# DBIG-US: A two-stage under-sampling algorithm to face the class imbalance problem

**Article** *in* Expert Systems with Applications · November 2020

DOI: 10.1016/j.eswa.2020.114301

**4 authors**, including:

Angélica Guzmán
Universidad Autónoma del Estado de México (UAEM)
8 PUBLICATIONS 5 CITATIONS
SEE PROFILE

Josep Salvador Sánchez
Universitat Jaume I
155 PUBLICATIONS 3,135 CITATIONS
SEE PROFILE

Rosa María Valdovinos
Universidad Autónoma del Estado de México (UAEM)
72 PUBLICATIONS 576 CITATIONS
SEE PROFILE

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

# DBIG-US: A two-stage under-sampling algorithm to face the class imbalance problem

A. Guzmán-Ponce [a,b,*], J.S. Sánchez [b], R.M. Valdovinos [a], J.R. Marcial-Romero [a]

[a] Facultad de Ingeniería, Universidad Autónoma del Estado de México, Cerro de Coatepec s/n, Ciudad Universitaria, 50100 Toluca, Mexico
[b] Institute of New Imaging Technologies, Department of Computer Languages and Systems, Universitat Jaume I, 12071 Castelló de la Plana, Spain

ARTICLE INFO

*Keywords:*
Imbalance problem
Under-sampling
Cluster analysis
DBSCAN
Graph theory

ABSTRACT

The class imbalance problem occurs when one class far outnumbers the other classes, causing most traditional classifiers perform poorly on the minority classes. To tackle this problem, a plethora of techniques have been proposed, especially centered around resampling methods. This paper introduces a two-stage method that combines the DBSCAN clustering algorithm to filter noisy majority class instances with a graph-based procedure to overcome the class imbalance. We then experimentally evaluate the behavior of the proposed method on a collection of two-class imbalanced data sets. The experimental results show an improvement in the classification performance measured by the geometric mean of the accuracy on each class and also a higher reduction in the imbalance ratio when compared to several state-of-the-art under-sampling techniques.

## 1. Introduction

The term 'class imbalance' refers to the situation in which the prior class probabilities are very different, that is, the number of instances in one class (often the one of prime interest) is much smaller than the amount of data available in the other classes. This form of imbalance is known as between-class imbalance or intrinsic imbalance, while the within-class imbalance or small disjuncts occurs when one class consists of several sub-clusters of different amount of instances (Ali et al., 2015; Thabtah et al., 2020). For a two-class problem, the minority and majority classes are usually referred to as positive ($C^+$) and negative ($C^-$), respectively (Kang et al., 2017).

Class imbalance is considered as one of the most challenging problems in data mining, machine learning and knowledge discovery, especially because standard learning models are strongly biased to favor the majority class. Since positive instances are under-represented, they are likely guessed as noise or outliers, or assigned to the majority class regardless the value of their features (García et al., 2019b; Haixiang et al., 2017). The problem of class imbalance is common to many real-world application domains such as fraud detection (Hassan & Abraham, 2016; Zhu et al., 2020b), medical diagnosis (Bach et al., 2017; Wang et al., 2020), credit risk and bankruptcy prediction (García et al., 2019a; Kim et al., 2015; Marqués et al., 2013), fault detection (Codetta-Raiteri & Portinale, 2015; Yang et al., 2020) and document categorization (Bruni & Bianchi, 2020; Jiang et al., 2019), to cite just a few examples.

Most techniques proposed in the literature to face the class imbalance problem can be categorized into three main groups: data-level or external, algorithmic-level or internal, and cost-sensitive methods (Fernández et al., 2018). The data-level approach is based on a preprocessing step to rebalance the class distribution by either under-sampling the majority class and/or over-sampling the minority class. The algorithmic-level approach creates or adapts the learning model for biasing the discrimination process toward the minority class. Finally, the cost-sensitive methods combine both data- and algorithmic-level approaches considers the varying costs of different misclassification types, and assign higher misclassification costs to the minority class. Apart from these strategies, ensemble learning appears as another option to handle the class imbalance problem (Galar et al., 2012; Haixiang et al., 2017). In general, the data-level solutions are the most popular and the most widely used because they are independent of the underlying classifier and can be easily implemented for any problem (Lin et al., 2017).

This paper is motivated by two observations. First, we concentrate on the under-sampling approach because the ever-increasing amount of data generated in many scientific and engineering domains makes over-sampling techniques much more expensive in memory and computational requirements for further classification tasks. On the other hand,

* Corresponding author at: Facultad de Ingeniería, Universidad Autónoma del Estado de México, Cerro de Coatepec s/n, Ciudad Universitaria, 50100 Toluca, Mexico.
*E-mail addresses:* aguzmanp643@alumno.uaemex.mx (A. Guzmán-Ponce), sanchez@uji.es (J.S. Sánchez), rvaldovinosr@uaemex.mx (R.M. Valdovinos), jrmarcialr@uaemex.mx (J.R. Marcial-Romero).

https://doi.org/10.1016/j.eswa.2020.114301
Received 17 January 2020; Received in revised form 6 November 2020; Accepted 9 November 2020
Available online 12 November 2020
0957-4174/Published by Elsevier Ltd.

the use of under-sampling techniques has also shown a positive effect on classifier performance, becoming a de facto strategy to work with asymmetric distributions (Drummond & Holte, 2003; Pozzolo et al., 2015). From the various strategies proposed for under-sampling the class-imbalanced data sets, we have chosen to use a clustering-based design because several related works have shown better performance when compared to neighborhood-based developments (Cao & Shen, 2019; Haixiang et al., 2017; Tsai et al., 2019; Yen & Lee, 2009). According to the experiments carried out in those works, the clustering-based under-sampling strategy can reduce the risk of removing meaningful data from the majority class (Lin et al., 2017); however, practical difficulties of this approach relate to the need of setting the number of clusters for an optimal clustering result, the possibility of changing the original distribution in the majority class because of the process to select or define representative samples from each cluster, and also the limited reduction in the imbalance ratio (Tsai et al., 2019).

Taking into account the pros and cons of the clustering-based under-sampling techniques, we propose a new method, called DBIG-US, which is based on the use of the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) clustering algorithm (Ester et al., 1996) and a graph-based procedure. Firstly, the DBSCAN depicts a filtering step to identify and remove noisy negative instances and then, the graph-based step produces a representative sub-sample of the majority class with a pre-established maximum imbalance ratio. By exhaustive experimentation, the proposed method is compared with some state-of-the-art under-sampling techniques on a large pool of two-class imbalanced data sets.

The main benefits of the DBIG-US algorithm can be summarized as follows:

- It strengthens the visibility of the minority class.
- It “cleans” the majority class by eliminating noise and removing borderline instances.
- It allows setting a maximum degree of imbalance.

Henceforward, this paper is organized as follows. Section 2 summarizes a collection of related works. The DBIG-US algorithm proposed here is introduced in Section 3. Next, Section 4 presents the experimental databases and describes the experimental setup. The results are discussed in Section 5. Finally, Section 6 remarks the main conclusions and outlines some avenues for further research.

## 2. Related works

Some data preprocessing methods combine under-sampling with an ensemble through the well-known boosting and bagging techniques (Lin et al., 2017). For instance, the RUSBoost (RBt) algorithm proposed by Seiffert et al. (2010) combines random under-sampling with a boosting algorithm for building an ensemble of classifiers. Other examples are EasyEnsemble (EE) and BalanceCascade (BC) (Liu et al., 2009), which both follow the idea of under-sampling by splitting the data into multiple subsets of the majority class and training an ensemble built on each subset; after this, the resampled data set is obtained by merging the best instances from each subset. The main difference between EE and BC is that the former is an unsupervised strategy to explore the data set by using random sampling with replacement, while the latter explores the data set in a supervised way. On the other hand, Kang et al. (2017) developed a noise-filtered under-sampling method (EE-KF), where the minority class is noise-filtered through removing instances whose all neighbors belong to the majority class and then, the majority class is under-sampled by the EasyEnsemble algorithm.

Solutions for class-imbalanced problems within the data-level approach focus on changing the class distribution of the data set by either under-sampling and/or over-sampling. While the under-sampling algorithms reduce the size of the majority class by removing some negative instances, the over-sampling methods augment the number of minority class instances. In general, the result of the under-sampling approach benefits from some computational advantages because it picks up a subset of the majority class for training, making the learning process faster and more efficient (Liang & Zhang, 2012).

Under-sampling can be conducted simply just by removing some majority class instances at random, but it can lead to loss of some potentially meaningful information. More advanced under-sampling techniques are based on some intelligent identification of less important negative instances to be eliminated. These methods can be categorized into two groups: neighborhood-based and clustering-based.

### 2.1. Neighborhood-based algorithms

The Hart’s condensing (CNN) algorithm (Hart, 1968) uses the concept of consistent subset to eliminate the majority class instances that are sufficiently far away from the decision boundary because these are considered to be irrelevant for learning. Analogously, the Tomek links (TL) (Tomek, 1976) have also been employed to remove the majority class instances since, if two instances form a Tomek link, then either one of these is noise or both instances are borderline.

Kubat and Matwin (1997) proposed the one-sided selection (OSS) technique, which selectively removes only those negative instances that either are redundant or border the minority class instances (assuming that these bordering cases are noise): the borderline instances are detected using the Tomek links, while the redundant ones are eliminated with Hart’s condensing. A similar method corresponds to the CNN-TL algorithm (Batista et al., 2004) which firstly finds a consistent subset and then applies the procedure based on the Tomek links.

Unlike the one-sided selection technique, the neighborhood cleaning (NCL) rule (Laurikkala, 2001) concentrates more on data filtering than on data reduction; to this end, Wilson’s editing (ENN) (Wilson, 1972) is employed to identify and remove noisy negative instances. According to the authors, NCL performs better than OSS and processes noisy instances more carefully. However, this method is strongly biased in favor of the minority class and leads to poor specificity and overall accuracy.

### 2.2. Clustering-based algorithms

Yoon and Kwek (2005) proposed the class purity maximization algorithm, which intends to split the majority class into dense clusters; the idea is to determine negative instances that are far away from the decision boundary, that is, to find as many clusters of majority class instances as possible that do not contain any positive instances or at most very few minority instances. The under-sampling based on clustering (SBC) proposed by Yen and Lee (2006) clusters all the training instances into $K$ clusters and selects an appropriate number of majority class samples from each cluster by considering the ratio of the number of majority class instances to the number of minority class instances in the derived cluster. Similarly, Longadge (2013) proposed an under-sampling method that firstly clusters the majority class instances into $K$ groups using the $K$-means algorithm, and then selects $|C^{+}| * IR_i$ majority class instances from each cluster $i$, where $IR_i$ denotes the imbalance ratio in the cluster $i$.

Rahman and Davis (2013) proposed a method based on clustering the majority class into $K$ clusters. Each cluster is combined with all the minority class instances to create $K$ training sets, which are then classified with a decision tree and the fuzzy unordered rule induction algorithm. Finally, the training set with the highest accuracy is designated as the final under-sampled data set for classification. The aim of this algorithm is not to obtain a data set with a majority to minority ratio of 1:1, but to reduce the difference in the size of the majority and minority classes.

The ClusterOSS technique introduced by Barella et al. (2014) is a modification of the OSS method. It starts by using the $K$-means algorithm to cluster the majority class instances. Then, the closest instances to the center of each cluster are used to start the application of OSS, removing borderline and noisy negative instances using the Tomek links. Unlike OSS, this technique does not start the under-sampling process at random, but defines how many and which instances should be chosen by applying the $K$-means clustering algorithm.

Sobhani et al. (2015) developed the ClusFirstClass algorithm that employs the $K$-means clustering and ensemble learning methods in order to obtain a balanced data set. First, majority class instances are clustered into $K$ groups and then, at least one instance from each cluster is selected to combine with all minority class instances, prior to training. Finally, to reduce information loss due to choosing small number of majority instances in highly imbalanced data sets, an ensemble is used to train several base learners with different subsets of majority class instances.

Chennuru and Timmappareddy (2017) proposed the MahalCUSFilter algorithm, which divides all the majority class instances into $K$ subsets based on the distance from their centroid and takes a number of negative instances from each subset using stratified sampling; this means that the number of instances to be chosen from each cluster depends on its size, and the total number of majority class instances to be chosen from all the groups is equal to the number of minority class instances in the data set. Thus the set with all minority class instances and selected the majority class instances form a balanced set. Finally, the negative instances that are misclassified with the nearest neighbor classifier are removed.

Lin et al. (2017) introduced two under-sampling strategies in which the $K$-means clustering technique is used. Unlike the proposals described previously, the number of clusters with majority class instances is defined to be equal to the size of the minority class. The first strategy uses the cluster centers to represent the majority class, whereas the second strategy uses the nearest neighbors of the centers. In the Fast-CBUS (fCBUS) method designed by Ofek et al. (2017), the minority class is clustered into $K$ clusters by the $K$-means algorithm and for each cluster, a similar number of majority class instances close to the minority class instances are sampled.

Tsai et al. (2019) presented the CBIS approach, which is based on clustering analysis and an instance selection process. Firstly, the affinity propagation algorithm groups similar majority class instances into a number $K$ of clusters. After this first step, the instance selection process is applied to filtering out instances in each cluster. Finally, all reduced $K$ clusters together with the instances of minority class form a balanced data set.

## 3. The DBIG-US algorithm

The under-sampling method proposed here differs from most cluster-based under-sampling algorithms in at least two questions. First it does not require to specify the number of clusters, which can be seen as an important advantage because this number is often unknown in advance, and second it allows to set the maximum imbalance ratio, say $maxIR$, that the resampled data set can tolerate.

Fig. 1 shows the general workflow of the BIG-US algorithm, which comprises two stages: the filtering step and the under-sampling step. It starts by dividing a two-class imbalanced data set of $n = n^+ + n^-$ instances (where $n^+ \ll n^-$ denote the number of positive and negative instances, respectively) into a subset $C^-$ with the majority class instances and a subset $C^+$ containing the minority class instances. Afterward, it comprises two steps: (i) the filtering step runs the DBSCAN clustering algorithm on the majority class to produce a noise-free set of majority class instances ($C_1^-$), and (ii) the graph-based step consists of the ShapeGraph algorithm to obtain an induced subgraph whose vertices represent a subset of the majority class ($C_2^-$) with an imbalance ratio $IR \leq maxIR$. Finally, $C^+$ is combined with $C_2^-$ to produce a balanced data set.

The proposed DBIG-US method is formally described in Algorithm 1. The filtering step (lines 3–6) applies DBSCAN until the free parameters do not change, and then the under-sampling step (see Section 3.2) is performed (line 7) to finally produce a balanced data set.

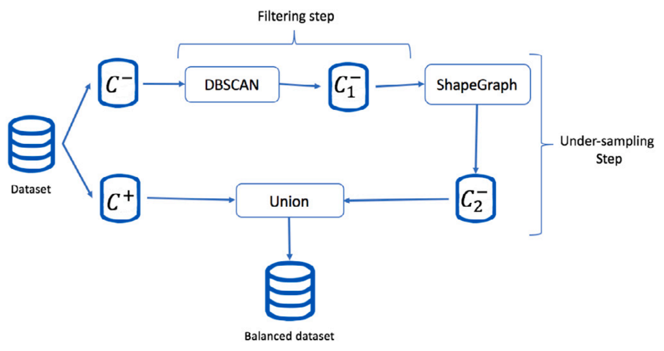


**Fig. 1.** General workflow of the proposed DBIG-US method.

**Algorithm 1** DBIG-US

```
Input: $DS = \{p_1, p_2, \dots, p_n\}$, $maxIR$
Output: $DS_{balanced}$
1: Split $DS$ into two subsets $C^-$ and $C^+$
2: $C_1^- \leftarrow C^-$
3: repeat
4:     Compute $\epsilon$ and $minPts$.
5:     $C_1^- \leftarrow$DBSCAN($C_1^-$, $\epsilon$, $minPts$).
6: until $\epsilon$ and $minPts$ do not change
7: $C_2^- \leftarrow$ ShapeGraph($C_1^-$, $R_{max}$, $C^+$)
8: $DS_{balanced} \leftarrow C^+ \cup C_2^-$
```

### 3.1. Filtering step

DBSCAN is a clustering algorithm, which assumes that the clusters correspond to dense regions in the space. This algorithm can discover groups without any prior knowledge about the number of clusters in the data, and it works well with noisy data sets. DBSCAN in its original proposal (Kumar & Rangan, 2007) requires two input parameters: $\epsilon$ defines the radius of the neighborhood region, and $minPts$ establishes the minimum number of instances that might be part of the $\epsilon$-neighborhood to form a cluster.

In our proposal, the input values of DBSCAN were modified according to Eqs. (1) and (2) to guarantee that the cluster formed by the majority class is free of noise and the decision border is cleaned.

$$\epsilon = \sqrt{\frac{\sum_{i=1}^{|C^-|} dist(m, p_i^-)}{|C^-|}} \tag{1}$$

where $m$ is the middle vector of the majority class, $p_i^-$ represents a majority class instance, $dist$ corresponds to the Euclidean distance between $p_i$ and $m$ and $|C^-|$ denotes the number of majority class instances.

$$minPts = \frac{\pi \times \epsilon^2}{TotalVolume} \times |C^+| \tag{2}$$

where $TotalVolume = \frac{4}{3} \times \pi \times \epsilon^3$ and $|C^+|$ is the number of minority class instances.

Bearing in mind that DBSCAN is here applied only on the majority class, it takes a negative instance $p_i^-$ at random and looks for all reachable instances with respect to $\epsilon$ and $minPts$. If $p_i^-$ does not have at least $minPts$ neighboring instances at a distance $\epsilon$, then it is marked as noisy and removed from the set $C^-$. This process is repeated until the values of the input parameters $minPts$ and $\epsilon$ do not change.

### 3.2. Under-sampling step

After removing noisy and borderline negative instances with DBSCAN, the process is refined using a graph-based technique to reduce the size of the majority class with the aim of obtaining an imbalance ratio lower than or equal to $maxIR$. Appendix A provides the concepts and notation on graph theory that are needed in this section. Let $C_1^-$ be the majority class produced by DBSCAN, then a weighted graph $G_w = (V, E)$ can be built as follows:

- $V(G_w) = C_1^-$,
- $E(G_w) = \{\{v, u\} \mid v, u \in V(G_w)\}$, and
- for each $e = \{v, u\} \in E(G_w)$, $w(e) = d(v, u)$ where $d(v, u)$ is the Euclidean distance between $v$ and $u$.

The rationale behind considering the majority class as the set of vertices of a weighted graph $G_w$ is to determine which instances are furthest away from each other. The ShapeGraph algorithm will be in charge of under-sampling the majority class by generating an induced subgraph of $G_w$ with a set of vertices whose size will be updated according to Eq. (3) and stored in the variable *Sample* until reaching an imbalance ratio equal to $maxIR$ (lines 2–7 in Algorithm 2).

$$f(x) = \begin{cases} \dfrac{\frac{|C_1^-| \quad \sigma^2 Z^2}{}}{e^2(|C_1^-|-1)+\sigma^2 Z^2} & \text{if } x = 1 \\ \\ \dfrac{f(x-1) \quad \sigma^2 Z^2}{e^2(|C_1^-|-1)+\sigma^2 Z^2} & \text{otherwise} \end{cases} \tag{3}$$

where $Z = 1.96$, $\sigma = 0.5$ and $e = 0.05$ for a confidence level of 95%.

After obtaining the size of the majority class subset (stored in *Sample*), the algorithm computes the adjacency matrix of $G_w$ (line 9) with the distance between each pair of instances (edges). The function GetMaximum gives a set of edges (pairs of instances) with a maximum distance in the adjacency matrix (line 11). For each edge in this set, the vertices that have not been visited yet will be marked and added to the majority class subset $C_2^-$ (lines 12–20).

**Algorithm 2** ShapeGraph

```
Input: $C_1^-$, $R_{max}$, $C^+$
Output: $C_2^-$
 1: Build $G_w = (V, E)$ a weighted graph with $V = C_1^-$
 2: $Sample \leftarrow |C_1^-|$
 3: $IR \leftarrow \frac{Sample}{|C^+|}$
 4: while $IR > maxIR$ do
 5:     $Sample \leftarrow \frac{Sample \quad \sigma^2 Z^2}{e^2(|C_1^-|-1)+\sigma^2 Z^2}$
 6:     $IR \leftarrow \frac{Sample}{|C^+|}$
 7: end while
 8: $C_2^- \leftarrow []$
 9: $M_G \leftarrow$ adjacencyMatrix($G_w$)
10: while $|C_2^-| < Sample$ do
11:     Maximum ← GetMaximum($M_G$)
12:     for all $\{v, u\}$ in Maximum do
13:         if $v$ has not been visited then
14:             Mark $v$ as visit
15:             $C_2^- \leftarrow C_2^- \cup \{v\}$
16:         end if
17:         if $u$ has not been visited then
18:             Mark $u$ as visit
19:             $C_2^- \leftarrow C_2^- \cup \{u\}$
20:         end if
21:         Remove $\{v, u\}$ from $M_G$
22:     end for
23: end while
24: return $C_2^-$
```

In general, the graph-based process discards instances that are not close enough to the decision border. Note that Eq. (3) determines the size of the majority class, and it is applied iteratively until the ratio between the computed value and the cardinality of the minority class is close to that given by the user.

**Table 1**
Main characteristics of the data sets.

| # | Data set | #Features | Class distribution | #Instances | IR |
|---|---|---|---|---|---|
| 1 | yeast05679v4 | 8 | 51–477 | 528 | 9.35 |
| 2 | vowel0 | 13 | 90–898 | 988 | 9.98 |
| 3 | glass016v2 | 9 | 17–175 | 192 | 10.29 |
| 4 | glass2 | 9 | 17–197 | 214 | 11.59 |
| 5 | shuttle0v4 | 9 | 123–1706 | 1829 | 13.87 |
| 6 | yeast1v7 | 7 | 30–429 | 459 | 14.30 |
| 7 | glass4 | 9 | 13–201 | 214 | 15.47 |
| 8 | ecoli4 | 7 | 20–316 | 336 | 15.80 |
| 9 | pagBks13v4 | 10 | 28–444 | 472 | 15.86 |
| 10 | glass016v5 | 9 | 9–175 | 184 | 19.44 |
| 11 | shuttle2vs4 | 9 | 6–123 | 129 | 20.50 |
| 12 | yeast1458vs7 | 8 | 30–663 | 693 | 22.10 |
| 13 | glass5 | 9 | 9–205 | 214 | 22.78 |
| 14 | yeast2vs8 | 8 | 20–462 | 482 | 23.10 |
| 15 | flareF | 11 | 43–1023 | 1066 | 23.79 |
| 16 | yeast4 | 8 | 51–1433 | 1484 | 28.10 |
| 17 | yeast1289v7 | 8 | 30–917 | 947 | 30.57 |
| 18 | yeast5 | 8 | 44–1440 | 1484 | 32.73 |
| 19 | ecoli0137v26 | 7 | 7–274 | 281 | 39.14 |
| 20 | abalone17v78910 | 8 | 58–2280 | 2338 | 39.31 |
| 21 | yeast6 | 8 | 35–1449 | 1484 | 41.40 |
| 22 | shuttle2v5 | 9 | 49–3267 | 3316 | 66.67 |
| 23 | kddBfOfw-b | 41 | 30–2203 | 2233 | 73.43 |
| 24 | poker89v5 | 10 | 25–2050 | 2075 | 82.00 |

### 3.3. Time complexity

As can be seen in Algorithm 1, DBIG-US consists of two steps. The time complexity of the first step is given by the run time complexity of the DBSCAN clustering algorithm, which is $\mathcal{O}(n^2)$ in the worst-case (Suthar et al., 2013).

The second step of DBIG-US corresponds to ShapeGraph (Algorithm 2) whose time complexity is also $\mathcal{O}(n^2)$. In this case, the complexity is mostly governed by the following set of instruction:

- Updating *Sample* and $IR$ (lines 4–7) has a complexity of $n$.
- Computing the adjacency matrix (line 9) has a complexity of $n^2$.
- Generating a majority class subset of size *Sample* (lines 10–23) has a complexity of $n^2$, which comes from $n$ iterations to compute the set $Maximum$ with a complexity of $n$.

Taking into account that DBIG-US depends on DBSCAN and ShapeGraph, the time complexity of the proposed algorithm becomes $\mathcal{O}(n^2)$ in the worst-case.

## 4. Experimental study

A set of experiments were designed to evaluate the performance of the under-sampling algorithm proposed here. The experiments consist of comparing DBIG-US with various well-known under-sampling methods on 24 two-class data sets taken from the KEEL Data Set Repository (https://sci2s.ugr.es/keel/imbalanced.php#subA). The number of instances varies from 129 to 3.316, the number of features ranges from 7 to 41, and the imbalance ratio (IR, the ratio of the majority class size to the minority class size) is between 9.35 and 82. Table 1 summarizes the main characteristics for these data sets, which are sorted in ascending order of the imbalance ratio.

Apart from DBIG-US, we also implemented the random under-sampling (RUS) approach, 4 neighborhood-based algorithms, 3 ensemble-based methods and 4 clustering-based algorithms to validate the performance of the proposal.

- Random under-sampling (RUS).
- Neighborhood-based: NCL, TL, ENN, OSS.
- Ensemble-based: BC, RUSBoost (RBt), EE-KF (EEKF).
- Clustering-based: SBC, CBU, Fast-CBUS (fCBUS), CBIS, DBIG-US.

**Table 2**
Confusion matrix.

| | Classified as positive | Classified as negative |
|---|---|---|
| Actual positive | True Positive (TP) | False Negative (FN) |
| Actual negative | False Positive (FP) | True Negative (TN) |

The base classifiers of all tested algorithms were the nearest neighbor rule (1-NN) and the C4.5 decision tree in order to obtain fair comparisons between the under-sampling approaches. The parameters of C4.5 were adopted as default values of the WEKA framework (Hall et al., 2009). For all data sets, the maximum imbalance ratio ($R_{max}$) allowed in the proposed DBIG-US algorithm was set to 1.

The experiments were carried out using the 10-fold cross-validation method. Each original data set is randomly divided into ten stratified parts of size $n/10$, where $n$ denotes the total number of instances in the data set; for each fold, nine blocks are used as a training set, and the remaining portion comprises the test set.

### *4.1. Assessment metrics*

To assess the classification performance on two-class imbalanced data sets, it is commonly used a $2 \times 2$ confusion matrix as that given in Table 2, where each entry $(i, j)$ contains the number of correct/incorrect classifications (Kang et al., 2017). From this, four counts can be easily derived: the number of true positives (actually positive, and classified as positive), true negatives (actually negative, and classified as negative), false positives (actually negative, but classified as positive), and false negatives (actually positive, but classified as negative).

Apart from the overall accuracy, other metrics can be calculated based on these counts, including the balanced correction rate or geometric mean of the accuracy on each class, the area under the ROC curve and the F-measure. All these reflect a trade-off between TPR and TNR without regard to class distribution or misclassification costs and therefore, these metrics are more suitable for class-imbalanced problems (García et al., 2014). In the experiments of this paper, the classification performance of algorithms was evaluated with the geometric mean of the accuracies:

$$g = \sqrt{TPR \cdot TNR}$$

where $TPR = TP/(TP + FN)$ is the sensitivity or true-positive rate (i.e., the accuracy on the minority class) and $TNR = TN/(TN + FP)$ is the specificity or true-negative rate (i.e., the accuracy on the majority class).

## 5. Results and discussion

To investigate the performance of the DBIG-US algorithm, this section is divided into four blocks. First, the effect of DBSCAN and the induced subgraph procedure has been analyzed separately to determine the synergy between both methods in DBIG-US. Second, the geometric mean will evaluate the effectiveness of the proposed method as compared with 14 state-of-the-art under-sampling techniques. Third, the imbalance ratio and its reduction percentage that result from the application of the under-sampling approaches will be analyzed. Finally, both performance measures will be put together to better understand the behavior of each under-sampling algorithm.

### *5.1. DBIG-US vs its components*

Results in Table 3 show the performance of the two algorithms involved in DBIG-US separately: the DBSCAN clustering algorithm and the induced subgraph (ShapeGraph). Results obtained over the data sets with no preprocessing have also been included for comparison purposes. Values reported in bold type represent the best result obtained on each data set.

As expected, the DBSCAN algorithm by itself improved the results obtained with the non-preprocessed data sets, which is likely due to the noise filtering process of this algorithm. However, DBSCAN did not achieve the best performance results.

Analyzing the results of the induced subgraph and the DBIG-US method, it can be observed that both outperformed the results obtained over the data sets with no preprocessing and also those processed by DBSCAN. Although ShapeGraph achieved the highest average geometric mean, differences with DBIG-US were mostly small.

Taking these results into account, it is important to highlight the advantages of the DBIG-US. First, remember that the induced subgraph is constructed by considering only the farthest samples from the center and those closest to the border between classes. In some cases, this situation could hinder the behavior of the classifier, mainly when the classes are strongly overlapped.

On the other hand, even though DBIG-US also employs the induced subgraph, it firstly cleans the decision boundary by applying the DBSCAN algorithm, thus moving away the intra-class boundaries. In this way, it is possible to have the benefits of both algorithms: to obtain a representative subset from the majority class, clean the decision border and reduce the class imbalance. Hence, results in Table 3 suggest that our proposal (the two-stage method) can be a better solution rather than its components when the data set suffers not only from class imbalance but also from other intrinsic data complexities such as noise.

### *5.2. DBIG-US performance vs other methods*

In Appendix B, Tables B.11 and B.12 respectively report the geometric mean (averaged across the 10 runs) of each method using the 1-NN and C4.5 classifiers on all 24 data sets. We also included the classification results on the original (baseline) data sets, which can be deemed as a baseline (reference). Besides, the last rows in these tables show the average geometric mean and the Friedman's average rank for each under-sampling algorithm. Values in boldface highlight the best performing method in each data set.

Fig. 2 displays the Friedman's average rank of each algorithm (last row in Tables B.11 and B.12). From these bar graphs, we can observe that DBIG-US achieved the lowest average ranks, which means that the method proposed here can be deemed as the best performing algorithm.

We also conducted pair-wise comparisons between DBIG-US and the state-of-the-art algorithms to further validate the performance of DBIG-US. Tables 4 and 5 summarize the comparisons for the 1-NN and C4.5 classifiers respectively, where the symbol "+" denotes that DBIG-US outperformed the algorithm to be compared, "=" indicates that both achieved the same performance, and "–" means that DBIG-US performed worse than the method in comparison. The last column shows how many times the performance of DBIG-US was superior to that of the other algorithms on each data set, while the last row reports the total number of times that DBIG-US was better/same/worse (W/T/L) than the method of the column.

Table 4 shows that DBIG-US was better than RUSBoost (RBt) and CBU on 19 out of the 24 data sets, but it was worse on the remaining 4 and 3 data sets, respectively. DBIG-US also outperformed the non-preprocessed option, TL and OSS on 17 and 16 databases respectively, and it was better than RUS on 16 data sets. The performance of DBIG-US was still superior than ENN and NCL on 15 and 13 databases, respectively. From these results, it is possible to observe that DBIG-US yielded better classification performance than the state-of-the-art algorithms on most databases using the 1-NN classifier.

Similar reseults were obtained when the C4.5 classifier was used (Table 5). Firstly, DBIG-US was better than NCL on 19 data sets. It also outperformed the SBC method on 18 data sets and ENN on 17 data sets. Regarding the clustering-based methods, DBIG-US was better than Fast-CBUS and CBIS on 17 data sets respectively. Thus, the overall performance of DBIG-US was higher than the other methods on most databases.

**Table 3**
Classification results of DBIG-US and its components.

| | 1NN | | | | C4.5 | | | |
|---|---|---|---|---|---|---|---|---|
| | Baseline | DBSCAN | ShapeGraph | DBIG-US | Baseline | DBSCAN | ShapeGraph | DBIG-US |
| yeast05679v4 | 61.2 | 62.9 | **93.7** | 81.9 | 65.1 | 60.7 | **87.9** | 76.1 |
| vowel0 | **100** | 99.9 | **100** | 99.9 | 96.9 | **97.2** | 96.0 | 96.7 |
| glass016v2 | 47.0 | 67.7 | **91.1** | 88.4 | 52.3 | 63.04 | **94.1** | 77.5 |
| glass2 | 40.7 | 64.8 | **93.9** | 90.5 | 53.1 | 64.0 | **97.0** | 84.3 |
| shuttle0v4 | **99.6** | **99.6** | **99.6** | **99.6** | **100** | 99.6 | **100** | **100** |
| yeast1v7 | 62.3 | 57.9 | **94.9** | 79.3 | 54.3 | 50.8 | **84.9** | 63.3 |
| glass4 | 86.6 | 89.4 | 88.4 | **96.1** | 82.2 | **92.3** | 80.7 | 91.9 |
| ecoli4 | 86.3 | 86.3 | **94.9** | **94.9** | **82.9** | 80.1 | 81.4 | 81.5 |
| pagBks13v4 | 98.2 | **98.2** | 98.1 | 98.1 | **96.2** | **96.2** | 94.3 | 94.3 |
| glass016v5 | 81.0 | **94.3** | 88.9 | **94.3** | 99.4 | 99.7 | 83.1 | **100** |
| shuttle2vs4 | **91.3** | **91.3** | **91.3** | **91.3** | 90.9 | 91.3 | **100** | **100** |
| yeast1458vs7 | 44.1 | 37.3 | **90.6** | 89.2 | 0.0 | 0.0 | 91.3 | **94.0** |
| glass5 | 81.1 | **94.3** | **94.3** | **94.3** | 98.8 | 99.4 | 83.1 | **100** |
| yeast2vs8 | 77.0 | 78.3 | **92.2** | 87.7 | 22.4 | 21.0 | 79.1 | **81.6** |
| flareF | 30.1 | 77.9 | 76.6 | **94.2** | 15.2 | 73.5 | 78.9 | **91.8** |
| yeast4 | 58.8 | 55.4 | 88.4 | **91.5** | 53.9 | 56.0 | 82.3 | **88.6** |
| yeast1289v7 | 47.6 | 52.8 | **94.9** | 91.5 | 48.2 | 45.2 | **96.5** | 84.3 |
| yeast5 | 82.1 | 82.8 | 86.3 | **88.3** | 86.3 | 87.3 | **87.7** | 83.0 |
| ecoli0137v26 | **83.7** | **83.7** | 65.5 | 65.5 | **84.4** | **84.4** | 57.1 | 57.1 |
| abalone17v78910 | 50.6 | 44.7 | **97.2** | 97.8 | 34.7 | 33.3 | **92.7** | **92.7** |
| yeast6 | 71.1 | 70.9 | **93.5** | 85.7 | 73.4 | 70.4 | **90.8** | 85.4 |
| shuttle2v5 | **100** | **100** | **100** | **100** | **100** | **100** | **100** | **100** |
| kddBfOfw | **100** | **100** | 98.4 | 98.4 | **98.3** | **98.3** | 95.0 | 95.0 |
| poker89v5 | 19.9 | 80.5 | 87.7 | **90.0** | 0.0 | **73.0** | 56.6 | 70.0 |
| Avg. Gmean | 70.9 | 77.9 | **91.7** | 91.2 | 66.2 | 72.4 | **87.1** | 87.0 |

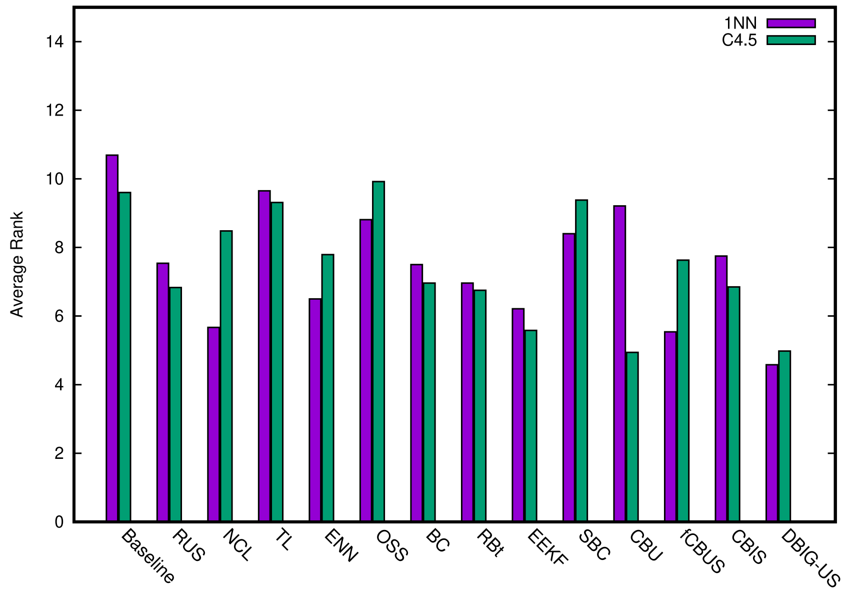


**Fig. 2.** Friedman's average ranks.

As a further confirmation of the findings using the ranks and pairwise comparisons, we also ran a Holm's post hoc test to find out statistically significant differences between our proposal and the rest of the algorithms. Holm's test is a step-down procedure that sequentially checks the hypothesis ordered by their significance (García et al., 2010).

Tables 6 and 7 report the $p$-values obtained by applying the Holm's post hoc test over the results of Friedman procedure. Holm's test rejects those hypotheses that have an unadjusted $p$-value $\leq 0.00625$ for both classifiers. The first column indicates the number of hypotheses, the order is according to their significance from the most to the lowest. DBIG-US and CBU were the control methods for the 1NN and C4.5 classifiers, respectively. The control method was chosen according to the best Friedman ranks: 4.58 (1NN) and 4.94 (C4.5).

Values in Table 6 indicate that DBIG-US was significantly better than the original (Baseline), TL, CBU, OSS, and SBC methods using the 1NN classifier. However, when using the C4.5 classifier (Table 7), the Holm's post hoc test found that there did not exist statistical differences between DBIG-US and CBU.

According to the results of Holm's post hoc test when the C4.5 classifier was used, the DBIG-US algorithm was not selected as the control method for comparison. Thus, the Wilcoxon's paired signed-rank test was also run to find out statistically significant differences between each pair of algorithms for two levels of significance ($\alpha = 0.90$ and $\alpha = 0.95$). This statistic ranks the differences in performances of two algorithms for each data set, ignoring the signs, and compares the ranks for the positive and the negative differences (Demšar, 2006).

The upper diagonal half of Table 8 (1-NN classifier) and Table 9 (C4.5 decision tree) corresponds to $\alpha = 0.90$ (10% or less chance), whereas the lower diagonal half is for a significance level of 0.95. The symbol "•" indicates that the method of the row was significantly better

**Table 4**
Comparison between DBIG-US and the other algorithms using 1-NN.

| Data set | Baseline | RUS | NCL | TL | ENN | OSS | BC | RBt | EEKF | SBC | CBU | fCBUS | CBIS | sum |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| yeast05679v4 | + | + | – | + | + | + | + | + | + | + | + | – | + | 11 |
| vowel0 | – | + | – | – | – | – | + | + | + | + | + | – | – | 6 |
| glass016v2 | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| glass2 | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| shuttle0v4 | = | = | = | = | = | – | = | = | – | = | = | + | = | 1 |
| yeast1v7 | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| glass4 | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| ecoli4 | + | – | + | + | + | + | – | + | + | + | + | – | + | 10 |
| pagBks13v4 | – | – | – | – | – | – | + | + | + | – | + | – | – | 4 |
| glass016v5 | + | + | – | + | – | + | = | + | + | + | + | – | – | 8 |
| shuttle2vs4 | = | = | = | = | = | – | – | – | – | = | + | – | – | 1 |
| yeast1458vs7 | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| glass5 | + | = | + | + | + | + | + | + | – | + | + | – | + | 10 |
| yeast2vs8 | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| flareF | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| yeast4 | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| yeast1289v7 | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| yeast5 | + | – | – | + | – | – | – | – | – | + | – | – | + | 4 |
| ecoli0137v26 | – | – | – | – | – | – | – | – | – | – | – | – | – | 0 |
| abalone17v78910 | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| yeast6 | + | + | – | + | + | + | + | + | – | + | + | + | + | 11 |
| shuttle2v5 | = | + | = | = | = | = | = | + | + | = | = | = | = | 3 |
| kddBfOfw | – | – | – | – | – | – | – | – | – | – | – | – | – | 0 |
| poker89v5 | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| W/T/L | 17/3/4 | 16/3/5 | 13/3/8 | 17/3/4 | 15/3/6 | 16/1/7 | 16/3/5 | 19/1/4 | 17/0/7 | 18/3/3 | 19/2/3 | 13/1/10 | 16/2/6 | |

**Table 5**
Comparison between DBIG-US and the other algorithms using C4.5.

| Data set | Baseline | RUS | NCL | TL | ENN | OSS | BC | RBt | EEKF | SBC | CBU | fCBUS | CBIS | sum |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| yeast05679v4 | + | + | + | + | + | + | + | – | + | + | – | – | + | 9 |
| vowel0 | – | + | + | – | – | – | + | + | + | + | + | + | + | 9 |
| glass016v2 | + | + | + | + | + | + | + | + | + | + | – | + | + | 12 |
| glass2 | + | – | + | + | + | + | + | + | + | + | – | + | + | 11 |
| shuttle0v4 | = | = | + | = | = | = | = | + | = | + | + | + | = | 5 |
| yeast1v7 | + | – | + | – | + | + | – | – | + | + | – | – | + | 7 |
| glass4 | + | + | + | + | + | + | – | + | + | + | + | + | + | 12 |
| ecoli4 | – | – | + | – | + | – | – | – | – | + | – | – | – | 3 |
| pagBks13v4 | – | – | – | – | – | – | – | – | + | – | – | – | – | 1 |
| glass016v5 | + | + | + | + | + | + | + | + | + | + | = | + | + | 12 |
| shuttle2vs4 | + | = | + | + | + | = | = | + | = | + | = | + | + | 8 |
| yeast1458vs7 | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| glass5 | + | = | + | + | + | + | + | + | = | = | = | + | + | 9 |
| yeast2vs8 | + | + | + | + | + | + | + | + | + | + | – | + | + | 12 |
| flareF | + | + | + | + | + | + | + | + | – | + | + | + | + | 12 |
| yeast4 | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| yeast1289v7 | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| yeast5 | – | – | – | – | – | – | – | – | – | – | – | – | – | 0 |
| ecoli0137v26 | – | – | – | – | – | – | – | – | – | – | – | – | – | 0 |
| abalone17v78910 | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| yeast6 | + | + | + | + | + | + | – | + | – | + | – | + | + | 10 |
| shuttle2v5 | = | = | = | = | = | = | = | = | = | = | + | + | = | 2 |
| kddBfOfw | – | = | – | – | – | – | = | – | – | – | + | – | – | 1 |
| poker89v5 | + | + | + | + | + | + | + | + | + | + | + | + | + | 13 |
| W/T/L | 16/2/6 | 13/5/6 | 19/1/4 | 15/2/7 | 17/2/5 | 15/3/6 | 13/4/7 | 16/1/7 | 14/4/6 | 18/2/4 | 11/3/10 | 17/0/7 | 17/2/5 | |

than the method of the column, and the symbol "∘" means that the method of the column performed significantly better than the method of the row. The two bottom rows show how many times the algorithm of the column was significantly better than the remaining techniques.

Results in Tables 8 and 9 show that DBIG-US was statistically better than the other methods for both levels of significance when the 1-NN classifier was used. However, when the C4.5 tree was applied, CBU was the best method for $\alpha = 0.95$ and, statistically equal than DBIG-US for $\alpha = 0.90$. Finally, analyzing both methods (DBIG-US and CBU) we can see that there are not statistical differences in both levels of significance (see Table 9). These results indicate that DBIG-US is a competitive proposal to face the class imbalance problem. On the other hand, the ensemble-based methods were among the top-5 under-sampling algorithms with the C4.5 classifier, which suggests that splitting the majority class into several subsets and training an ensemble may give rise to significantly better performance than the neighborhood-based under-sampling algorithms (NCL, TL, ENN, and OSS).

## 5.3. Rebalancing performance

A second question of interest to the present study is the rebalancing performance of the resampling algorithms, that is, their capability to produce an equal or similar number of instances in both classes. This will allow to gain further understanding of how the removal of majority class instances affects the classification performance. Here the rebalancing performance was evaluated using the imbalance ratio of the sets produced by the application of the under-sampling methods.

Table 10 provides the imbalance ratio of each under-sampled data set. Initially, the imbalance ratio of the 24 data sets was between 9.35 and 82 (see Table 1). After running the under-sampling algorithms

**Table 6**
Holm's post hoc test ($\alpha = 0.05$) for the 1NN classifier.

| | Algorithm | $z = (R_0 - R_i)/SE$ | $p$ | Holm |
|---|---|---|---|---|
| 13 | Baseline | 5.05473 | 0 | 0.003846 |
| 12 | TL | 4.192148 | 0.000028 | 0.004167 |
| 11 | CBU | 3.829864 | 0.000128 | 0.004545 |
| 10 | OSS | 3.502083 | 0.000462 | 0.005 |
| 9 | SBC | 3.15705 | 0.001594 | 0.005556 |
| 8 | CBIS | 2.622249 | 0.008735 | 0.00625 |
| 7 | RUS | 2.449733 | 0.014296 | 0.007143 |
| 6 | BC | 2.415229 | 0.015725 | 0.008333 |
| 5 | RBt | 1.966687 | 0.049219 | 0.01 |
| 4 | ENN | 1.587151 | 0.112478 | 0.0125 |
| 3 | EEKF | 1.345628 | 0.178423 | 0.016667 |
| 2 | NCL | 0.897085 | 0.369673 | 0.025 |
| 1 | fCBUS | 0.793575 | 0.427443 | 0.05 |

**Table 7**
Holm's post hoc test ($\alpha = 0.05$) for the C4.5 classifier.

| | Algorithm | $z = (R_0 - R_i)/SE$ | $p$ | Holm |
|---|---|---|---|---|
| 13 | OSS | 4.123142 | 0.000037 | 0.003846 |
| 12 | Baseline | 3.864367 | 0.000111 | 0.004167 |
| 11 | SBC | 3.674599 | 0.000238 | 0.004545 |
| 10 | TL | 3.622844 | 0.000291 | 0.005 |
| 9 | NCL | 2.932779 | 0.003359 | 0.005556 |
| 8 | ENN | 2.363475 | 0.018104 | 0.00625 |
| 7 | fCBUS | 2.225461 | 0.02605 | 0.007143 |
| 6 | BC | 1.673409 | 0.094247 | 0.008333 |
| 5 | CBIS | 1.587151 | 0.112478 | 0.01 |
| 4 | RUS | 1.569899 | 0.116439 | 0.0125 |
| 3 | RBt | 1.500893 | 0.133383 | 0.016667 |
| 2 | EEKF | 0.534801 | 0.592788 | 0.025 |
| 1 | DBIG-US | 0.034503 | 0.972476 | 0.05 |

enumerated in Section 4, the imbalance ratio decreased considerably (it was equal or close to 1.0) when RUS, BC, EE-KF, Fast-CBUS, CBIS and DBIG-US were used. Conversely, most data sets that were resampled by NCL, TL, ENN and OSS were still heavily class-imbalanced, thus leading to low rebalancing performance. For instance, taking a look at the results over the poker89v5 database, RUS and DBIG-US yielded an imbalance ratio equal to 1.00, while the imbalance ratios produced by the use of NCL, TL, ENN and OSS (79.4, 81.7, 80.5 and 80.6, respectively) were very similar to that of the original data set.

Another way to assess the rebalancing performance is to calculate the reduction percentage in the imbalance ratio. Results reveal that the top-2 under-sampling methods were RUS with an average reduction of 95.1% and the proposed DBIG-US algorithm with an average reduction of 94.2%. On the other hand, the average reductions achieved with NCL, TL, ENN and OSS were extremely low (7.5%, 2.6%, 6.8% and 35.1%, respectively), which corroborates the previous discussion of the results given in Table 10.

### 5.4. *Classification performance vs rebalancing performance*

As already remarked in Section 1, the behavior of standard learning models has shown a significant loss of performance when the class imbalance is presented. In fact, various studies (López et al., 2013; Thabtah et al., 2020; Zhu et al., 2020a) reveal that the relationship between the imbalance ratio and the classifier performance is tight, suggesting that a skewed class distribution could degrade the performance of standard learners. In general, when there is no class imbalance in a data set, the performance of the classifier is uniform, stable and acceptable, but this is not the case when the imbalance ratio is moderate or severe. This is the reason why we analyzed the imbalance ratio in conjunction with the geometric mean, with the aim of investigating the effect of the final imbalance ratio on the classifier performance.

To gain a better understanding of the behavior of each under-sampling, Fig. 3 displays all the algorithms in the space spanned by the average geometric mean on the $x$-axis and the reduction in the imbalance ratio on the $y$-axis. A method with perfect performance will be located on the upper right corner (100% geometric mean, 100% reduction in IR) of the plot. The closer the method is to the upper right corner, the better its behavior.

In summary, the following conclusions can be drawn from the combined analysis of classification performance and rebalancing performance:

- The proposed DBIG-US, CBU, Fast-CBUS and the three ensemble-based methods are the techniques located closest to the upper right corner of the plots, which means that these yielded the best classification and rebalancing performances.
- Taking into account both classifiers, DBIG-US obtained the best classification performance in terms of geometric mean and reduction in imbalance ratio.
- The neighborhood-based algorithms together with SBC and CBIS showed low performance in both classification and rebalancing the classes.

## 6. Conclusions and future research

This paper presents a novel under-sampling method (called DBIG-US) that combines the DBSCAN clustering algorithm for filtering the data set and a strategy based on a weighted graph to reduce the size of the majority class. This proposal takes advantage of the capability of DBSCAN for detecting and removing noisy instances, thus increasing the quality of the training set and obtaining a representative data set in which the classes were balanced. Another substantial issue is that the complexity of the proposed algorithm in the worst-case is $\mathcal{O}(n^2)$.

The behavior of the under-sampling techniques used in the experiments has been analyzed with the geometric mean of the accuracy on

**Table 8**
Summary of the Wilcoxon's test when using the 1-NN classifier.

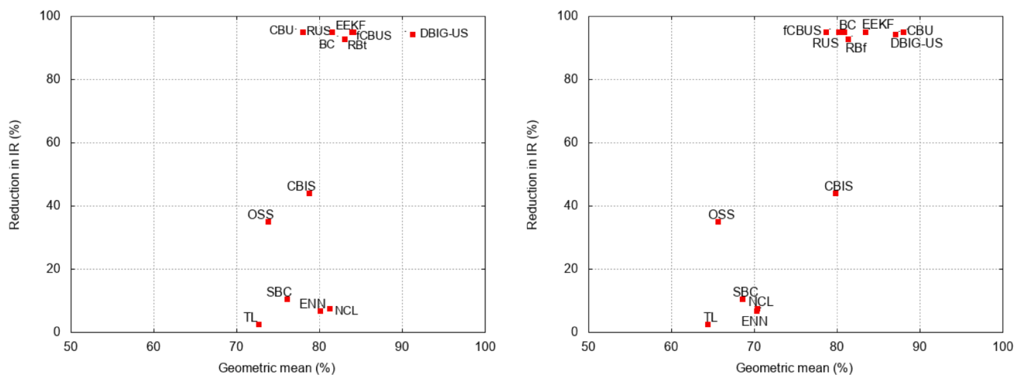
| | Baseline | RUS | NCL | TL | ENN | OSS | BC | RBt | EEKF | SBC | CBU | fCBUS | CBIS | DBIG-US |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Base line | – | ∘ | ∘ | ∘ | ∘ | ∘ | ∘ | ∘ | ∘ | ∘ | | ∘ | ∘ | ∘ |
| RUS | • | – | | • | | | | | | | | | | ∘ |
| NCL | • | | – | • | | • | | | | • | • | | • | ∘ |
| TL | • | ∘ | ∘ | – | ∘ | | ∘ | ∘ | ∘ | ∘ | | ∘ | ∘ | ∘ |
| ENN | • | | | • | – | • | | | | • | • | ∘ | | ∘ |
| OSS | • | | ∘ | | ∘ | – | ∘ | ∘ | ∘ | | | ∘ | | ∘ |
| BC | • | | | • | | • | – | | | | | | | ∘ |
| RBt | • | | | • | | • | | – | | | • | | • | ∘ |
| EEKF | • | | | • | | • | | | – | • | • | | • | ∘ |
| SBC | • | | ∘ | • | ∘ | | | | ∘ | – | | ∘ | | ∘ |
| CBU | | | ∘ | | | | | ∘ | ∘ | | – | ∘ | | ∘ |
| fCBUS | • | | | • | | • | | | | • | • | – | • | |
| CBIS | • | | | • | | | | | | | | ∘ | – | ∘ |
| DBIG-US | • | • | • | • | • | • | • | • | • | • | • | | • | – |
| $\alpha = 0.90$ | 0 | 2 | 6 | 1 | 5 | 1 | 3 | 5 | 6 | 2 | 0 | 7 | 2 | 12 |
| $\alpha = 0.95$ | 0 | 2 | 5 | 1 | 4 | 1 | 3 | 4 | 5 | 2 | 0 | 6 | 2 | 12 |

**Table 9**
Summary of the Wilcoxon's test when using the C4.5 classifier.

| | Baseline | RUS | NCL | TL | ENN | OSS | BC | RBt | EEKF | SBC | CBU | fCBUS | CBIS | DBIG-US |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Baseline | – | ∘ | ∘ | | ∘ | | ∘ | ∘ | ∘ | | ∘ | | ∘ | ∘ |
| RUS | • | – | • | • | • | • | | | | • | ∘ | | | ∘ |
| NCL | • | | – | • | | • | ∘ | ∘ | ∘ | | ∘ | | ∘ | ∘ |
| TL | | ∘ | ∘ | – | ∘ | | ∘ | ∘ | ∘ | | ∘ | ∘ | ∘ | ∘ |
| ENN | • | ∘ | | • | – | • | ∘ | ∘ | ∘ | | ∘ | | | ∘ |
| OSS | | ∘ | ∘ | | ∘ | – | ∘ | ∘ | ∘ | | ∘ | | ∘ | ∘ |
| BC | • | | • | • | • | • | – | | | • | ∘ | | | ∘ |
| RBt | • | | • | • | • | • | | – | | • | ∘ | | | ∘ |
| EEKF | • | | • | • | • | • | | | – | • | ∘ | | | ∘ |
| SBC | | ∘ | | | | | ∘ | ∘ | ∘ | – | ∘ | | ∘ | ∘ |
| CBU | • | • | • | • | • | • | • | • | | • | – | • | • | |
| fCBUS | | | | | | | | | | | ∘ | – | | ∘ |
| CBIS | • | | | • | | • | | | | • | ∘ | | – | ∘ |
| DBIG-US | • | • | • | • | • | • | | • | | • | | | • | – |
| $\alpha = 0.90$ | 0 | 6 | 3 | 0 | 3 | 0 | 6 | 6 | 6 | 0 | 12 | 1 | 5 | 12 |
| $\alpha = 0.95$ | 0 | 5 | 3 | 0 | 3 | 0 | 6 | 6 | 6 | 0 | 11 | 0 | 4 | 9 |

**Table 10**
Imbalance ratio of the resampled sets.

| Data set | RUS | NCL | TL | ENN | OSS | BC | RBt | EEKF | SBC | CBU | fCBUS | CBIS | DBIG-US |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| yeast05679v4 | 1.0 | 8.0 | 9.1 | 8.2 | 7.3 | 1.0 | 1.6 | 1.0 | 7.5 | 1.0 | 1.0 | 3.9 | 1.3 |
| vowel0 | 1.0 | 9.9 | 9.9 | 9.9 | 8.3 | 1.0 | 1.5 | 1.0 | 8.9 | 1.0 | 1.0 | 9.0 | 1.2 |
| glass016v2 | 1.0 | 8.7 | 10.1 | 8.4 | 9.6 | 1.0 | 1.5 | 1.0 | 8.9 | 1.0 | 1.0 | 6.4 | 1.0 |
| glass2 | 1.0 | 10.1 | 11.4 | 9.8 | 10.5 | 1.0 | 1.5 | 1.0 | 9.9 | 1.0 | 1.0 | 7.7 | 1.0 |
| shuttle0v4 | 1.0 | 13.8 | 13.8 | 13.8 | 0.3 | 1.0 | 1.5 | 1.0 | 13.8 | 1.0 | 1.0 | 8.9 | 1.1 |
| yeast1v7 | 1.0 | 12.1 | 14.0 | 12.5 | 12.1 | 1.0 | 1.5 | 1.0 | 11.4 | 1.0 | 1.0 | 5.6 | 1.0 |
| glass4 | 1.0 | 14.5 | 15.4 | 14.9 | 4.2 | 1.0 | 1.5 | 1.0 | 15.5 | 1.0 | 1.0 | 10.2 | 1.0 |
| ecoli4 | 1.0 | 15.2 | 15.8 | 15.4 | 10.3 | 1.0 | 1.5 | 1.0 | 14.9 | 1.0 | 1.0 | 8.0 | 1.5 |
| pagBks13v4 | 1.0 | 15.3 | 15.8 | 15.2 | 4.5 | 1.0 | 1.5 | 1.0 | 14.7 | 1.0 | 1.0 | 12.7 | 1.5 |
| glass016v5 | 1.0 | 18.3 | 19.3 | 18.7 | 5.9 | 1.0 | 1.4 | 1.0 | 18.7 | 1.0 | 1.0 | 12.0 | 1.0 |
| shuttle2vs4 | 1.0 | 19.8 | 20.5 | 20.5 | 8.7 | 1.0 | 1.5 | 1.0 | 20.5 | 1.0 | 1.0 | 18.5 | 1.0 |
| yeast1458vs7 | 1.0 | 19.6 | 21.8 | 19.8 | 21.8 | 1.0 | 1.5 | 1.0 | 16.9 | 1.0 | 1.0 | 8.2 | 1.4 |
| glass5 | 1.0 | 21.6 | 22.7 | 22.0 | 7.6 | 1.0 | 1.4 | 1.0 | 21.9 | 1.0 | 1.0 | 15.0 | 1.0 |
| yeast2vs8 | 1.0 | 21.9 | 22.9 | 22.0 | 21.2 | 1.0 | 1.5 | 1.0 | 21.0 | 1.0 | 1.0 | 10.9 | 1.5 |
| flareF | 1.0 | 22.0 | 23.7 | 21.8 | 23.7 | 1.0 | 1.5 | 1.0 | 23.2 | 1.0 | 1.0 | 6.0 | 1.1 |
| yeast4 | 1.0 | 26.4 | 27.8 | 26.6 | 24.7 | 1.0 | 1.5 | 1.0 | 22.1 | 1.0 | 1.0 | 10.5 | 1.4 |
| yeast1289v7 | 1.0 | 28.3 | 30.2 | 28.4 | 30.1 | 1.0 | 1.5 | 1.0 | 29.6 | 1.0 | 1.0 | 11.5 | 1.3 |
| yeast5 | 1.0 | 32.0 | 32.6 | 32.0 | 25.0 | 1.0 | 1.5 | 1.0 | 32.0 | 1.0 | 1.0 | 12.6 | 1.4 |
| ecoli0137v26 | 1.0 | 38.1 | 38.9 | 38.0 | 17.3 | 1.0 | 1.4 | 1.0 | 37.6 | 1.0 | 1.0 | 24.1 | 1.0 |
| abalone17v78910 | 1.0 | 37.5 | 39.1 | 38.2 | 37.3 | 1.0 | 1.5 | 1.0 | 31.0 | 1.0 | 1.0 | 25.6 | 1.3 |
| yeast6 | 1.0 | 40.0 | 41.1 | 39.9 | 35.5 | 1.0 | 1.5 | 1.0 | 40.2 | 1.0 | 1.0 | 15.0 | 1.3 |
| shuttle2v5 | 1.0 | 66.6 | 66.6 | 66.6 | 44.2 | 1.0 | 1.5 | 1.0 | 62.7 | 1.0 | 1.0 | 56.6 | 1.3 |
| kddBfOfw | 1.0 | 73.4 | 73.4 | 73.4 | 0.5 | 1.0 | 1.5 | 1.0 | 71.0 | 1.0 | 1.0 | 66.1 | 1.3 |
| poker89v5 | 1.0 | 79.4 | 81.7 | 80.5 | 80.6 | 1.0 | 1.5 | 1.0 | 69.2 | 1.0 | 1.0 | 26.3 | 1.0 |



(a) 1-NN classifier (b) C4.5 classifier

**Fig. 3.** Geometric mean versus reduction in imbalance ratio.

each class to assess the classification performance and also the reduction in the imbalance ratio to evaluate the rebalancing performance.

Experimental results on 24 two-class imbalanced data sets have allowed to validate the superior performance of the DBIG-US method as compared with some state-of-the-art under-sampling algorithms. The results have revealed that the proposed under-sampling algorithm yields a considerable reduction in size without decreasing the classifier performance.

Despite the encouraging performance obtained with the proposed algorithm, there is still space for further research. An interesting issue

**Table B.11**
Geometric mean results obtained by 1-NN.

| Data set | Baseline | RUS | NCL | TL | ENN | OSS | BC | RBt | EEKF | SBC | CBU | fCBUS | CBIS | DBIG-US |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| yeast05679v4 | 61.2 | 73.5 | **83.0** | 69.0 | 78.9 | 72.2 | 73.5 | 76.7 | 74.3 | 77.1 | 74.5 | 82.1 | 73.5 | 81.9 |
| vowel0 | **100** | 97.2 | **100** | **100** | **100** | **100** | 96.6 | 98.7 | 97.8 | 99.8 | 97.8 | **100** | **100** | 99.9 |
| glass016v2 | 47.0 | 74.4 | 67.0 | 52.7 | 67.9 | 52.9 | 73.5 | 64.1 | 85.2 | 53.1 | 69.6 | 47.1 | 51.7 | **88.4** |
| glass2 | 40.7 | 76.2 | 53.1 | 41.0 | 67.1 | 47.4 | 67.6 | 68.7 | 75.6 | 53.1 | 78.9 | 66.1 | 51.9 | **90.5** |
| shuttle0v4 | 99.6 | 99.6 | 99.6 | 99.6 | 99.6 | **100** | 99.6 | 99.6 | **100** | 99.6 | 99.6 | 98.7 | 99.6 | 99.6 |
| yeast1v7 | 62.3 | 74.8 | 76.8 | 67.6 | 70.4 | 69.9 | 58.1 | 65.9 | 57.2 | 67.2 | 64.5 | 62.3 | 68.4 | **79.3** |
| glass4 | 86.6 | 79.9 | 91.3 | 86.6 | 87.7 | 83.6 | 84.3 | 87.4 | 81.7 | 86.6 | 80.7 | 81.7 | 87.4 | **96.1** |
| ecoli4 | 86.3 | 95.0 | 89.4 | 86.3 | 92.2 | 86.2 | **100** | 92.2 | 92.5 | 91.9 | 82.2 | **100** | 83.4 | 94.9 |
| pagBks13v4 | 98.2 | 98.2 | 98.2 | 98.2 | 98.2 | 98.2 | 96.4 | 96.6 | 87.9 | 99.9 | 94.6 | **100** | **100** | 98.1 |
| glass016v5 | 81.0 | 77.0 | **100** | 80.9 | **100** | 79.3 | 94.3 | 91.0 | 92.6 | 94.0 | 81.7 | **100** | 99.5 | 94.3 |
| shuttle2vs4 | 91.3 | 91.3 | 91.3 | 91.3 | 91.3 | **100** | **100** | 98.2 | **100** | 91.3 | 74.54 | **100** | **100** | 91.3 |
| yeast1458vs7 | 44.1 | 64.5 | 62.7 | 47.9 | 48.0 | 54.2 | 60.0 | 65.1 | 72.1 | 57.0 | 63.0 | 61.2 | 59.8 | **89.2** |
| glass5 | 81.1 | 94.3 | 93.8 | 81.3 | 94.0 | 79.8 | 72.0 | 91.2 | **100** | 93.3 | 77.0 | **100** | 93.9 | 94.3 |
| yeast2vs8 | 77.0 | 59.8 | 80.6 | 77.3 | 80.5 | 77.4 | 72.5 | 75.1 | 77.5 | 80.2 | 64.2 | 52.9 | 73.0 | **87.7** |
| flareF | 30.1 | 81.4 | 65.9 | 33.8 | 72.6 | 26.1 | 68.5 | 77.3 | 85.5 | 26.1 | 77.8 | 81.7 | 74.9 | **94.2** |
| yeast4 | 58.8 | 83.3 | 80.1 | 62.3 | 71.3 | 58.9 | 82.4 | 77.1 | 73.6 | 78.6 | 69.1 | 81.4 | 72.0 | **91.5** |
| yeast1289v7 | 47.6 | 56.6 | 65.4 | 54.2 | 60.4 | 60.0 | 69.9 | 65.0 | 61.5 | 47.6 | 51.4 | 61.2 | 69.6 | **91.5** |
| yeast5 | 82.1 | **100** | 96.3 | 87.6 | 94.1 | 88.7 | 96.5 | 94.6 | 93.1 | 82.1 | 89.6 | 98.3 | 84.4 | 88.3 |
| ecoli0137v26 | 83.7 | 78.3 | 84.4 | 84.1 | 84.4 | 84.2 | 71.4 | 81.9 | **100** | 84.2 | 78.3 | **100** | 83.0 | 65.5 |
| abalone17v78910 | 50.6 | 77.6 | 66.7 | 45.3 | 61.4 | 50.7 | 73.2 | 78.6 | 77.3 | 58.3 | 63.4 | 92.9 | 48.7 | **97.8** |
| yeast6 | 71.1 | 69.9 | 86.1 | 79.0 | 82.7 | 80.6 | 81.4 | 83.7 | **87.1** | 71.2 | 75.9 | 85.5 | 76.6 | 85.7 |
| shuttle2v5 | **100** | 99.0 | **100** | **100** | **100** | **100** | **100** | 99.9 | 99.0 | **100** | **100** | **100** | **100** | **100** |
| kddBfOfw | **100** | **100** | **100** | **100** | **100** | **100** | **100** | **100** | **100** | **100** | **100** | **100** | **100** | 98.4 |
| poker89v5 | 19.9 | 53.1 | 20.0 | 20.0 | 20.0 | 20.0 | 63.5 | 65.1 | 43.3 | 34.5 | 64.5 | 66.1 | 39.8 | **90.0** |
| Avg. Gmean | 70.9 | 81.5 | 81.3 | 72.7 | 80.1 | 73.8 | 81.5 | 83.0 | 83.9 | 76.1 | 78.0 | 84.1 | 78.8 | **91.2** |
| Avg. Rank | 10.69 | 7.54 | 5.67 | 9.65 | 6.5 | 8.81 | 7.5 | 6.96 | 6.21 | 8.4 | 9.21 | 5.54 | 7.75 | **4.58** |

open for future investigation relates to the generalization of DBIG-US for tackling the multi-class imbalance problem, which emerges as a more realistic situation in many practical applications; for instance, the goal for a cancer diagnosis problem expects to categorize the multiple types of tumor rather than only to distinguish between cancerous and normal tissues. Another subject that deserves further research is the multi-label classification of imbalanced data where the classes are not mutually exclusive, which represents a much more complex problem because an instance may belong to more than one class; for example, in the scene classification domain, images may belong to multiple semantic classes (e.g., beach scene, urban scene, nature scene).

Another open line is focused on analyzing the behavior of the two stages of the DBIG-US method separately, which will be further investigated in a future work with synthetic data. On the other hand, since our proposal is very dependent on the distance metric (Euclidean distance), the effect of this can be an interesting point to be studied in the future, especially over data sets with a very high number of features. Furthermore, the heuristic metric to dynamically computing the DBSCAN free parameters, which depends on the data set characteristics, could let further research to check whether or not our heuristic metric is optimal and compare it against other empirical approaches.

## CRediT authorship contribution statement

**A. Guzmán-Ponce:** Conceptualization, Software, Investigation, Writing - review & editing. **J.S. Sánchez:** Conceptualization, Writing - review & editing, Supervision. **R.M. Valdovinos:** Conceptualization, Writing - review & editing, Supervision. **J.R. Marcial-Romero:** Formal analysis, Writing - review & editing.

## Declaration of competing interest

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

## Acknowledgment

This work was partially supported by the Universitat Jaume I, Spain under grant [UJI-B2018-49], the 5046/2020CIC Universidad Autónoma del Estado de México project and the Mexican Science and Technology Council (CONACYT) under scholarship [702275].

## Appendix A. Concepts and notation on graph theory

Fundamental concepts and notations regarding graphs (Bondy & Murty, 1976) that are necessary for a full understanding of the second step in the DBIG-US algorithm are introduced in this appendix. Let $G = (V, E)$ be an undirected simple graph (i.e., finite, loop-less and without parallel edges) with a set of vertices or nodes $V(G)$ and a set of edges $E(G)$, where each edge is a set of two vertices $\{v, u\}$ (order does not matter). The two vertices forming an edge are said to be the endpoints of this edge, and the edge is said to be incident to the vertices.

**Definition 1** (*Subgraph*)**.** $H$ is a subgraph of $G$ if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$.

**Definition 2** (*Induced Subgraph*)**.** An induced subgraph of a graph is another graph, formed from a subset of the vertices of the graph and all of the edges connecting pairs of vertices in that subset. Let $X \in V(G)$ be a set of vertices deleted, then the induced subgraph is denoted by $G - X$; if $Y = V \setminus X$ represents the set of vertices that remain undeleted, the induced subgraph is denoted by $G[Y]$ and referred to as subgraph of $G$ induced by $Y$, where $Y$ is the set of vertices of $G$ and whose set of edges consists of all edges of $G$ that have both ends in $Y$.

**Definition 3** (*Adjacency*)**.** A vertex $v \in V(G)$ is said to be adjacent to another vertex $u \in V(G)$ if there is an edge $\{v, u\} \in E(G)$. A *complete graph* is a simple graph in which any two vertices are adjacent.

**Definition 4** (*Neighborhood*)**.** The neighborhood of a vertex $v \in V(G)$ is a subgraph induced by all vertices that are adjacent to $v$. It can be denoted by $N[v] = \{u \in V(G) | \{v, u\} \in E(G)\}$.

**Definition 5** (*Path*)**.** A path from a vertex $v \in V(G)$ to a vertex $u \in V(G)$ in a graph $G$ is a sequence of vertices $(v_1, v_2, \dots, v_n)$ such that $v = v_1$, $v_n = u$ and each consecutive pair $\{v_i, v_{i+1}\}$ is an edge in $G$. The path is *simple* if all the vertices are distinct. A *cycle* is just a non-empty path where the first vertex $v_1$ is the same as the end vertex $v_n$. A *simple cycle* is a cycle in which all vertices are distinct, except the first and last vertices.

**Table B.12**
Geometric mean results obtained by C4.5.

| Data set | Baseline | RUS | NCL | TL | ENN | OSS | BC | RBt | EEKF | SBC | CBU | fCBUS | CBIS | DBIG-US |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| yeast05679v4 | 65.1 | 70.6 | 70.6 | 65.8 | 74.5 | 64.3 | 68.6 | 77.0 | 70.5 | 74.3 | 85.2 | **85.6** | 72.9 | 76.1 |
| vowel0 | 96.9 | 92.8 | 94.4 | 96.9 | **97.5** | 97.4 | 95.0 | 95.9 | 96.1 | 94.5 | 95.5 | 94.7 | 95.8 | 96.7 |
| glass016v2 | 52.3 | 60.0 | 66.7 | 53.1 | 53.3 | 58.3 | 45.6 | 63.5 | 57.64 | 52.0 | **81.5** | 33.3 | 66.1 | 77.5 |
| glass2 | 53.1 | 90.8 | 57.7 | 33.2 | 58.0 | 52.4 | 61.1 | 65.2 | 79.4 | 58.2 | **94.1** | 52.7 | 66.7 | 84.3 |
| shuttle0v4 | **100** | **100** | 99.9 | **100** | **100** | **100** | **100** | 99.9 | **100** | 99.9 | 99.6 | 97.5 | **100** | **100** |
| yeast1v7 | 54.3 | 64.5 | 57.2 | 65.2 | 62.6 | 54.2 | **81.5** | 67.4 | 58.3 | 62.8 | 78.3 | 69.8 | 56.5 | 63.3 |
| glass4 | 82.2 | 73.0 | 82.1 | 82.2 | 86.6 | 79.3 | **92.3** | 89.1 | 91.7 | 72.4 | 80.7 | 66.7 | 82.3 | 91.9 |
| ecoli4 | 82.9 | 87.5 | 80.1 | 82.9 | 80.1 | 86.0 | 92.5 | 86.3 | **95.0** | 77.1 | 87.5 | 91.3 | 85.8 | 81.5 |
| pagBks13v4 | 96.2 | **100** | 99.8 | 99.8 | 96.1 | 97.4 | 94.5 | 97.5 | 92.0 | 97.1 | 94.6 | 95.7 | 97.5 | 94.3 |
| glass016v5 | 99.4 | 88.2 | 99.7 | 93.7 | 99.7 | 97.1 | 94.3 | 92.5 | 92.6 | 99.7 | **100** | 81.7 | 99.5 | **100** |
| shuttle2vs4 | 90.9 | **100** | 91.3 | 90.9 | 91.3 | **100** | **100** | 99.4 | **100** | 90.9 | **100** | 66.7 | 91.3 | **100** |
| yeast1458vs7 | 0.0 | 59.6 | 0.0 | 0.0 | 0.0 | 0.0 | 44.7 | 53.1 | 60.6 | 25.8 | 74.5 | 61.2 | 77.1 | **94.0** |
| glass5 | 98.8 | **100** | 99.7 | 99.5 | 99.8 | 98.5 | 94.3 | 93.9 | **100** | **100** | **100** | 81.7 | 99.3 | **100** |
| yeast2vs8 | 22.4 | 61.2 | 22.3 | 0.0 | 31.6 | 22.4 | 76.5 | 72.1 | 74.3 | 31.6 | **84.4** | 4.0 | 79.4 | 81.6 |
| flareF | 15.2 | 85.9 | 62.4 | 0.0 | 56.8 | 0.0 | 79.5 | 84.2 | **91.9** | 0.0 | 83.7 | 89.0 | 74.5 | 91.8 |
| yeast4 | 53.9 | 78.3 | 60.7 | 57.4 | 65.4 | 53.9 | 84.1 | 78.2 | 82.9 | 68.4 | 88.2 | 82.9 | 87.3 | **88.6** |
| yeast1289v7 | 48.2 | 53.8 | 44.6 | 44.6 | 44.6 | 44.6 | 69.7 | 67.3 | 54.3 | 40.8 | 79.7 | 61.2 | 51.2 | **84.3** |
| yeast5 | 86.3 | **96.6** | 96.4 | 89.0 | 93.9 | 88.9 | 89.8 | 94.2 | 90.9 | 88.9 | 92.0 | 96.5 | 88.4 | 83.0 |
| ecoli0137v26 | 84.4 | 71.4 | 84.5 | 84.4 | 84.5 | 83.5 | 78.2 | 74.5 | **100** | 84.4 | 85.7 | **100** | 84.3 | 57.1 |
| abalone17v78910 | 34.7 | 75.0 | 43.3 | 34.6 | 39.3 | 32.1 | 74.0 | 75.5 | 74.1 | 57.0 | 75.7 | 91.4 | 32.1 | **92.7** |
| yeast6 | 73.4 | 72.9 | 77.2 | 73.3 | 73.5 | 67.3 | 88.6 | 81.9 | 85.7 | 71.4 | **91.3** | 82.2 | 84.4 | 85.4 |
| shuttle2v5 | **100** | **100** | **100** | **100** | **100** | **100** | **100** | **100** | **100** | **100** | 99.0 | 97.5 | **100** | **100** |
| kddBfOfw | 98.3 | 95.0 | 98.3 | 98.3 | 98.3 | 96.6 | 95.0 | 97.7 | 98.3 | 98.3 | 93.1 | **100** | 98.3 | 95.0 |
| poker89v5 | 0.0 | 47.8 | 0.0 | 0.0 | 0.0 | 0.0 | 42.3 | 44.6 | 56.5 | 0.0 | 68.6 | 69.7 | 44.7 | **70.0** |
| Avg. Gmean | 66.2 | 80.2 | 70.4 | 64.4 | 70.3 | 65.6 | 80.9 | 81.3 | 83.4 | 68.6 | **88.0** | 78.7 | 79.8 | 87.0 |
| Avg. Rank | 9.60 | 6.83 | 8.48 | 9.31 | 7.79 | 9.92 | 6.96 | 6.75 | 5.58 | 9.38 | **4.94** | 7.63 | 6.85 | 4.98 |

**Definition 6** (*Connected Graph*)**.** A graph $G = (V, E)$ is connected if every pair of nodes in $G$ has a path between them. If the graph is not connected, each maximal connected piece is called a *component*.

**Definition 7** (*Weighted Graph*)**.** A weighted graph $G_w = (V, E)$ is a graph where each edge $e \in E(G)$ has been assigned a real number $w(e)$.

**Definition 8** (*Adjacency Matrix*)**.** The adjacency matrix of a weighted graph $G_w = (V, E)$ is a $V \times V$ matrix $M_G = (w_{vu})$ where each element $(v_i, v_j)$ contains the weight $w(e)$ assigned to the edge $e = \{v_i, v_j\}$ or 0 according to whether the vertices $v_i$ and $v_j$ are adjacent or not in the graph.

## Appendix B. Classification results

This appendix provides the classification performance of the 1-NN classifier and the C4.5 decision tree when using the training sets produced by each under-sampling algorithm.

## References

Ali, A., Shamsuddin, S. M., & Ralescu, A. L. (2015). Classification with class imbalance problem: a review. *International Journal of Advances in Soft Computing and its Applications*, *7*(3), 176–204.

Bach, M., Werner, A., Zywiec, J., & Pluskiewicz, W. (2017). The study of under- and over-sampling methods' utility in analysis of highly imbalanced data on osteoporosis. *Information Sciences*, *384*, 174–190.

Barella, V. H., Costa, E. P., & Carvalho, A. C. P. L. F. (2014). ClusterOSS: A new undersampling method for imbalanced learning. In *Proceedings of 3rd Brazilian conference on intelligent systems*. Universidade de São Paulo-USP.

Batista, G., Prati, R., & Monard, M. (2004). A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations Newsletter*, [ISSN: 1931-0145] *6*(1), 20–29.

Bondy, J. A., & Murty, U. S. (1976). *Graph theory with applications (vol. 290)*. New York, NY: Macmillan Press.

Bruni, R., & Bianchi, G. (2020). Website categorization: A formal approach and robustness analysis in the case of e-commerce detection. *Expert Systems with Applications*, [ISSN: 0957-4174] *142*, Article 113001.

Cao, L., & Shen, H. (2019). Imbalanced data classification using improved clustering algorithm and under-sampling method. In *Proceedings of 20th international conference on parallel and distributed computing, applications and technologies* (pp. 358–363).

Chennuru, V. K., & Timmappareddy, S. R. (2017). MahalCUSFilter: A hybrid undersampling method to improve the minority classification rate of imbalanced datasets. In A. Ghosh, R. Pal, & R. Prasath (Eds.), *Proceedings of 5th international conference on mining intelligence and knowledge exploration* (pp. 43–53). Springer International Publishing.

Codetta-Raiteri, D., & Portinale, L. (2015). Dynamic Bayesian networks for fault detection, identification, and recovery in autonomous spacecraft. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *45*(1), 13–24.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Drummond, C., & Holte, R. C. (2003). C4.5, Class imbalance, and Cost Sensitivity: Why Under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II* (vol. 11) (pp. 1–8).

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd international conference on knowledge discovery and data mining* (pp. 226–231). AAAI Press.

Fernández, A., García, S., Galar, M., Prati, R., Krawczyk, B., & Herrera, F. (2018). *Learning from imbalanced data sets (vol. 1)*. Cham, Switzerland: Springer.

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

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, [ISSN: 0020-0255] *180*(10), 2044–2064.

García, V., Marqués, A. I., & Sánchez, J. S. (2019). Exploring the synergetic effects of sample types on the performance of ensembles for credit risk and corporate bankruptcy prediction. *Information Fusion*, [ISSN: 1566-2535] *47*, 88–101.

García, V., Mollineda, R. A., & Sánchez, J. S. (2014). A bias correction function for classification performance assessment in two-class imbalanced problems. *Knowledge-Based Systems*, *59*, 66–74.

García, V., Sánchez, J. S., Marqués, A. I., Florencia, R., & Rivera, G. (2019). Understanding the apparent superiority of over-sampling through an analysis of local information for class-imbalanced data. *Expert Systems with Applications*, [ISSN: 0957-4174] Article 113026.

Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, *73*, 220–239.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, *11*(1), 10–18.

Hart, P. (1968). The condensed nearest neighbor rule. *IEEE Transaction on Information Theory*, *14*(3), 515–516.

Hassan, A. K. I., & Abraham, A. (2016). Modeling insurance fraud detection using imbalanced data classification. In N. Pillay, A. P. Engelbrecht, A. Abraham, M. C. du Plessis, V. Snášel, & A. K. Muda (Eds.), *Advances in nature and biologically inspired computing* (pp. 117–127). Cham: Springer International Publishing.

Jiang, X., Ringwald, M., Blake, J. A., Arighi, C., Zhang, G., & Shatkay, H. (2019). An effective biomedical document classification scheme in support of biocuration: addressing class imbalance. *Database*, 1–10.

Kang, Q., Chen, X., Li, S., & Zhou, M. (2017). A noise-filtered under-sampling scheme for imbalanced classification. *IEEE Transactions on Cybernetics*, *47*(12), 4263–4274.

Kim, M.-J., Kang, D.-K., & Kim, H. B. (2015). Geometric mean based boosting algorithm with over-sampling to resolve data imbalance problem for bankruptcy prediction. *Expert Systems with Applications*, *42*(3), 1074–1082.

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided selection. In *Proceedings of 14th international conference on machine learning* (pp. 179–186).

Kumar, K. A., & Rangan, C. P. (2007). Privacy preserving DBSCAN algorithm for clustering. In R. Alhajj, H. Gao, J. Li, X. Li, & O. R. Zaïane (Eds.), *Lecture notes in computer science*: *vol. 4632*, *Advanced data mining and applications* (pp. 57–68). Berlin, Heidelberg: Springer.

Laurikkala, J. (2001). Improving identification of difficult small classes by balancing class distribution. In *Proceedings of 8th conference on artificial intelligence in medicine* (pp. 63–66). Springer.

Liang, G., & Zhang, C. (2012). An efficient and simple under-sampling technique for imbalanced time series classification. In *Proceedings of the 21st ACM international conference on information and knowledge management* (pp. 2339–2342). ACM.

Lin, W.-C., Tsai, C.-F., Hu, Y.-H., & Jhang, J.-S. (2017). Clustering-based undersampling in class-imbalanced data. *Information Sciences*, *409–410*, 17–26.

Liu, X., Wu, J., & Zhou, Z. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *39*(2), 539–550.

Longadge, M. (2013). Multi-cluster based approach for skewed data in data mining. *IOSR Journal of Computer Engineering*, *12*, 66–73.

López, V., Fernández, A., García, S., Palade, V., & Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, [ISSN: 0020-0255] *250*, 113–141.

Marqués, A. I., García, V., & Sánchez, J. S. (2013). On the suitability of resampling techniques for the class imbalance problem in credit scoring. *The Journal of the Operational Research Society*, *64*(7), 1060–1070.

Ofek, N., Rokach, L., Stern, R., & Shabtai, A. (2017). Fast-CBUS: A fast clustering-based undersampling method for addressing the class imbalance problem. *Neurocomputing*, *243*, 88–102.

Pozzolo, A. D., Caelen, O., & Bontempi, G. (2015). When is undersampling effective in unbalanced classification tasks? In A. Appice, P. P. Rodrigues, V. Santos Costa, C. Soares, J. Gama, & A. Jorge (Eds.), *Machine learning and knowledge discovery in databases* (pp. 200–215). Springer International Publishing.

Rahman, M., & Davis, D. N. (2013). Cluster based under-sampling for unbalanced cardiovascular data. In *Proceedings of the world congress on engineering (vol. 3)* (pp. 1480–1485). Springer.

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2010). RUSboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *40*(1), 185–197.

Sobhani, P., Viktor, H., & Matwin, S. (2015). Learning from imbalanced data using ensemble methods and cluster-based undersampling. In A. Appice, M. Ceci, C. Loglisci, G. Manco, E. Masciari, & Z. W. Ras (Eds.), *Lecture notes in computer science*: *vol. 8983*, *New frontiers in mining complex patterns* (pp. 69–83). Springer International Publishing.

Suthar, N., Indr, P., & Vinit, P. (2013). A technical survey on DBSCAN clustering algorithm. *International Journal of Scientific & Engineering Research*, *4*, 1775–1781.

Thabtah, F., Hammoud, S., Kamalov, F., & Gonsalves, A. (2020). Data imbalance in classification: Experimental evaluation. *Information Sciences*, [ISSN: 0020-0255] *513*, 429–441.

Tomek, I. (1976). An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-6*(6), 448–452.

Tsai, C.-F., Lin, W.-C., Hu, Y.-H., & Yao, G.-T. (2019). Under-sampling class imbalanced datasets by combining clustering analysis and instance selection. *Information Sciences*, *477*, 47–54.

Wang, Q., Zhou, Y., Zhang, W., Tang, Z., & Chen, X. (2020). Adaptive sampling using self-paced learning for imbalanced cancer data pre-diagnosis. *Expert Systems with Applications*, [ISSN: 0957-4174] *152*, Article 113334.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-2*(3), 408–421.

Yang, J., Xie, G., & Yang, Y. (2020). An improved ensemble fusion autoencoder model for fault diagnosis from imbalanced and incomplete data. *Control Engineering Practice*, [ISSN: 0967-0661] *98*, Article 104358.

Yen, S.-J., & Lee, Y.-S. (2006). Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. In *Proceedings of international conference on intelligent computing* (pp. 731–740). Springer.

Yen, S.-J., & Lee, Y.-S. (2009). Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications*, *36*(3), 5718–5727, Part 1.

Yoon, K., & Kwek, S. (2005). An unsupervised learning approach to resolving the data imbalanced issue in supervised learning problems in functional genomics. In *Proceedings of 5th international conference on hybrid intelligent systems* (pp. 303–308). IEEE.

Zhu, R., Guo, Y., & Xue, J. H. (2020). Adjusting the imbalance ratio by the dimensionality of imbalanced data. *Pattern Recognition Letters*, [ISSN: 0167-8655] *133*, 217–223.

Zhu, H., Liu, G., Zhou, M., Xie, Y., Abusorrah, A., & Kang, Q. (2020). Optimizing Weighted Extreme Learning Machines for imbalanced classification and application to credit card fraud detection. *Neurocomputing*, [ISSN: 0925-2312] *407*, 50–62.