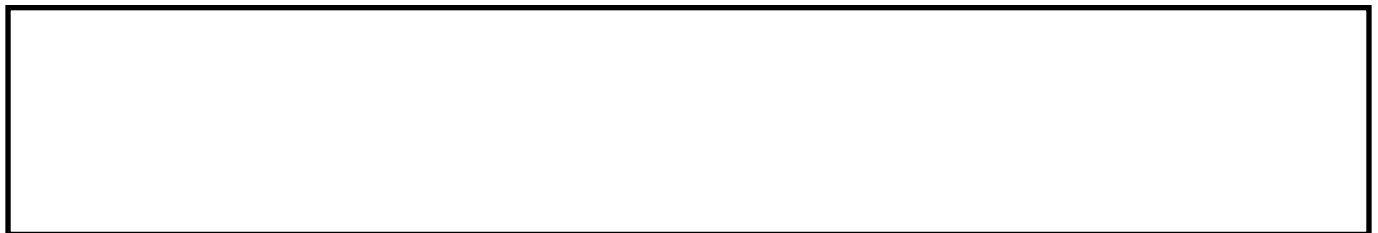


# CDSMOTE: class decomposition and synthetic minority class oversampling technique for imbalanced-data classification.

ELYAN, E., MORENO-GARCIA, C.F. and JAYNE, C.

2021





# CDSMOTE: class decomposition and synthetic minority class oversampling technique for imbalanced-data classification

Eyad Elyan<sup>1</sup> · Carlos Francisco Moreno-Garcia<sup>1</sup> · Chrisina Jayne<sup>2</sup>

Received: 17 June 2019 / Accepted: 16 June 2020 / Published online: 18 July 2020  
© The Author(s) 2020

## Abstract

Class-imbalanced datasets are common across several domains such as health, banking, security, and others. The dominance of majority class instances (negative class) often results in biased learning models, and therefore, classifying such datasets requires employing some methods to compact the problem. In this paper, we propose a new hybrid approach aiming at reducing the dominance of the majority class instances using class decomposition and increasing the minority class instances using an oversampling method. Unlike other undersampling methods, which suffer data loss, our method preserves the majority class instances, yet significantly reduces its dominance, resulting in a more balanced dataset and hence improving the results. A large-scale experiment using 60 public datasets was carried out to validate the proposed methods. The results across three standard evaluation metrics show the comparable and superior results with other common and state-of-the-art techniques.

**Keywords** Machine learning · Class-imbalance · Classification · Undersampling · Oversampling

## 1 Introduction

Class-imbalance classification is a long withstanding problem in the literature [1–5] where a binary dataset contains a disproportionately larger amount of samples of the majority class (i.e., negative class) [6]. Such datasets are common in many domains including life sciences [7], protein classification [8], DNA sequence recognition [9], financial sector [10], Medical domain [11], Medicine rating and recommendations [12], engineering drawings analysis [13–15] and others. An example of a binary classification problem is shown in Eq. 1. In a classification task, the aim is to learn a function  $h(x)$  that maps an instance  $\mathbf{x}_i \in X$  to a class  $\mathbf{y}_i$ , where  $y_i \in Y = \{C_N, C_P\}$ , denoting negative and positive class, respectively. In an imbalanced dataset, the positive class  $C_P$  (class of interest) is often underrepresented in the dataset, causing a learning algorithm to be biased toward the majority class instances  $C_N$ .

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & \vdots & \dots & x_{mn} \end{bmatrix}, \quad Y = \begin{bmatrix} y_1 \\ y_i \\ \vdots \\ y_m \end{bmatrix} \quad (1)$$

Consider the banking sector, a dataset for handling fraudulent transactions. Most transactions are legitimate (i.e., 90–99%,  $x_i \in C_N$ ), and few are fraud (class  $C_P$  in Eq. 1). In such scenario, an accuracy more than 90% can be easily obtained. However, it is easy to miss-classify the class of interest (i.e.,  $x_i \in C_P$ ), and hence, the need for different solutions accounts for the data distribution. Solutions for handling such a problem can be broadly categorized as data-based, algorithmic-based or cost-sensitive [2]. Data-based solutions are commonly used to compact class-imbalanced datasets. These methods are focused on either undersampling the data to reduce the dominance of the majority class instances, oversample the minority class, or a hybrid approach that combines both methods. Algorithmic-based solutions tend to modify the learning algorithms. Such algorithms include C4.5,  $k$  nearest neighbors (k-NN), support vector machine (SVM) and others.

Unlike most learning models which assign the same cost for all misclassifications in the learning process, cost-

✉ Eyad Elyan  
e.elyan@rgu.ac.uk

<sup>1</sup> Robert Gordon University, Aberdeen, UK

<sup>2</sup> Teesside University, Middlesbrough, UK

sensitive methods are based on the actual class and aiming at minimizing the total cost [16]. These methods emphasize the class of interest (positive class) by assigning higher costs for misclassifying it.

In this paper, we propose a new method for handling the class-imbalance problem based on class decomposition (CD) of the majority class and synthetic oversampling of the minority class. For short, we will refer to the proposed method as CDSMOTE. Our method is designed to first find the similarities within the majority class instances and group them accordingly. This results in reducing the dominance of the majority class without causing information loss, as it is the case with other undersampling techniques. To ensure a balanced distribution of the data, we then apply an oversampling method to improve the representation of the minority class. Extensive experiments were carried out, and the results show the superiority of the proposed method in improving different metrics when compared with the common and state-of-the-art methods.

The rest of the paper is organized as follows. Section 2 provides the necessary background and relevant literature. In Sect. 3, we present our method. Section 4 details the experiments with thorough evaluation and discussion of the results. Finally, Sect. 5 concludes the work and discusses possible future directions.

## 2 Related work

Handling the class-imbalance is most commonly achieved using either data-based [17, 18] or algorithmic-based solutions [19]. Because of the purpose and scope of this paper, we will focus on data-based methods. For those interested in algorithmic-based solutions, we refer the reader to a recent survey [20] for more details. Data resampling is one of the standard methods for handling class-imbalanced datasets classification. These methods include undersampling, which aims at reducing the dominance of the majority class instances, oversampling, aiming at increasing the visibility of the minority class instances or hybrid approaches (combining both methods).

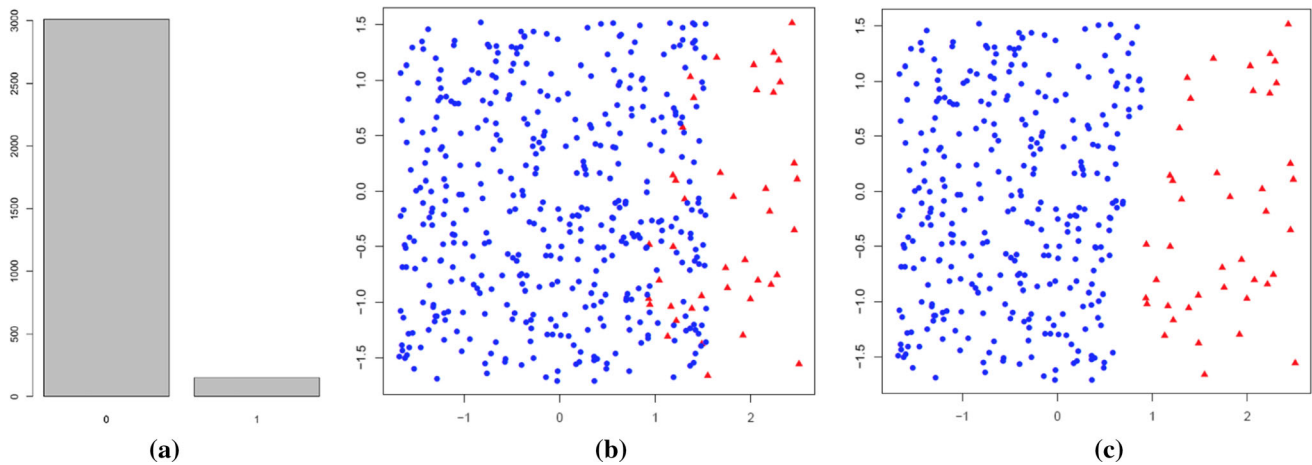
Random sampling is one of the most basic methods used to handle the class-imbalance problem. It can be applied as random undersampling (RUS), with the aim of rebalancing a dataset by randomly sampling a subset of the majority class instances, or as random oversampling (ROS), to multiply the instances of the minority class. This approach is simple, and thus, it is almost certain that it will result in losing data or overfitting. Therefore, these methods are rarely used alone. For example, in [21] a hybrid approach of RUS and a boosting algorithm (RUSBoost) was implemented to improve the classification results.

Other methods are focused on undersampling data from the overlapping region aiming at minimizing the overlap between positive and negative instances. Figure 1a shows a typical example of a hugely imbalanced dataset with the overlapping region highlighted in Fig. 1b, while Fig. 1c shows a possible solution where undersampling is carried out within that region. Several techniques are available to facilitate undersampling from the overlapping region. Among these, Tomek Link (T-Link) [22] which is a popular concept is originally proposed to edit the nearest neighbor rule which is used to remove instances in an overlapping region. The main idea is simple, given a dataset  $Z$ , two samples ( $a$  from the majority class and  $b$  from the minority class) and a distance function  $d$  between them, a T-Link is obtained if there is an example  $z \in Z$  such that:

$$\text{dist}(z, a) > \text{dist}(a, b) \wedge \text{dist}(z, b) > \text{dist}(a, b) \quad (2)$$

The basic idea then is to discard sample  $a$  from the dataset whenever a T-link is obtained. This method is proved to be useful in handling class-imbalance and provided a better alternative to random sampling. For example, Kubat and Matwin [23] proposed an undersampling method by shrinking the overlapping region using T-Link [22]. This was achieved by selectively removing redundant majority class instances close to the class boundary. Better performance was reported based on real datasets. Devi et al. [24] proposed a more recent method based on T-Link which also aimed at removing noise and redundant negative instances from the overlapping region. Other similar approaches include neighboring cleaning rule (NCL) for small sets [25] and the majority undersampling technique (MUTE) [26]. Removing negative class instances selectively (i.e., from the overlapping region) often yields to better results. However, this does not prevent data loss which might affect the overall accuracy. Therefore, in some scenarios or application domains where the overall accuracy matters, alternatives should be considered to minimize the risk of losing information [2]. A recent work presented in [6] by Vuttipittayamongkol and Elyan followed a similar approach to selectively remove the negative instances from the overlapping region by using fuzzy C-Means and reported the comparable results with the state of the art. More recently, the authors extended their work by proposing new methods for handling class-imbalance where unlike other common resampling methods, they introduced a novel way to detect and remove negative instances from the overlapping region using neighborhood searching techniques and reported the comparable results with state-of-the-art methods [27].

Oversampling methods aiming at improving the presence of the minority class instances are also common practice. Synthetic minority oversampling technique



**Fig. 1** Undersampling imbalanced datasets. **a** Imbalanced dataset, **b** overlapping region and **c** undersampling from overlapping region

(SMOTE) proposed by Chawla et al. [28] is still widely used in this domain. SMOTE is based on generating synthetic data points using a neighborhood-based technique (i.e.,  $k$ -NN). Several extensions of SMOTE have been proposed since its introduction, including SMOTEBoost [29], Borderline-SMOTE [17], DBSMOTE [30], MWMOTE [31] and others. ADASYN [32] is another common oversampling method that is widely used. This method is based on assigning a higher weight to *harder-to-learn* samples (samples in the overlapping region) using  $k$ -NN.

Clustering-based methods are common practice across different domains [33] and widely used for undersampling data. A clustering method such as  $k$ -means or fuzzy C-means (FC-means) [34] is applied to cluster the majority class instances into  $k$  clusters. Data are then sampled from each cluster aiming at having a smaller and yet representative sample. As a result, a more balanced dataset is obtained. Bunkhumpornpat et al. [35] proposed a majority class undersampling technique based on density-based spatial clustering algorithm (DBMUTE). DBMUTE was designed to eliminate negative instances from the overlapping region. Lin et al. [36] presented another clustering-based undersampling method where the negative instances were first clustered with the number of clusters set to equal the number of data points in the minority class. The undersampling was then carried out using cluster centers and clusters nearest neighbors, respectively. An experiment using 44 public datasets showed the competitive results.

Clustering-based methods were also used to handle minority classes in the dataset. For example, Yong et al. [37] used  $k$ -means to divide the minority class into smaller clusters, and genetic algorithm was then used to generate new samples based on those clusters. This technique, however, will not be applicable when the number of minority class instances is minimal. Similarly, Seoane

Santos et al. [38] handled patients data by clustering the minority class instances and then rebalanced the data using SMOTE. Puntumapon et al. [39] proposed a new method called TRIM, as a preprocessing stage before applying oversampling methods such as SMOTE or one of its extensions. Lim et al. [40] implemented an evolutionary ensemble learning framework by clustering the minority class instances using mini-batch  $k$ -means and hierarchical agglomerative clustering before generating synthetic samples.

Overall, it can be said that undersampling minority class instances contributed to improving the results before applying oversampling; however, such methods require enough samples from the minority class instances before they can be applied. More recently, Generative Adversarial Neural Networks (GANs) have been applied successfully to handle class-imbalance, by synthesizing new samples of the minority class's instances to handle the imbalance problem. A typical example was presented in [14, 41, 42] where a new data augmentation approach using variants of GANs to handle the class-imbalance problem was presented. Using image-based datasets, the methods showed favorable performance over other traditional sampling techniques.

### 3 Methods

The method presented in this paper is designed to first reduce the dominance of the majority class instances in the dataset by applying unsupervised learning algorithm to group it into subclasses. An oversampling technique is then used to improve the presence of the minority class instances in the dataset. Algorithm 1 provides a schematic overview of the proposed method, where for any dataset  $A$ , first, it is transformed into a decomposed dataset  $A_c$ .

(Sect. 3.1), followed by oversampling of the minority class instances (*oversample*) subject to reassessing the decomposed dataset  $A_c$  (Sect. 3.2). If oversampling is then applied, then a dataset  $A_{cm}$  is created which is a result of class decomposition and oversampling combined. Finally, a learning algorithm is applied to the resulting dataset.

**Algorithm 1** CDSMOTE overview

```

Result:  $Gmean, AUC, F_1$  Score
i = number of datasets
while (i > 0) do
   $A \leftarrow Dataset[i]$ ;
   $A_c \leftarrow decomposeSet(A, K)$ ;
  if (New minority in  $A_c$ ) then
     $A_{cm} \leftarrow oversample(A_c)$ ;
     $Gmean, AUC, F_1$  Score  $\leftarrow classify(A_{cm})$ 
  else
     $Gmean, AUC, F_1$  Score  $\leftarrow classify(A_c)$ ;
  end
  Results[i]  $\leftarrow Gmean, AUC, F_1$  Score
end
    
```

**3.1 Class decomposition**

Class decomposition is achieved by applying a clustering algorithm to a training set and aims at minimizing the bias-variance trade-off [43] by creating more local boundaries within the dataset. The method was presented by Vilalta et al. [44], where experiments using 20 datasets showed an improvement in performance in Naive Bayes and SVM. In [45], Polaka used hierarchical and *k-means* clustering to decompose the majority class instances and reported an improvement in RF performance. More recently in [7], Elyan and Gaber extended this approach by applying class decomposition to all classes in the dataset. The results showed significant improvement. The *K* value (number of clusters) was set experimentally in this work. Later on, the authors [46] showed that RF performance via decomposition could be optimized using genetic algorithm. More recently, CD was applied to a set of engineering symbols extracted from engineering drawings, and it proved that the performance of SVM, RF and convolutional neural networks (CNN) was improved significantly [15].

In this paper, we follow a similar approach to [7, 46] by applying *k-means* clustering algorithm to the majority class instances. By decomposing the majority class into *k* subclasses, we aim to achieve two goals. First, reduce the dominance of the majority class instances and also avoid the loss of information which often results from applying other undersampling methods. This is illustrated in Fig. 2.

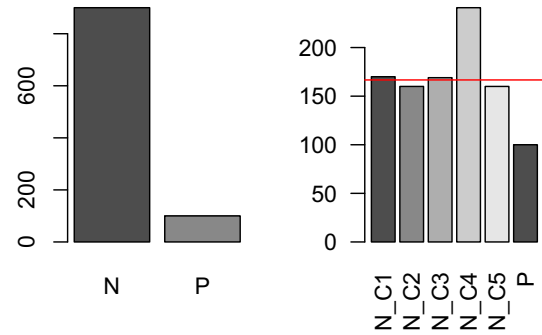


Fig. 2 Class decomposition applied to an imbalanced binary dataset

Figure 2 (left) shows the original dataset with the minority class instances (*P*), while the right side shows the dataset after applying class decomposition, which resulted in the same dataset but with different subclasses (clusters) representing the majority class instances (*N*) as  $N_{C1}, N_{C2}, \dots$ . Notice that with such an approach, we transform the dataset into different distributions and at the same time preserve all information. Consider the binary classification task in Eq. 3, where we want to learn  $h(x)$  that maps each instance  $x_i$  to a class  $y_i \in \{C_N, C_P\}$ .

$$h(X) : X \rightarrow Y \tag{3}$$

Notice that in a classification task such as in Eq. 5, we aim to minimize the number of misclassification as shown in Eq. 4

$$\min \left( \sum_{i=1}^m (y_i \neq \hat{y}_i) \right) \tag{4}$$

where  $y_i$  is the actual class label,  $\hat{y}_i$  is the predicted class label and  $m$  is the number of instances in the dataset. When we apply class decomposition to the dataset  $X$  in Eq. 3, we get a new classification task (Eq. 5).

$$h'(X) : X \rightarrow Y' \tag{5}$$

Here, we want to learn a function  $h'(x)$  that maps each instance  $x_i$  to the corresponding label  $y'_i \in \{C_{N1}, C_{N2}, \dots, C_{NK}, C_P\}$ , where *K* denotes the number of clusters. Notice that with such approach, we transform a binary classification problem into a multiclass classification problem. Here, transforming the data will not only reduce the dominance of the negative class  $C_N$  by clustering it into *K* subclasses, but will also allow training of the learning algorithm at a fine-grained level. The same objective function in Eq. 4 holds but with a minor change, such that each prediction is considered correct as far as it is within the main class of labels. In other words for any negative instance  $x_i \rightarrow C_N$ , a predicted label  $\hat{y}_i$  is considered correct, if and only if  $\hat{y}_i \in \{C_{N1}, C_{N2}, \dots, C_{NK}\}$ .

**Table 1** Datasets

Dataset	No. samples	No. features	Imbalance ratio
Glass1	214	9	1.82
Wisconsin	683	9	1.86
Ecoli0_vs_1	220	7	1.86
Pima	768	8	1.87
Iris0	150	4	2
Glass0	214	9	2.06
Yeast1	1484	8	2.46
Haberman	306	3	2.78
Vehicle2	846	18	2.88
Vehicle1	846	18	2.9
Vehicle3	846	18	2.99
Glass0123_vs_456	214	9	3.2
Vehicle0	846	18	3.25
Ecoli1	336	7	3.36
New-thyroid2	215	5	4.92
New-thyroid1	215	5	5.14
Ecoli2	336	7	5.46
Segment0	2308	19	6.02
Glass6	214	9	6.38
Yeast3	1484	8	8.1
Ecoli3	336	7	8.6
Page-blocks0	5472	10	8.79
Ecoli034_vs_5	200	7	9
Yeast2_vs_4	514	8	9.08
Ecoli067_vs_35	220	6	9.09
Ecoli0234_vs_5	202	7	9.1
Yeast0359_vs_78	506	8	9.12
Yeast0256_vs_3789	1004	8	9.14
Ecoli046_vs_5	203	6	9.15
Ecoli0346_vs_5	205	7	9.25
Ecoli0347_vs_56	257	7	9.28
Yeast05679_vs_4	528	8	9.35
Vowel0	988	13	9.98
Ecoli067_vs_5	220	6	10
Glass016_vs_2	192	9	10.29
Led7digit02456789_vs_1	443	7	10.97
Ecoli01_vs_5	240	6	11
Glass2	214	9	11.59
Ecoli0147_vs_56	332	6	12.28
Shuttle0_vs_4	1829	9	13.87
Yeast1_vs_7	459	7	14.3
Glass4	214	9	15.47
Ecoli4	336	7	15.8
Page-blocks13_vs_2	472	10	15.85
Abalone9-18	731	8	16.4
Dermatology6	358	34	16.87
Glass016_vs_5	184	9	19.44
Shuttle2_vs_4	129	9	20.5

**Table 1** (continued)

Dataset	No. samples	No. features	Imbalance ratio
Yeast1458_vs_7	693	8	22.1
Glass5	214	9	22.78
Yeast2_vs_8	482	8	23.1
Yeast4	1484	8	28.1
Winequality-red4	1599	11	29.17
Yeast1289_vs_7	947	8	30.57
Yeast5	1484	8	32.73
Ecoli0137_vs_26	281	7	39.14
Yeast6	1484	8	41.4
Winequality-red8_vs_67	855	11	46.5
Winequality-white39_vs_5	1482	11	58.28
Abalone19	731	8	129.44

### 3.2 Minority class oversampling

Applying CD to a dataset will result in different data distribution. In other words, new minority/majority-class instances may appear (from within the clusters of the majority class instances). So, first, we check whether the number of samples in the minority class is close to the average number of samples of the majority subclasses. For instance, in Fig. 2, it is shown that the minority class is below the average number of samples of the five subclasses after class decomposition. (The horizontal line represents the average in red color in Fig. 2.) In this case, an oversampling is applied to the minority class. To oversample, we chose SMOTE [28] due to its efficiency and popularity as one of the most common oversampling methods. SMOTE requires two classes as input (a minority and a majority) to perform the oversampling of the minority class using the majority class samples as a reference for the synthetic sample generation. In this paper, we use the majority subclass with the number of samples closest to the mean as the majority class input to SMOTE. In Fig. 2, this would be  $N_{C3}$ . In cases where a tie takes place (i.e., more than one majority class to chose), one is selected at random. It has to be noted that these simple heuristics were chosen empirically when implementing CDSMOTE to handle class-imbalance classification. In other words, it was found that oversampling the minority class when it falls below the average number of subclasses yields better results overall.

## 4 Experiments

A large-scale experiment has been carried out aiming at comparing CDSMOTE with other common undersampling methods for handling class-imbalance data classification. In this experiment, CDSMOTE is compared against SMOTE [28] and ADASYN [32]. These were chosen as they are among the most common undersampling methods in the literature. Moreover, CDSMOTE is compared against class decomposition [7] and with recent and state-of-the-art methods including [36, 47, 48]. The following subsections describe the experiment in details.

### 4.1 Datasets

A collection of 60 datasets was used in this experiment, and these are publicly available and commonly used in class-imbalance data classification (i.e., [36, 47, 48], ...). The datasets were obtained from the KEEL repository.<sup>1</sup> As can be seen in Table 1, these datasets are binary classification datasets with different imbalance ratios, different numbers of instances and a varied number of features.

### 4.2 Settings and implementation details

All datasets were partitioned into training and testing sets with a ratio of 80%, 20%, respectively, and fivefold cross-validation training. In all experiments, SVM with linear kernel was used as the learning algorithm. Other learning algorithms could have been considered, for example RF which showed the favorable results over other state-of-the-

<sup>1</sup> <http://sci2s.ugr.es/keel/imbalanced.php>.

**Table 2** CDSMOTE outperforming SMOTE, ADASYN and CD

Dataset	IR	Measure	Baseline	SMOTE	ADASYN	CD	CDSMOTE
Yeast1	2.46000	AUC	0.78900	0.79000	0.78800	0.87900	0.88400
Haberman	2.78000	AUC	0.70100	0.68900	0.67500	0.89200	0.89800
Haberman	2.78000	Fscore	0.06100	0.43900	0.44600	0.35400	0.47400
Vehicle2	2.88000	Gmean	0.96200	0.96200	0.95900	0.96600	0.96900
Vehicle2	2.88000	Fscore	0.94000	0.93400	0.93500	0.95000	0.95500
Glass0123_vs_456	3.20000	AUC	0.96800	0.96200	0.97000	0.97700	0.98000
Vehicle0	3.25000	Fscore	0.93500	0.94000	0.93400	0.93500	0.94500
Ecoli1	3.36000	Fscore	0.73800	0.77200	0.77100	0.79000	0.80000
Ecoli2	5.46000	Gmean	0.70000	0.91700	0.89400	0.78200	0.92000
Ecoli2	5.46000	Fscore	0.63200	0.72300	0.67000	0.65900	0.74700
Segment0	6.02000	Fscore	0.99000	0.99000	0.99000	0.99000	0.99500
Glass6	6.38000	AUC	0.94700	0.92300	0.92300	0.97900	0.98400
Yeast3	8.10000	AUC	0.96700	0.96900	0.96700	0.97700	0.97900
Yeast3	8.10000	Gmean	0.50600	0.89800	0.90900	0.74800	0.92300
Yeast3	8.10000	Fscore	0.40400	0.66900	0.62700	0.68600	0.79500
Ecoli3	8.60000	AUC	0.93300	0.94000	0.92600	0.95400	0.96000
Ecoli3	8.60000	Gmean	0.00000	0.88700	0.87500	0.00000	0.90300
Ecoli3	8.60000	Fscore	0.00000	0.57500	0.55100	0.00000	0.68600
Page-blocks0	8.79000	AUC	0.94200	0.96400	0.95900	0.98100	0.98400
Ecoli034_vs_5	9.00000	AUC	0.92200	0.87600	0.88100	0.97800	0.98500
Ecoli034_vs_5	9.00000	Gmean	0.88000	0.88400	0.87600	0.81200	0.90100
Ecoli034_vs_5	9.00000	Fscore	0.77900	0.72400	0.66800	0.75900	0.85000
Yeast2_vs_4	9.08000	AUC	0.88300	0.90300	0.91200	0.93800	0.96800
Yeast2_vs_4	9.08000	Fscore	0.45400	0.67200	0.62400	0.50900	0.74000
Ecoli0234_vs_5	9.10000	AUC	0.91100	0.88800	0.88400	0.97700	0.97800
Ecoli0234_vs_5	9.10000	Gmean	0.87600	0.90500	0.89300	0.85000	0.90600
Yeast0359_vs_78	9.12000	AUC	0.76400	0.77600	0.76500	0.93300	0.94000
Yeast0359_vs_78	9.12000	Fscore	0.33100	0.34600	0.30100	0.29300	0.39300
Ecoil046_vs_5	9.15000	AUC	0.87600	0.83000	0.80400	0.96600	0.98000
Ecoil046_vs_5	9.15000	Gmean	0.81100	0.83600	0.83000	0.78200	0.89500
Ecoil046_vs_5	9.15000	Fscore	0.78100	0.73400	0.64200	0.67900	0.85400
Ecoli0346_vs_5	9.25000	Gmean	0.84400	0.89900	0.87100	0.88300	0.93400
Ecoli0346_vs_5	9.25000	Fscore	0.77700	0.75700	0.61000	0.79000	0.82400
Ecoli0347_vs_56	9.28000	AUC	0.88400	0.88600	0.87000	0.97500	0.98100
Yeast05679_vs_4	9.35000	AUC	0.84000	0.84600	0.84500	0.94200	0.94500
Yeast05679_vs_4	9.35000	Fscore	0.00000	0.43300	0.42400	0.00000	0.45500
Vowel0	9.98000	Gmean	0.88400	0.94900	0.95800	0.93200	0.97600
Vowel0	9.98000	Fscore	0.83700	0.78900	0.81600	0.89900	0.91700
Glass016_vs_2	10.29000	Gmean	0.00000	0.51800	0.51600	0.00000	0.73700
Glass016_vs_2	10.29000	Fscore	0.00000	0.17600	0.19500	0.00000	0.22200
Led7digit02456789_vs_1	10.97000	AUC	0.95400	0.94900	0.92500	0.98000	0.98100
Ecoli01_vs_5	11.00000	AUC	0.93600	0.91000	0.91000	0.98600	0.98700
Glass2	11.59000	Fscore	0.00000	0.16200	0.14500	0.00000	0.17800
Ecoli0147_vs_56	12.28000	AUC	0.96200	0.95200	0.93900	0.98600	0.99000
Yeast1_vs_7	14.30000	Gmean	0.00000	0.76200	0.77500	0.00000	0.78400
Yeast1_vs_7	14.30000	Fscore	0.00000	0.29300	0.28400	0.00000	0.39100
Glass4	15.47000	AUC	0.99300	0.99300	0.99000	0.98800	0.99700
Glass4	15.47000	Gmean	0.00000	0.91200	0.90800	0.14100	0.97600



**Table 2** (continued)

Dataset	IR	Measure	Baseline	SMOTE	ADASYN	CD	CDSMOTE
Glass4	15.47000	Fscore	0.00000	0.69900	0.66200	0.18200	0.82300
Ecoli4	15.80000	Gmean	0.28300	0.94300	0.93700	0.38300	0.96100
Ecoli4	15.80000	Fscore	0.30200	0.69100	0.69100	0.38400	0.81300
Abalone9-18	16.40000	AUC	0.92000	0.92000	0.92800	0.96600	0.97600
Abalone9-18	16.40000	Fscore	0.00000	0.42000	0.42400	0.00000	0.50600
Glass5	22.78000	Fscore	0.00000	0.66700	0.61100	0.48600	0.77400
Yeast2_vs_8	23.10000	AUC	0.78700	0.82400	0.80600	0.97700	0.98100
Yeast2_vs_8	23.10000	Gmean	0.73400	0.74400	0.68200	0.73400	0.76500
Yeast2_vs_8	23.10000	Fscore	0.69100	0.62600	0.20200	0.69100	0.79200
Yeast4	28.10000	Fscore	0.00000	0.29000	0.27900	0.00000	0.31200
Winequality-red4	29.17000	Fscore	0.00000	0.12600	0.12600	0.00000	0.14000
Yeast1289_vs_7	30.57000	Fscore	0.00000	0.12800	0.12800	0.00000	0.15200
Yeast5	32.73000	Gmean	0.00000	0.96700	0.96600	0.00000	0.96900
Yeast5	32.73000	Fscore	0.00000	0.47300	0.47300	0.00000	0.54000
Yeast6	41.40000	Gmean	0.00000	0.89200	0.87400	0.00000	0.89700
Yeast6	41.40000	Fscore	0.00000	0.32700	0.24200	0.00000	0.36600
Winequality-red8_vs_67	46.50000	Fscore	0.00000	0.09200	0.09200	0.00000	0.10800
Abalone19	129.44000	Fscore	0.00000	0.03900	0.05800	0.00000	0.07500

**Table 3** CDSMOTE versus other methods using Gmean,  $F_1$  score and AUC ( $t$  test)

	Method	Gmean	$F_1$ -score	AUC
1	Baseline	0.00055801	0.00000107	0.00001961
2	SMOTE	0.02154869	0.00000004	0.00000756
3	ADASYN	0.00537313	0.00003774	0.00000731
4	CD	0.00090928	0.00000125	0.00030986

art methods [49] such as boosting and SVM. However, RF had shown already the favorable results concerning accuracy when class-decomposition was applied to the dataset as discussed in [7, 46]. In this work, we have chosen SVM with a linear kernel and default settings to establish the impact of class decomposition on class-imbalanced dataset classification.

Table 2 shows that each dataset was processed using SMOTE, ADASYN, CD, CDSMOTE and finally the baseline where no undersampling or oversampling. Regarding SMOTE and ADASYN, the number of nearest neighbors was set to equal 4 ( $k = 4$ ), and for class decomposition, we used  $k$ -means with  $k = 2$ . It is worth pointing out that we held these parameters fixed throughout, and no-parameter tuning was carried out to ensure a

fair comparison between methods and to assess the impact of CDSMOTE on learning from imbalanced datasets using the three different evaluation metrics. First, we evaluate the results using Area Under the Curve (AUC) of the receiving operating characteristic (ROC) curve, which is a plot of the sensitivity or true positive rate (TPR) as a function of the false positive rate (FPR). The second evaluation metric we used is geometric mean (Gmean), which measures the balance between the TPR and the true negative rate (TNR) and is defined as  $\sqrt{\text{TPR} \times \text{TNR}}$ . Finally, we used  $F_1$  Score between the TPR and the FPR [35] and is defined as  $F_1 \text{ Score} = \beta \times \frac{\text{TPR} \times \text{FPR}}{\text{TPR} + \text{FPR}}$ , with  $\beta$  value = 2.

The experiments were implemented using Python 3.6 and were carried out on a Windows 10 machine with 16 GB RAM and a 2.7 GHz processor.

### 4.3 Results

As can be seen in Table 2, CDSMOTE outperformed all methods across one or more evaluation metric in 39 datasets. Moreover, it was observed that across the 60 datasets, CDSMOTE outperformed at least one method in one or more comparison. The comparison against CD was made to establish the need for applying oversampling after reducing the dominance of the majority class instances.

**Table 4** CDSMOTE against SMOTE+HACT [47] using Gmean

Dataset	IR	CDSMOTE	SMOTE+HACT [47]
Wisconsin	1.86	0.96	<b>0.977</b>
Yeast1	2.46	0.582	<b>0.662</b>
Haberman	2.78	0.596	<b>0.625</b>
Vehicle1	2.9	<b>0.78</b>	0.63
Vehicle3	2.99	<b>0.723</b>	0.649
Glass0123_vs_456	3.2	<b>0.91</b>	0.866
Ecoli1	3.36	<b>0.88</b>	0.874
Ecoli2	5.46	<b>0.92</b>	0.892
Segment0	6.02	<b>0.997</b>	0.753
Glass6	6.38	<b>0.94</b>	0.884
Yeast3	8.1	<b>0.923</b>	0.869
Ecoli3	8.6	<b>0.903</b>	0.875
Ecoil034_vs_5	9	<b>0.901</b>	0.89
Yeast0256_vs_3789	9.14	0.766	<b>0.768</b>
Ecoli046_vs_5	9.15	<b>0.895</b>	0.878
Ecoli0346_vs_5	9.25	<b>0.934</b>	0.881
Ecoli0347_vs_56	9.28	0.823	<b>0.864</b>
Yeast05679_vs_4	9.35	0.776	<b>0.81</b>
Ecoli067_vs_5	10	<b>0.899</b>	0.835
Led7digit02456789_vs_1	10.97	0.874	<b>0.888</b>
Ecoli01_vs_5	11	<b>0.897</b>	0.894
Yeast1_vs_7	14.3	<b>0.784</b>	0.693
Glass4	15.47	<b>0.976</b>	0.867
Ecoli4	15.8	<b>0.961</b>	0.938
Yeast1458_vs_7	22.1	0.567	<b>0.646</b>
Glass5	22.81	<b>0.788</b>	0.497
Yeast2_vs_8	23.1	<b>0.765</b>	0.74
Yeast4	28.1	0.794	<b>0.829</b>
Yeast1289_vs_7	30.57	0.602	<b>0.691</b>
Ecoli0137_vs_26	39.14	0.733	<b>0.816</b>
Yeast6	41.4	<b>0.897</b>	0.873

Bold font indicates the winning method

A closer look at Table 2, and comparing the performance of CDSMOTE against all other methods using Gmean,  $F_1$  Score and AUC, we can see an improvement gained by applying CDSMOTE. Statistical significance of the results was measured using the paired  $t$  test. With 95% confidence, the  $p$ -values for paired  $t$ -tests between CDSMOTE and all other methods across the three evaluation metrics are shown in Table 3, which clearly show a statistically significant improvement in performance using CDSMOTE.

It was also observed from the results that the best improvement across the three evaluation metrics was achieved using  $F_1$  Score. This suggests that CDSMOTE improves the presence of the minority-class instances and reduces the dominance of the majority-class cases. The results show also that CDSMOTE did not lose in any dataset against the three different methods (ADASYN,

SMOTE and CD) combined. It was, however, observed that a similar performance (tie) was recorded in six different datasets and across the three evaluation metrics. These include Iris0, New-thyroid1, New-thyroid2, Shuttle0\_vs\_4, Shuttle2\_vs\_4 and Dermatology6. These are the datasets where 100% accuracy was recorded (i.e.,  $F_1$  Score = 1).

For further evaluation, we compared our method with recent state-of-the-art techniques using the most recent results and reported the same experiment settings and datasets. First, we consider Cleofas-Sanchez et al. [47], who attempted class-imbalance classification using 31 of the datasets through a hybrid associative classifier with translation (HACT) based on SMOTE and used Gmean for evaluating the results. Table 4 lists the performance of CDSMOTE against this method. Then, we considered Lin et al. [36] who presented a clustering-based undersampling

**Table 5** CDSMOTE against Clust+C4.5<sub>Ab</sub> [36] using AUC

Dataset	IR	CDSMOTE	Clust+C4.5 <sub>Ab</sub> [36]
Glass1	1.82	0.832	<b>0.834</b>
Ecoli0_vs_1	1.86	0.979	<b>0.983</b>
Wisconsin	1.86	0.983	<b>0.99</b>
Pima	1.87	<b>0.905</b>	0.758
Iris0	2	<b>0.99</b>	<b>0.99</b>
Glass0	2.06	<b>0.898</b>	0.89
Yeast1	2.46	<b>0.884</b>	0.747
Haberman	2.78	<b>0.898</b>	0.641
Vehicle2	2.88	0.992	<b>0.995</b>
Vehicle1	2.9	<b>0.933</b>	0.832
Vehicle3	2.99	<b>0.93</b>	0.848
Glass0123_vs_456	3.2	0.98	<b>0.982</b>
Vehicle0	3.25	<b>0.992</b>	0.946
Ecoli1	3.36	0.974	<b>0.94</b>
New-thyroid2	4.92	<b>0.996</b>	0.956
New-thyroid1	5.14	<b>0.999</b>	0.973
Ecoli2	5.46	<b>0.967</b>	0.947
Segment0	6.02	<b>1</b>	0.996
Glass6	6.38	<b>0.984</b>	0.917
Yeast3	8.1	<b>0.979</b>	0.967
Ecoli3	8.6	<b>0.96</b>	0.926
Page-blocks0	8.79	0.984	<b>0.986</b>
Yeast2_vs_4	9.08	0.968	<b>0.977</b>
Yeast05679_vs_4	9.35	<b>0.945</b>	0.869
Vowel0	9.98	<b>0.997</b>	0.987
Glass016_vs_2	10.29	<b>0.915</b>	0.75
Glass2	11.59	<b>0.923</b>	0.76
Shuttle0_vs_4	13.87	<b>1</b>	<b>1</b>
Yeast1_vs_7	14.3	<b>0.934</b>	0.768
Glass4	15.47	<b>0.997</b>	0.853
Ecoli4	15.8	<b>0.996</b>	0.95
Page-blocks13_vs_2	15.85	<b>0.995</b>	0.992
Abalone9-18	16.4	<b>0.976</b>	0.831
Glass016_vs_5	19.44	<b>0.991</b>	0.964
Shuttle2_vs_4	20.5	<b>1</b>	<b>1</b>
Yeast1458_vs_7	22.1	<b>0.858</b>	0.632
Glass5	22.78	<b>0.995</b>	0.949
Yeast2_vs_8	23.1	<b>0.981</b>	0.868
Yeast4	28.1	<b>0.958</b>	0.874
Yeast1289_vs_7	30.57	<b>0.936</b>	0.7
Yeast5	32.73	0.984	<b>0.987</b>
Ecoli0137_vs_26	39.14	<b>0.993</b>	0.838
Yeast6	41.4	0.973	<b>0.909</b>
Abalone19	129.44	<b>0.972</b>	0.728

Bold font indicates the winning method

method on 44 datasets and reported performance using AUC. An Ensemble Adaboost C4.5 classifier was used for classification. The results in comparison with CDSMOTE

is shown in Table 5. Finally, Zhu et al. [48] used 31 of the datasets used in this paper, and the authors adopted an algorithmic-based approach by designing their own classifier: Boundary-Eliminated Pseudoinverse Linear Discriminant (BEPILD). Table 6 compares CDSMOTE performance against BEPILD using the two metrics reported by the authors (AUC and Gmean).

Table 4 compares CDSMOTE with [47] in terms of Gmean. Notice that CDSMOTE obtains the better results in 20 out of 31 datasets. Using a paired *t* test on the 20 datasets where CDSMOTE wins shows a statistically significant difference with *p* value equal to 0.000506.

Table 5 compares CDSMOTE and [36] across AUC, where it is shown that CDSMOTE outperformed [36] in 37 out of 44 datasets. A paired *t* test shows significant statistical improvement with a *p* value of  $2.633 \times 10^{-7}$ .

Table 6 shows the comparison of CDSMOTE against the BEPILD method presented in [48] for Gmean and AUC. In terms of Gmean, CDSMOTE obtains the better results in 13 out of 31 datasets. The difference in performance in these datasets is not statistically significant (using *t* test resulted in a *p* value = 0.0695); however, for some application domains, such as health, life science and security, such improvement in performance could be crucial. Considering only the 13 winning datasets in Table 6, we found out a statistically significant improvement using CDSMOTE using a paired *t* test resulting in a *p* value of 0.006054. When measuring performance using AUC Table 6, our proposed method proved to be superior across almost all datasets. Out of 31 datasets, CDSMOTE outperformed BEPILD in 30 datasets. Using a *t* test, a *p* value of  $2.821 \times 10^{-10}$  was obtained.

#### 4.4 Discussion

To summarize, out of 60 datasets, CDSMOTE proved to be superior to the most common and established methods used in handling class-imbalanced datasets classification. These methods include SMOTE [28] and ADASYN [32] and CD [46]. The improvement across three common evaluation metrics (*Gmean*, *F<sub>1</sub> Score* and *AUC*) was statistically significant as shown in Tables 3. These results suggest that applying class decomposition to a majority class instances in a binary dataset does not only reduce the dominance of the majority class but also such decomposition provides a more linearly separable space within the local class boundaries.

The proposed method also showed superior performance over recent and state-of-the-art methods presented in the literature such as Cleofas-Sanchez et al. [36, 47], and [48], as can be seen in Tables 4, 5 and 6. An improvement over these methods was statistically significant. The results also showed that the best trade-off between AUC and Gmean,

**Table 6** CDSMOTE against BEPILD [48] using Gmean and AUC

Dataset	IR	CDSMOTE <sub>Gmean</sub>	BEPILD <sub>Gmean</sub>	CDSMOTE <sub>AUC</sub>	BEPILD <sub>AUC</sub>
Wisconsin	1.86	0.96	0.975	<b>0.983</b>	0.975
Pima	1.87	0.728	0.748	<b>0.905</b>	0.749
Yeast1	2.46	0.582	0.71	<b>0.884</b>	0.711
Vehicle1	2.9	0.78	0.787	<b>0.933</b>	0.788
Vehicle0	3.25	<b>0.968</b>	0.946	<b>0.992</b>	0.946
Glass6	6.38	<b>0.940</b>	0.919	<b>0.984</b>	0.92
Yeast3	8.1	<b>0.923</b>	0.899	<b>0.979</b>	0.899
Ecoli3	8.6	<b>0.903</b>	0.875	<b>0.960</b>	0.88
Ecoli034vs5	9	0.901	0.904	<b>0.985</b>	0.908
Ecoli067vs35	9.09	0.795	0.865	<b>0.973</b>	0.88
Ecoli0234vs5	9.1	0.906	0.908	<b>0.978</b>	0.917
Yeast0359vs78	9.12	0.695	0.744	<b>0.940</b>	0.755
Ecoil046vs5	9.15	<b>0.895</b>	0.892	<b>0.980</b>	0.906
Yeast05679vs4	9.35	0.776	0.784	<b>0.945</b>	0.789
Ecoli067vs5	10	<b>0.899</b>	0.874	<b>0.981</b>	0.877
Glass016vs2	10.29	<b>0.737</b>	0.734	<b>0.915</b>	0.75
Led7digit02456789vs1	10.97	0.874	0.898	<b>0.981</b>	0.902
Ecoli0147vs56	12.28	<b>0.902</b>	0.887	<b>0.990</b>	0.89
Yeast1vs7	14.3	<b>0.784</b>	0.764	<b>0.934</b>	0.766
Ecoli4	15.8	<b>0.961</b>	0.919	<b>0.996</b>	0.923
Abalone9-18	16.4	0.812	0.994	<b>0.976</b>	0.883
Dermatology6	16.87	<b>0.999</b>	0.882	<b>0.999</b>	0.994
Yeast1458vs7	22.1	0.567	0.62	<b>0.858</b>	0.632
Yeast4	28.1	0.794	0.83	<b>0.958</b>	0.831
Winequality-red4	29.17	0.662	0.696	<b>0.961</b>	0.7
Yeast1289vs7	30.57	0.602	0.716	<b>0.936</b>	0.725
Yeast5	32.73	<b>0.969</b>	0.962	0.984	0.987
Yeast6	41.4	<b>0.897</b>	0.875	<b>0.973</b>	0.878
Winequality-red8vs67	46.5	0.636	0.721	<b>0.970</b>	0.74
Winequality-white39vs5	58.28	0.586	0.629	<b>0.982</b>	0.657
Abalone19	129.44	0.723	0.775	<b>0.972</b>	0.728

Bold font indicates the winning method

the two classically used metrics for imbalanced datasets, was obtained by CDSMOTE. Overall, the proposed method obtains the better results in terms of AUC than other methods. CDSMOTE maximizes the  $F_1$  Score results, meaning that it effectively offers the best trade-off between the precision and recall for the minority class. Therefore, CDSMOTE can provide an alternative to handle the class-imbalance problem in specific scenarios. It has to be pointed out that there is a large room for improving these results. This includes hyper-parameters tuning and optimization (optimize the  $k$  value), further experiments with different learning algorithms (i.e., ensemble-based methods), using alternative clustering methods (i.e., soft clustering techniques, density-based and others) or using different oversampling methods such as GANs.

## 5 Conclusions and future work

In this paper, we have presented a new approach for handling class-imbalance problem by means of class decomposition. Unlike most common undersampling methods, our method suffers no data loss and preserves all majority class instances. A large-scale experiment showed that CDSMOTE produces the comparable results with state-of-the-art methods, while significantly outperforming some of the most established methods across metrics such as AUC, Gmean and  $F_1$  Score. The number of datasets used in this experiment with different sizes, dimensions and imbalance ratios suggests that the proposed methods can generalize and scalable across larger and more diverse datasets. It has to be noted that these results were obtained using default parameters settings and with one classifier, namely SVM

with a linear kernel, meaning that further improvement can be made at the data level as well as the algorithmic level. At the data level, the method presented in this paper can benefit from better clustering and grouping of the majority class instances. This might include isolating the instances within the overlapping region. At the algorithmic level, we intend to examine other learning algorithms, in particular, ensemble-based classification methods such as RF, which has proved to outperform other learning methods. Also, the use of other clustering methods can be explored for further improvement in the results. This might include considering density-based clustering methods instead of using *k-means*, which is often sensitive to noise. Finally, the results can be further improved by applying some parameter tuning techniques, to ensure that the best parameter setting is chosen.

## Compliance with ethical standards

**Conflict of interest** The authors declare that they have no conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Barandela R, Sánchez JS, García V, Rangel E (2003) Strategies for learning in class imbalance problems. *Pattern Recognit* 36:849–851
- Japkowicz N, Stephen S (2002) The class imbalance problem: a systematic study. *Intell Data Anal* 6(5):429–449
- Kotsiantis S, Kanellopoulos D, Pintelas P (2006) Handling imbalanced datasets: a review. *GESTS Int Trans Comput Sci Eng* 30(1):25–36
- Chawla NV (2005) Data mining for imbalanced datasets: an overview. In: Maimon O, Rokach L (eds) *Data mining and knowledge discovery handbook*. Springer, Boston, MA
- Haixiang G, Yijing L, Shang J, Mingyun G, Yuanue H (2017) Learning from class-imbalanced data: review of methods and applications. *Expert Syst Appl* 73:220–239
- Vuttipittayamongkol P, Elyan E, Petrovski A, Jayne C (2018) Overlap-based undersampling for improving imbalanced data classification. In: Yin H, Camacho D, Novais P, Tallón-Ballescros AJ (eds) *Intelligent data engineering and automated learning—IDEAL 2018*. Springer, Cham, pp 689–697
- Elyan E, Gaber MM (2016) A fine-grained random forests using class decomposition: an application to medical diagnosis. *Neural Comput Appl* 27(8):2279–2288
- Zhao XM, Li X, Chen L, Aihara K (2008) Protein classification with imbalanced data. *Proteins* 70(2):311–319
- García-Pedrajas N, Pérez-Rodríguez J, García-Pedrajas M, Ortiz-Boyer D, Fyfe C (2012) Class imbalance methods for translation initiation site recognition in DNA sequences. *Knowl Based Syst* 25(1):22–34
- Kim MJ, Kang DK, Kim HB (2015) Geometric mean based boosting algorithm with over-sampling to resolve data imbalance problem for bankruptcy prediction. *Expert Syst Appl* 42(3):1074–1082
- Vuttipittayamongkol P, Elyan E (2020) Overlap-based undersampling method for classification of imbalanced medical datasets. In: Maglogiannis I, Iliadis L, Pimenidis E (eds) *Artificial intelligence applications and innovations*. Springer, Cham, pp 358–369
- Li S, Hao F, Li M, Kim H-C (2013) Medicine rating prediction and recommendation in mobile social networks. In: Park JJ, Arabnia HR, Kim C, Shi W, Gil J-M (eds) *Grid and pervasive computing*. Springer, Berlin, pp 216–223
- Elyan E, Moreno-García CF, Johnston P (2020) Symbols in engineering drawings (SIED): an imbalanced dataset benchmarked by convolutional neural networks. In: Iliadis L, Angelov PP, Jayne C, Pimenidis E (eds) *Proceedings of the 21st EANN (engineering applications of neural networks) 2020 conference*. Springer, Cham, pp 215–224
- Elyan E, Jamieson L, Ali-Gombe A (2020) Deep learning for symbols detection and classification in engineering drawings. *Neural Netw* 129:91–102
- Elyan E, Moreno-García CF, Jayne C (2018) Symbols classification in engineering drawings. In: *International joint conference on neural networks (IJCNN)*
- Thai-Nghe N, Gantner Z, Schmidt-Thieme L (2010) Cost-sensitive learning methods for imbalanced data. In: *Proceedings of the international joint conference on neural networks (IJCNN)*, pp 1–8
- Estabrooks A, Jo T, Japkowicz N (2004) A multiple resampling method for learning from imbalanced data sets. *Comput Intell* 20(1):18–36
- Krawczyk B, Galar M, Jeleń Ł, Herrera F (2016) Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy. *Appl Soft Comput J* 38:714–726
- Stefanowski J (2013) Overlapping, rare examples and class decomposition in learning classifiers from imbalanced data. In: *Emerging paradigms in machine learning*. Springer, Berlin, pp 277–306
- Branco P, Torgo L, Ribeiro RP (2016) A survey of predictive modeling on imbalanced domains. *ACM Comput Surv* 49(2):31:1–31:50
- Seiffert C, Khoshgoftaar TM, Van Hulse J, Napolitano A (2010) Rusboost: a hybrid approach to alleviating class imbalance. *IEEE Trans Syst Man Cybern Part A Syst Hum* 40(1):185–197
- Tomek I (1976) An experiment with the edited nearest-neighbor rule. *IEEE Trans Syst Man Cybern SMC-6(6):448–452*
- Kubat M, Matwin S (1997) Addressing the curse of imbalanced training sets: one sided selection. *Int Conf Mach Learn* 97:179–186
- Devi D, Biswas S, Biswajit P (2017) Redundancy-driven modified torek-link based undersampling: a solution to class imbalance. *Pattern Recognit Lett* 93:3–12
- Laurikkala J (2001) Improving identification of difficult small classes by balancing class distribution. In: Quaglini S, Barahona P, Andreassen S (eds) *Artificial intelligence in medicine*. Springer, Berlin, pp 63–66

26. Bunkhumpornpat C, Sinapiromsaran K, Lursinsap C (2011) MUTE: majority under-sampling technique. In: International conference on information, communications and signal processing, pp 1–4
27. Vuttipittayamongkol P, Elyan E (2020) Neighbourhood-based undersampling approach for handling imbalanced and overlapped data. *Inf Sci* 509:47–70
28. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. *J Artif Intell Res* 16:321–357
29. Chawla NV, Lazarevic A, Hall LO, Bowyer KW (2003) Smoteboost: improving prediction of the minority class in boosting. In: Knowledge discovery in databases: KDD 2003. Springer, Berlin, pp 107–119
30. Bunkhumpornpat C, Sinapiromsaran K, Lursinsap C (2012) DBSMOTE: density-based synthetic minority over-sampling technique. *Appl Intell* 36(3):664–684
31. Barua S, Islam M, Yao X, Murase K (2014) MWMOTE: majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Trans Knowl Data Eng* 26:405–425
32. Haibo H, Bai Y, Garcia EA, Li S (2008) Adaptive synthetic sampling approach for imbalanced learning. *Int Jt Conf Neural Netw (IJCNN)* 3:1322–1328
33. Li S, Chen W, Li S, Leung K-S (2019) Improved algorithm on online clustering of bandits. In: Proceedings of the 28th international joint conference on artificial intelligence, AAAI Press, pp 2923–2929
34. Bezdek JC, Ehrlich R, Full W (1984) FCM: the fuzzy c-means clustering algorithm. *Comput Geosci* 10(2–3):191–203
35. Bunkhumpornpat C, Sinapiromsaran K (2017) DBMUTE: density-based majority under-sampling technique. *Knowl Inf Syst* 50(3):827–850
36. Lin WC, Tsai CF, Hu YH, Jhang JS (2017) Clustering-based undersampling in class-imbalanced data. *Inf Sci* 409–410:17–26
37. Yong Y (2012) The research of imbalanced data set of sample sampling method based on K-means cluster and genetic algorithm. *Energy Procedia* 17:164–170
38. Seoane Santos M, Henriques Abreu P, García-Laencina PJ, Simão A, Carvalho A (2015) A new cluster-based oversampling method for improving survival prediction of hepatocellular carcinoma patients. *J Biomed Inform* 58:49–59
39. Puntumapon K, Rakthamamon T, Waiyamai K (2016) Cluster-based minority over-sampling for imbalanced datasets. *IEICE Trans Inf Syst* 99(12):3101–3109
40. Lim P, Goh CK, Tan KC (2017) Evolutionary cluster-based synthetic oversampling ensemble (ECO-ensemble) for imbalance learning. *IEEE Trans Cybern* 47(9):2850–2861
41. Ali-Gombe A, Elyan E, Jayne C (2019) Multiple fake classes GAN for data augmentation in face image dataset. In: 2019 International joint conference on neural networks (IJCNN), pp 1–8
42. Ali-Gombe A, Elyan E (2019) MFC-GAN: class-imbalanced dataset classification using multiple fake class generative adversarial network. *Neurocomputing* 361:212–221
43. Geman S, Bienenstock E, Doursat R (1992) Neural networks and the bias/variance dilemma. *Neural Comput* 4(1):1–58
44. Vilalta R, Rish I (2003) A decomposition of classes via clustering to explain and improve naive Bayes. In: Machine learning: ECML 2003, pp 1–12
45. Polaka I (2013) Clustering algorithm specifics in class decomposition. In: Proceedings of the international scientific conference
46. Elyan E, Gaber MM (2017) A genetic algorithm approach to optimising random forests applied to class engineered data. *Inf Sci* 384:220–234
47. Cleofas-Sánchez L, Sánchez JS, García V, Valdovinos RM (2016) Associative learning on imbalanced environments: an empirical study. *Expert Syst Appl* 54:387–397
48. Zhu Y, Wang Z, Zha H, Gao D (2017) Boundary-eliminated pseudoinverse linear discriminant for imbalanced problems. *IEEE Trans Neural Netw Learn Syst* 29(6):2581–2594
49. Fernández-Delgado M, Cernadas E, Barro S, Amorim D (2014) Do we need hundreds of classifiers to solve real world classification problems? *J Mach Learn Res* 15:3133–3181

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.