwoodruff/pareto.py (accessed: 2019).

[34] J. Derrac, S. García, D. Molina, F. Herrea, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms", Swarm and Evolutionary Computation, Vol. 1, No. 1, 2011, pp. 3-18.