Applied Soft Computing 108 (2021) 107447

Contents lists available at ScienceDirect

Applied Soft Computing

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

Experimental evaluation of ensemble classifiers for imbalance in Big Data

Mario Juez-Gil *, Álvar Arnaiz-González, Juan J. Rodríguez, César García-Osorio

Escuela Politécnica Superior, University of Burgos, 09006 Burgos, Spain

ARTICLE INFO

Article history:
Received 9 October 2020
Received in revised form 21 February 2021
Accepted 13 April 2021
Available online 7 May 2021

Keywords:
Unbalance
Imbalance
Ensemble
Resampling
Big Data
Spark

ABSTRACT

Datasets are growing in size and complexity at a pace never seen before, forming ever larger datasets known as Big Data. A common problem for classification, especially in Big Data, is that the numerous examples of the different classes might not be balanced. Some decades ago, imbalanced classification was therefore introduced, to correct the tendency of classifiers that show bias in favor of the majority class and that ignore the minority one. To date, although the number of imbalanced classification methods have increased, they continue to focus on normal-sized datasets and not on the new reality of Big Data. In this paper, in-depth experimentation with ensemble classifiers is conducted in the context of imbalanced Big Data classification, using two popular ensemble families (Bagging and Boosting) and different resampling methods. All the experimentation was launched in Spark clusters, comparing ensemble performance and execution times with statistical test results, including the newest ones based on the Bayesian approach. One very interesting conclusion from the study was that simpler methods applied to unbalanced datasets in the context of Big Data provided better results than complex methods. The additional complexity of some of the sophisticated methods, which appear necessary to process and to reduce imbalance in normal-sized datasets were not effective for imbalanced Big Data.

1. Introduction

In computing science, a new term has emerged to describe the sheer size of datasets, which are rapidly expanding throughout the world: Big Data. The main characteristics of Big Data were initially established by Laney [1] around the 3 Vs: Volume, Velocity and Variety. Additional Vs have since been added such as Value [2] and Veracity [3].[1] Big Data can essentially be defined at the intersection between these concepts: Volume (large datasets), Velocity (data-processing speeds must respond to data-generation speeds), Variety (different forms of data), Veracity (must be resilient under uncertain or imprecise data) and Value (usefulness) [4]. Real-world classification problems are not usually balanced; *i.e.*, the number of instances that belong to each class is unevenly distributed [5]. These classification problems, known as imbalanced or unbalanced learning, have received a lot of attention over recent years [6–8]. The applications of imbalanced learning are broad and diverse, because balanced datasets in real-life are unfortunately rare to find. Moreover, the class in which we are usually interested is the underrepresented one [9]. Some applications of imbalanced learning include activity recognition [10], behavior analysis [11], industrial systems monitoring [12,13], video mining [14] and cancer malignancy grading [15], among others. A general classifier applied with no strategy to process imbalanced datasets will tend to ignore the minority class and will therefore almost inevitably misclassify it. Imbalanced classification is frequent in standard classification, but it is crucial within Big Data environments [8,16,17]. Several alternative approaches for processing imbalanced data have been proposed and can be grouped into four categories [18][2]:

- Algorithm-level: the algorithms are internally modified to process the imbalance, *i.e.*, the algorithms are biased towards focusing on minority class instances.
- Data-level: instead of changing the algorithms, the idea is to resample or to rebalance the class distribution within the input dataset.
- Cost-sensitive: the algorithms incorporate different misclassification costs for the different classes within the dataset.
- Classifier ensembles: the capability of ensembles to improve on the results of difficult classification tasks has been demonstrated. A common ensemble construction method

* Corresponding author.
E-mail addresses: mariojg@ubu.es (M. Juez-Gil), alvarag@ubu.es (Á. Arnaiz-González), jjrodriguez@ubu.es (J.J. Rodríguez), cgosorio@ubu.es (C. García-Osorio).

[1] Despite the fact that some authors have identified other Vs within the Big Data equation, the five Vs presented in this paper are by far the most representative.

[2] As pointed out in [7], these categories are not mutually exclusive.

https://doi.org/10.1016/j.asoc.2021.107447

for imbalanced learning is at the data-level, by training each base classifier with a pre-processed dataset. Another approach is the use of heterogeneous ensembles [19].

The decision to use one or another approach is highly problem dependent, nevertheless algorithm-level approaches and cost-sensitive approaches are, according to [18], more data dependent, while the other two methods are more versatile. Data-level is currently the most popular approach to process imbalanced data [20]. The present paper is broadly focused on data-level and classifier ensemble approaches and specifically on how the resampling techniques help ensemble learning to address imbalanced classification in Big Data problems.

The topic of imbalanced classification in Big Data is still at an early stage [8,21]. Two gaps have been identified: the few ensemble methods designed for Big Data problems [22] and perhaps even fewer for processing imbalance within Big Data [17]. The aim of this work is therefore to conduct an experimental evaluation of some of the most popular ensembles in the context of imbalanced Big Data classification: Bagging and Boosting using various resampling techniques. The Apache Spark framework, a widely used Big Data platform, will be used for the experimentation. Since proposing new implementations for Big Data is beyond the scope of this study, the experiments will only be run on currently available (or easily implementable) ensemble and resampling methods in Apache Spark.

The rest of the paper will be structured as follows. In Section 2, research into ensembles applied to the imbalance problem will be explained. In Section 3, the state-of-the-art of data pre-processing techniques for Big Data will be described. In Section 4, details of the experimentation will be reported, followed by an explanation of the results that will be given in Section 5. Finally, the main conclusions and future research lines will be discussed in Section 6.

2. Ensemble learning for imbalanced problems

As is well-known, ensemble learning is based on the construction and the subsequent combination of several classifiers to obtain a new classifier that can outperform the base classifiers [18]. Desirable properties of the classifiers that make up ensembles are accuracy, instability and diversity, among others. Instability is a useful property of the base classifiers of an ensemble (*i.e.*, small variations within an input dataset can generate significant changes within the classifier). Diversity is essential, because if all the base classifiers were to predict the same class, then there might be little point in building an ensemble. There are multiple methods to force diversity, many of which are based on either Bagging [23] (which resamples the dataset) or Boosting [24] (which assigns a weight to the instances depending on the difficulty of their classification). However, there are some other methods that involve alternative techniques [25–29].

Pre-processing techniques that balance the class proportions can be applied in a straightforward manner within an ensemble and will usually either reduce the size of the majority class, increase the size of the minority class, or even achieve both at the same time [7]. Although there are several techniques, we will only summarize those that are generally used to deal with imbalance in ensemble learning:

- Random UnderSampling (RUS): consists of randomly removing examples of the majority class. The number of examples removed reduces the imbalance ratio, and it can balance the dataset, or even unbalance it in the opposite direction.
- Random OverSampling (ROS): consists of randomly replicating examples of the minority class. As with the previous case, the number of examples generated reduces the imbalance ratio.
- Synthetic Minority Oversampling TEchnique (SMOTE) [30]: generates synthetic instances of the minority class by interpolation of two original minority instances. It applies *k*-nearest neighbors to ensure that the instances selected for the interpolation are close to each other.
- Random OverSampling Examples (ROSE) [31]: generates a new synthetic dataset of a certain size and a certain imbalance ratio. A typical strategy is to create a balanced dataset of the same size as the input dataset. ROSE generates new artificial examples according to a smoothed bootstrap approach.
- Random Balance (RB) [32]: as the balancing technique, either the generation or the elimination of instances (more suitable for a dataset), and the optimal imbalance ratio will *a priori* be unknown, a generative method (SMOTE) and a reduction of instances method (RUS) are randomly combined in each classifier by the RB ensemble method. The imbalance ratio of the datasets that are generated is also random, all of which yields additional diversity from which the ensemble may also benefit.

These pre-processing techniques can either be performed once at the beginning (before training the ensemble classifier), or before training each single base classifier of the ensemble. For example, the use of undersampling in each base classifier of a bagging ensemble, is known as under-bagging, in the same way that over-bagging consists in using oversampling within bagging [18].

During the last two decades, different combinations of ensembles and pre-processing techniques have been proposed to improve classification performance [18], both for Bagging and Boosting [33] families, as well as hybrid solutions [34].

3. Imbalanced data pre-processing for big data

Despite the intense research over past decades in imbalanced learning for normal-sized datasets, the imbalance problem in relation to computational scalability has yet to be thoroughly explored [35]. The first comparison of several resampling methods in both Hadoop and Spark was presented in [8]. For a complete review of these methods, we would recommend the following papers [16,17].

Simple sampling techniques, such as Random OverSampling (ROS) and Random UnderSampling (RUS) were among the first attempts to deal with imbalance in Big Data classification [36]. The effect of RUS on Big Data with simulated class imbalance datasets was likewise studied in [37].

SMOTE is another popular algorithm to deal with imbalanced datasets, the straightforward implementation and sound performance of which has increased its popularity and led to a proliferation of SMOTE variants, that number over 85 [38] today. The Big Data version of SMOTE (SMOTE-BD) was recently presented in [39]. A Big Data implementation of a SMOTE algorithm based on the Neighborhood Rough Set Model [40] (NRSBoundary-SMOTE) was presented in [41].

The ROSEFW-RF [42] algorithm (Random OverSampling and Evolutionary Feature Weighting for Random Forest) was the winning algorithm in the ECBDL'14 Big Data competition. This algorithm balances the classes using ROS and identifies the most relevant features through an evolutionary feature-selection process, before building a Random Forest classifier. The algorithm demonstrated its strengths winning the competition.

Despite the fact that most studies are based on ensemble and simple sampling techniques, more sophisticated methods have been presented, such as evolutionary undersampling [43], among others.

Table 1
Sampling techniques used in the experimental study and their configuration.

Abbr.	Method	Details and parameters
RUS	Random Undersampling	Undersampling without replacement until 50% of the data belongs to minority class.
ROS	Random Oversampling	Oversampling until 50% of the data belongs to minority class.
SMOTE	SMOTE (approximated)	$K = 5$, Oversampling until 50% of the data belongs to minority class.
ROSE	ROSE	*shrinkFactor* $= 1$, Generate n synthetic examples with 50% probability of belonging to any class. (n is the size of the original dataset)
RB	Random Balance	A random imbalance ratio is applied for each base classifier.

Table 2
Ensemble classifiers used in the experimental study and their configuration.

Abbr.	Method	Details and parameters
BAG	Bagging	*numTrees* $= 100$, *maxDepth* $= 5$, *subsamplingRate* $= 1.0$, *featureSubsetStrategy* $=$ *all*
RANF	Random Forest	*numTrees* $= 100$, *maxDepth* $= 5$, *subsamplingRate* $= 1.0$, *featureSubsetStrategy* $=$ *auto*
GBT	Gradient Boosting Trees	*maxIter* $= 100$, *maxDepth* $= 5$

The lack of methods applicable to imbalanced learning in Big Data frameworks poses difficulties when extracting knowledge from large datasets with unevenly distributed classes. The scarcity of methods becomes even more noticeable when it is compared with the large number of methods for normal-sized datasets [17]. Furthermore, some of these methods were implemented only for Hadoop before Spark became more popular, and thus, their comparison is even more challenging. To the best of our knowledge, no comprehensive experimentation has been conducted to date, to compare resampling-based ensemble methods with Big Data imbalance classification, and we consider that there is a need for that kind of experimental evaluation. Knowing whether resampling techniques can and to what extent they can benefit imbalanced Big Data classification is essential to accomplish meaningful and successful future research.

4. Experimental set-up

The experimental set-up was organized into two groups, in order to determine the effects of the balancing/resampling strategies on the performance of the ensembles:

- Dataset resampling and then training: the dataset was balanced once at the beginning (following one of the strategies) and an ensemble classifier was trained using the resampled dataset.
- Resampling within the ensemble: the ensemble performs a resampling strategy before the training of each base classifier (*i.e.*, there are as many resamples as there are base classifiers).

4.1. Methods

Five popular sampling techniques were tested in the experimentation: RUS, ROS, SMOTE, ROSE, and RB. All the algorithms were implemented in Scala and executed within the Apache Spark framework. The implementation of RUS and ROS was straightforward using the Spark API. As mentioned above, SMOTE-BD [39] is an implementation of the SMOTE algorithm for Spark, however, we used our own implementation of SMOTE[3] based on Fang's approximated KNN,[4] that uses a hybrid spill-tree approach [44], to achieve high accuracy and search efficiency. A parallel version of ROSE was implemented under the Spark framework following the existing R implementation [45]. Finally, the RB implementation is a variant of the original algorithm specifically adapted to Big Data.[5] Table 1 lists the sampling methods that were tested.

The performance of the sampling techniques was compared using three different ensemble classifiers available for Spark: Bagging (BAG), Random Forest (RANF), and Gradient Boosting Trees (GBT), thus covering the Bagging and Boosting families of ensembles.

In the first group of experiments (resampling before training), the ensembles were trained using the balanced datasets obtained after applying the sampling techniques: RUS, ROS, SMOTE, and ROSE. In the second group of experiments (resampling within the ensemble), specific implementations of Bagging and Gradient Boosting Trees were developed for processing the imbalance problem. In these implementations, a new sampling was performed for each base classifier in the ensemble, for which the following methods were used: RUS, ROS and RB.

So that the comparison of the ensemble algorithms was on equal terms, all the ensembles were composed of 100 trees, each with a depth of 5. The details of the algorithms and their parameters are listed in Table 2.

The methods without resampling and therefore with no established method of processing imbalance were also included in the experiments, as a baseline performance reference, these methods are referred to in this paper as the Gini variant. Traditionally, classification and regression trees (CART) have used the Gini index as a split criterion. Nevertheless, when datasets are imbalanced, the Gini index is biased in favor of the majority class [46]. For this reason, the weighted Gini (WGini) index could be beneficial for the application of trees to imbalanced learning, thus it was also included in this experimental evaluation.[6]

To sum up, different ensemble classifiers were evaluated: BAG, RANF, and GBT. For each ensemble, two impurity indexes were used as a baseline (Gini and WGini) and some popular resampling techniques were tested: ROS, RUS, SMOTE, ROSE, and RB. The use of weighted Gini with the resampling techniques was not explored because, as we resampled until both classes were totally balanced, the result would be equivalent to the use of the default Gini index.

[3] Approx-smote on GitHub: https://github.com/mjuez/approx-smote.

[4] Saurfang spark-knn on GitHub: https://github.com/saurfang/spark-knn.

[5] To speed up its execution with Big Data, instead of the SMOTE and RUS methods in the original algorithm, the new one used a combination of ROS and RUS.

[6] Due to the fact that the weighted Gini index is not available on the Spark trees, the instances were weighted according to the imbalance ratio of their class, as proposed by Chen et al. [47].

Table 3
Main characteristics of the datasets used in the experiments: number of instances, number of features, number of classes of the minority and majority classes, imbalance ratio, and dataset size in libsvm [48] format.

Dataset	# instances	# attributes	# maj/min	IR	Size (GB)
COVTYPE 1 vs 4	229 207	54	211 840/17 367	12.20	0.02
COVTYPE 1 vs 3	232 350	54	211 840/20 510	10.33	0.02
COVTYPE 1 vs 2	247 594	54	211 840/35 754	5.92	0.02
COVTYPE 0 vs 4	300 668	54	283 301/17 367	16.31	0.03
COVTYPE 0 vs 3	303 811	54	283 301/20 510	13.81	0.03
COVTYPE 0 vs 2	319 055	54	283 301/35 754	7.92	0.03
ECBDL'14 1M	999 716	893	978 128/21 588	45.31	15.70
SUSY IR16	2 881 796	18	2 712 173/169 623	15.99	1.04
SUSY IR4	3 389 320	18	2 712 173/677 147	4.00	1.23
KDDCUP dos vs r2l	3 884 496	119	3 883 370/1 126	3 448.82	0.50
KDDCUP dos vs normal	4 856 151	119	3 883 370/972 781	3.99	0.62
HEPMASS IR16	5 578 586	28	5 250 124/328 462	15.98	3.20
HIGGS IR16	6 194 093	28	5 829 123/364 970	15.97	3.26
HEPMASS IR4	6 561 364	28	5 250 124/1 311 240	4.00	3.77
HIGGS IR4	7 284 166	28	5 829 123/1 455 043	4.00	3.94
ECBDL'14 10M	9 998 491	893	9 783 328/215 163	45.47	45.80

4.2. Experimental framework

Six popular classification datasets for Big Data were used in the experiments.[7] Two standard techniques were used to generate the imbalanced datasets [39]. Several imbalanced datasets were generated from the multi-class datasets (*i.e.*, covtype and kddcup), by selecting the majority class and one of the minority classes. Since the binary datasets (*i.e.*, susy, higgs, and hepmass) are actually almost evenly balanced, the imbalance was forced by subsampling the class with fewer instances using two different imbalance ratios: 4 and 16. The ECBDL'14 is already unbalanced, so it was used as is without any transformation. Only two subsamples (approximately 1 million and 10 million instances) with the original imbalance ratio were processed, because of the huge size of this dataset and budgetary limitations on the computing time that may be requested on Google Cloud clusters. Table 3 summarizes the resulting 16 datasets. All the datasets were stored in libsvm [48] format. As the implementation of some algorithms, such as SMOTE or ROSE, cannot process nominal features, all nominal features were binarized using one-hot encoding.

The experiments were performed using 5 repetitions of a 2-fold cross-validation. A random seed value was set at 46 to enable experimental repeatability.

For evaluating and comparing statistical differences in performance, average ranks and statistical tests were used. Average ranks were computed by assigning, for each dataset, one to the best classifier, two to the second, and so on. When there was a tie between several classifiers, their average rankings were assigned to each one. The final value assigned to each method was the mean of its rankings across all datasets. The Friedman test, followed by the Hochberg $1\times N$ (one vs. all) and the Nemenyi $N\times N$ (all vs. all) *post hoc* procedures [50] were used to evaluate any statistical difference in the results.

Bayesian analysis [51] (*baycomp*[8] library was used) was conducted using Bayesian hierarchical sign tests, which separately account for the results of all folds and repetitions, for comparative purposes. The number of samples for all Bayesian comparisons was set at 50 000. A common graphical representation of this type of analysis is a ternary plot [52] where the region of practical equivalence (ROPE) appears on the top corner, and on the right and left corners are the regions of the methods under comparison. The ROPE was set to 0.05, which means that two algorithms with a difference in performance of less than 0.05 will be considered equivalent.

The experiments were executed in cloud-based clusters provided by the Google Cloud Platform. The cloud cluster was composed of a total of twenty-eight nodes (one master node and twenty-seven worker nodes). All nodes were of the n1-highmem-8 type, which had 8 virtual CPUs, and 52 GB of RAM. Hence, the cluster size was 224 vCPUs and occupied 1 456 GB of RAM. At that point in time, the vCPUs of n1-highmem nodes could be of the four following types: Intel Xeon (Skylake), Intel Xeon E5 (Sandy Bridge), Intel Xeon E5 v2 (Ivy Bridge), Intel Xeon E5 v3 (Haswell), or Intel Xeon E5 v4 (Broadwell E5). The Apache Spark environment was provided by Google Dataproc software, version 1.4, running on Debian 9, with Apache Hadoop 2.9.2, and Apache Spark 2.4.5.[9] Google Cloud Storage, as a distributed file system, was used for storing the datasets and the experimental results.

4.3. Performance metrics

Standard accuracy metrics tend to focus on all target classes equally, which is a problem for the minority ones [53]. For this reason, the use of a single metric in imbalance classification is not recommended, as it is commonly preferred to use several dedicated metrics to contrast interpretations.

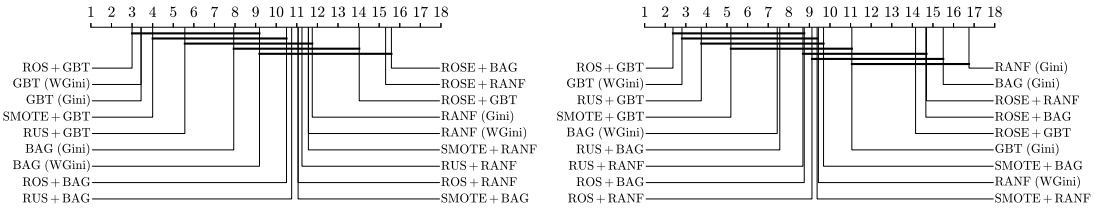
In this study, the four distinct metrics under consideration were: F_1 Score (F_1-score), Matthews Correlation Coefficient (MCC), Geometric Mean (G-mean), and Area Under the Curve (AUC). All these metrics were defined using different values given in the confusion matrix. If we consider a problem with two classes: a class of interest, the "positive" class (+), and another usually much larger class, the "negative" (−) one; then the mistakes (False Positives and False Negatives) and successes (True Positives and True Negatives) of a classifier can be sorted within a confusion matrix, as follows:

		Predicted labels		
		+	−	total
Actual labels	+	TP	FN	P
	−	FP	TN	N
	total	$\hat{P}$	$\hat{N}$	

[7] The covtype, susy, higgs, hepmass, and kddcup datasets are available from the UCI Machine Learning repository [49] https://archive.ics.uci.edu/ml/index.php. The ECBDL14 dataset is available from the Evolutionary Computation for Big Data and Big Learning Workshop competition website http://cruncher.ico2s.org/bdcomp/.

[8] The *baycomp* library is publicly available at https://baycomp.readthedocs.io/en/latest/.

[9] The experiments of the Weighted Gini (WGini) was launched within a Apache Spark 3.0.1 cluster, since the instance weighting on the trees training were not supported in previous versions.



(a) MCC (b) AUC

Fig. 1. A Friedman–Nemenyi test comparison of the different ensemble classifiers performing resampling before training, according to MCC (a) and AUC (b) metrics. The methods connected with a thick horizontal line show no significant differences between each other at a level of $\alpha = 0.05$. For many methods there is a close statistical equivalence.

The F_1-score is the harmonic mean of *precision* $\left(\frac{TP}{TP+FP}\right)$ and *recall* $\left(\frac{TP}{TP+FN}\right)$:

$$\begin{aligned} F_1\text{-score} &= \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} \\ &= 2 \times \frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \end{aligned} \tag{1}$$

The MCC is a metric strongly advised for imbalance classification [54]:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{\hat{P} \times P \times \hat{N} \times N}} \tag{2}$$

The G-mean is the geometric mean of *sensitivity* and *specificity*:

$$\text{G-mean} = \sqrt{\text{sensitivity} \times \text{specificity}} = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \tag{3}$$

The domains of all the metrics, except MCC, were between 0 and 1, where 1 was the best value. Furthermore, MCC is a metric that expresses the correlation between the actual and the predicted class, with values ranging between −1 (total disagreement) and +1 (perfect agreement).

The definition of the AUC is slightly less straightforward. When the result of a binary classifier is the probability that the instance belongs to the class of interest, the final assignment of the class will ultimately depend on a threshold value from which we consider that the probability is sufficient to assign the class of interest. For each threshold value, we would have different results of the classifier, and therefore different *TP* and *FP* rate values (*TPR* and *FPR*). If we consider the entire range of values for the threshold, from zero to one, we can represent the results obtained for each threshold as points with the coordinates (*TPR*, *FPR*). The curve obtained by considering all these points is called the Receiver Operating Characteristic curve, or ROC curve. If we now consider the area under the curve (AUC), its value serves as an indicator of classifier performance for all possible thresholds. A good classifier will have a value close to 1 and a random (non-informative) classifier will have a value close to 0.5.

For the sake of simplicity, only the MCC and the AUC metrics are reported in the following sections. The reason is because, for this experimental evaluation, we have found that the four metrics can be paired in such a way that both metrics of each pair lead to almost the same interpretations. The two groups of metrics are F_1-score and MCC, and G-mean and AUC. Nonetheless the results of the four metrics may be found in the Appendix A.

5. Results and discussion

As previously stated, the experimentation was divided into two groups: resampling before training (see Section 5.1) and resampling within the ensemble (see Section 5.2). A comparison of both approaches may be found in Section 5.3; a comparison of their execution times is presented in Section 5.4; and, finally, all the results are discussed in Section 5.5.

5.1. Resampling before training ensemble

This subsection shows the performance results of the ensemble methods trained after obtaining a dataset by resampling the original dataset. These experiments correspond with the data-level approach.

Table 4.a details the results of the experiments for MCC, and Table 4.b for AUC. The best results of the datasets from each ensemble classifier group (*i.e.*, BAG, RANF, and GBT) are shown within gray boxes, and the best overall results are highlighted within black boxes. The blueness intensity of each cell is used for highlighting the results, the higher the metric, the darker the blue. A clear superiority of GBT over BAG and RANF ensemble classifiers was shown. The impact of the RUS, ROS, and SMOTE pre-processing methods gave quite similar performances at a glance. Moreover the classifier trained with the original dataset using the weighted Gini index, could also be included in that similar-performing group of methods (*i.e.*, row-wise blueness intensity is almost the same for the four variants: WGini, RUS, ROS, and SMOTE). ROSE by contrast, generally achieved worse performance. Focusing on each metric separately, the results for MCC tended to reveal that it is better not to pre-process at all and to train the classifier using the Gini index; on the other hand, AUC highlighted a better performance when pre-processing techniques were used or, at least, when the weighted Gini index was used. Nevertheless, it is worth noting that the results of the classifiers trained without taking the imbalance into account (*i.e.*, Gini variants) were, for some datasets, totally unacceptable, such as for example ECBDL or HIGGS, while the other alternatives offered good overall results.

Table 5 shows the average ranks computed for each metric, the dashed line represents the limit below which the methods differ statistically from the best one at a significance level of 95%. The best method was ROS+GBT for both metrics, followed by GBT (WGini). The AUC metric revealed that some resampling, or at least the use of weighted Gini, contributed to overcoming the imbalance, because regular ensembles (*i.e.*, Gini variants) were ranked extremely low. The methods that showed the worst results were, in general, RANF (Gini) and all the variants involving ROSE.

The results of the Friedman–Nemenyi test can be seen in Fig. 1. It shows all groups of methods that performed equivalently, providing therefore, additional information to the average rankings discussed above. Also, the number of groups (*i.e.*, from five to seven) could be considered high. Fig. 1.a shows that SMOTE+GBT, ROS+GBT, and GBT (WGini) performed better than over half of the evaluated methods, which include all RANF variants and three

Table 4
Experimental results performing resampling before the ensemble training for MCC (a) and AUC (b). The best results for each classifier appear within gray boxes, while the best results overall are within black boxes. GBT showed the best performance for both metrics. For MCC, Gini variants, specially the GBT one, showed the best performance for the majority of the datasets. WGini, RUS, ROS, and SMOTE were all preferable options for AUC, depending on the ensemble in use.

(a) MCC

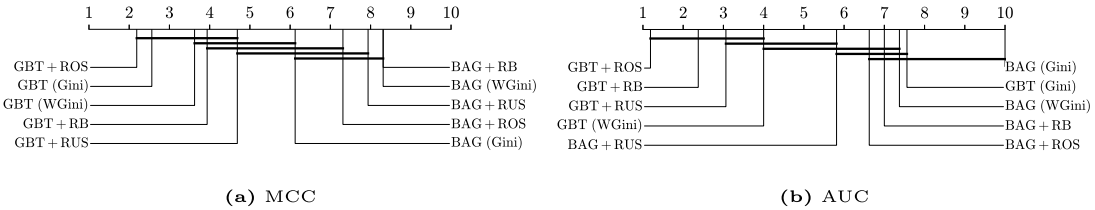
Dataset	BAG						RANF						GBT					
	Gini	WGini	RUS	ROS	SMOTE	ROSE	Gini	WGini	RUS	ROS	SMOTE	ROSE	Gini	WGini	RUS	ROS	SMOTE	ROSE
COVTYPE 1vs4	0.9522	0.9438	0.9422	0.9314	0.9392	0.8976	0.9206	0.9227	0.9229	0.9235	0.9244	0.5602	0.9906	0.9776	0.9686	0.9801	0.9791	0.8963
COVTYPE 1vs3	0.6753	0.5789	0.5696	0.5655	0.5592	0.0043	0.2925	0.5263	0.5186	0.5173	0.5210	0.0000	0.8548	0.7897	0.7675	0.7917	0.7947	0.0043
COVTYPE 1vs2	0.9801	0.9756	0.9734	0.9745	0.9738	0.9447	0.9759	0.9663	0.9680	0.9689	0.9665	0.6207	0.9975	0.9938	0.9930	0.9959	0.9958	0.9445
COVTYPE 0vs4	0.7408	0.6848	0.6741	0.6739	0.6803	0.2409	0.6734	0.6577	0.6580	0.6572	0.6640	0.2405	0.9051	0.8036	0.7822	0.8095	0.8108	0.2409
COVTYPE 0vs3	0.8993	0.7829	0.7870	0.7762	0.7763	0.5864	0.8184	0.7471	0.7514	0.7504	0.7413	0.0835	0.9751	0.9463	0.9218	0.9493	0.9487	0.5868
COVTYPE 0vs2	0.8852	0.8033	0.7995	0.8008	0.8008	0.3267	0.7828	0.7987	0.7921	0.7905	0.7961	0.3325	0.9317	0.8858	0.8816	0.8878	0.8921	0.3267
ECBDL'14 1M	0.0000	0.1275	0.1269	0.1226	0.1203	0.0883	0.0000	0.1353	0.1340	0.1344	0.1282	0.1180	0.0602	0.1555	0.1405	0.1560	0.1169	0.1166
SUSY IR16	0.4385	0.3061	0.3068	0.3002	0.2917	0.3334	0.4284	0.3091	0.3123	0.3117	0.3017	0.3381	0.4970	0.3537	0.3519	0.3541	0.3445	0.4524
SUSY IR4	0.5345	0.4685	0.4695	0.4665	0.4604	0.4830	0.5174	0.4752	0.4793	0.4775	0.4705	0.4884	0.5796	0.5295	0.5281	0.5287	0.5221	0.5569
KDDCUP dos vs r2l	0.9757	0.9596	0.4275	0.8650	0.7890	0.3895	0.9508	0.8245	0.5325	0.8818	0.8316	0.5442	0.9955	0.9885	0.4128	0.9938	0.9956	0.3936
KDDCUP dos vs nor.	0.9987	0.9991	0.9989	0.9993	0.9993	0.9952	0.9980	0.9979	0.9978	0.9978	0.9977	0.9946	0.9999	0.9999	0.9998	0.9999	0.9999	0.9954
HEPMASS IR16	0.6565	0.4319	0.4190	0.4353	0.4562	0.3364	0.4255	0.3428	0.3431	0.3433	0.3529	0.3244	0.6706	0.4742	0.4714	0.4737	0.5406	0.3121
HIGGS IR16	0.1158	0.1597	0.1591	0.1569	0.1497	0.1168	0.0000	0.1750	0.1761	0.1751	0.1746	0.1203	0.1944	0.2306	0.2289	0.2301	0.2070	0.1691
HEPMASS IR4	0.6920	0.6153	0.6141	0.6158	0.5988	0.5253	0.6353	0.5360	0.5369	0.5363	0.5426	0.5159	0.7188	0.6636	0.6628	0.6630	0.6892	0.4917
HIGGS IR4	0.2004	0.2663	0.2654	0.2644	0.2541	0.1923	0.0843	0.2883	0.2900	0.2895	0.2956	0.1976	0.3440	0.3743	0.3723	0.3735	0.3496	0.2606
ECBDL'14 10M	0.0000	0.1252	0.1246	0.1250	0.1256	0.0922	0.0000	0.1348	0.1346	0.1351	0.1275	0.1213	0.0717	0.1620	0.1583	0.1617	0.1374	0.1210

(b) AUC

Dataset	BAG						RANF						GBT					
	Gini	WGini	RUS	ROS	SMOTE	ROSE	Gini	WGini	RUS	ROS	SMOTE	ROSE	Gini	WGini	RUS	ROS	SMOTE	ROSE
COVTYPE 1vs4	0.9694	0.9933	0.9931	0.9924	0.9929	0.9848	0.9313	0.9892	0.9901	0.9900	0.9901	0.9286	0.9941	0.9973	0.9971	0.9980	0.9977	0.9847
COVTYPE 1vs3	0.7939	0.8902	0.8898	0.8865	0.8876	0.5001	0.5474	0.8830	0.8816	0.8807	0.8821	0.5000	0.9055	0.9629	0.9605	0.9630	0.9628	0.5001
COVTYPE 1vs2	0.9930	0.9958	0.9955	0.9954	0.9953	0.9877	0.9822	0.9916	0.9919	0.9926	0.9924	0.9064	0.9988	0.9989	0.9989	0.9992	0.9992	0.9877
COVTYPE 0vs4	0.8366	0.9595	0.9588	0.9577	0.9577	0.7580	0.7802	0.9473	0.9484	0.9472	0.9475	0.7576	0.9396	0.9806	0.9786	0.9810	0.9803	0.7580
COVTYPE 0vs3	0.9221	0.9750	0.9749	0.9747	0.9744	0.9386	0.8545	0.9627	0.9621	0.9640	0.9639	0.5471	0.9828	0.9939	0.9924	0.9941	0.9941	0.9386
COVTYPE 0vs2	0.9339	0.9630	0.9628	0.9630	0.9628	0.7580	0.8479	0.9580	0.9582	0.9569	0.9580	0.7630	0.9614	0.9825	0.9822	0.9830	0.9833	0.7580
ECBDL'14 1M	0.5000	0.7043	0.7053	0.6986	0.6641	0.6515	0.5000	0.7103	0.7085	0.7091	0.6751	0.6870	0.5030	0.7270	0.7210	0.7285	0.5638	0.6952
SUSY IR16	0.6264	0.7701	0.7702	0.7680	0.7649	0.7466	0.6125	0.7716	0.7722	0.7718	0.7719	0.7553	0.6641	0.7934	0.7928	0.7934	0.7892	0.7278
SUSY IR4	0.7200	0.7689	0.7693	0.7685	0.7661	0.7495	0.6909	0.7712	0.7720	0.7717	0.7718	0.7572	0.7433	0.7941	0.7937	0.7939	0.7918	0.7357
KDDCUP dos vs r2l	0.9761	0.9984	0.9993	0.9984	0.9989	0.9986	0.9524	0.9997	0.9996	0.9997	0.9999	0.9996	0.9953	0.9973	0.9992	0.9979	0.9975	0.9992
KDDCUP dos vs nor.	0.9997	0.9998	0.9998	0.9998	0.9998	0.9987	0.9996	0.9996	0.9996	0.9996	0.9995	0.9974	1.0000	1.0000	1.0000	1.0000	1.0000	0.9987
HEPMASS IR16	0.7550	0.8343	0.8348	0.8343	0.8336	0.8183	0.5976	0.8214	0.8220	0.8221	0.8223	0.8144	0.7645	0.8626	0.8619	0.8623	0.8506	0.8126
HIGGS IR16	0.5102	0.6669	0.6659	0.6644	0.6526	0.5961	0.5000	0.6774	0.6783	0.6776	0.6714	0.5986	0.5313	0.7265	0.7254	0.7261	0.6989	0.6061
HEPMASS IR4	0.8006	0.8342	0.8331	0.8337	0.8327	0.8171	0.7444	0.8212	0.8218	0.8216	0.8222	0.8139	0.8237	0.8620	0.8620	0.8619	0.8564	0.8053
HIGGS IR4	0.5381	0.6663	0.6659	0.6651	0.6565	0.5987	0.5047	0.6771	0.6779	0.6776	0.6795	0.6017	0.6119	0.7265	0.7256	0.7261	0.7099	0.6202
ECBDL'14 10M	0.5000	0.7000	0.7002	0.6985	0.6709	0.6587	0.5000	0.7108	0.7108	0.7106	0.6834	0.6891	0.5040	0.7462	0.7432	0.7458	0.5896	0.6994

variants of BAG. Regarding the bad performance of the ROSE variants, the test showed that those variants were only worse than the GBT-based methods. Fig. 1.b shows that only ROS+GBT and GBT (WGini) performed better than more than half of the tested methods. In contrast to the MCC results, the RANF variants were not so badly ranked by the AUC. Finally, it could be seen that the Gini variants performed worse than their respective WGini variants.

Table 6 shows the one-to-one comparisons using the Bayesian hierarchical sign test for each metric and ensemble classifier variant. For the sake of simplicity, instead of showing the ternary heatmap plots, for each comparison the three values (Left, Right, and ROPE) were reported. Left corresponds to the first element under comparison, and Right to the second (*e.g.*, for Gini *vs* WGini comparison, Left corresponds to Gini, and Right to WGini). The best results for each comparison (*i.e.*, a metric and a classifier) appear within black boxes.



(a) MCC (b) AUC

Fig. 2. A Friedman–Nemenyi test comparison of the different ensemble classifiers performing resampling for each base classifier, according to MCC (a) and AUC (b) metrics. The methods connected with a thick horizontal line showed no significant differences between each other (at a level of $\alpha = 0.05$). It was not clear whether resampling was beneficial or not for the MCC. The use of resampling improved AUC performance.

Table 5
Average ranks of the ensemble classifiers performing resampling before training, according to both the MCC (a) and the AUC (b) metrics. The methods below the dashed line showed significant differences with the best one according to the Hochberg procedure at a confidence level of 95%. ROS+GBT was the best method, followed by GBT (WGini).

Algorithm	Avg. rank
ROS+GBT	3.0000
GBT (WGini)	3.4375
GBT (Gini)	3.4375
SMOTE+GBT	4.0000
RUS+GBT	5.5625
BAG (Gini)	7.9375
BAG (WGini)	9.1875
ROS+BAG	10.5000
RUS+BAG	10.7500
SMOTE+BAG	11.0625
ROS+RANF	11.0625
RUS+RANF	11.2500
SMOTE+RANF	11.5625
RANF (WGini)	11.5625
RANF (Gini)	11.7500
ROSE+GBT	14.0312
ROSE+RANF	15.3125
ROSE+BAG	15.5938

(a) MCC

Algorithm	Avg. rank
ROS+GBT	2.3750
GBT (WGini)	2.8125
RUS+GBT	3.7500
SMOTE+GBT	5.1875
BAG (WGini)	7.4375
RUS+BAG	7.5625
RUS+RANF	8.6875
ROS+BAG	8.7500
ROS+RANF	9.1250
SMOTE+RANF	9.3750
RANF (WGini)	9.4375
SMOTE+BAG	9.6875
GBT (Gini)	11.0625
ROSE+GBT	14.1562
ROSE+BAG	14.6562
ROSE+RANF	14.6875
BAG (Gini)	15.5000
RANF (Gini)	16.7500

(b) AUC

For the first set of comparisons, Gini *vs* others, it is remarkable how much it depended on the chosen metric whether the ensemble (using Gini without any resampling) was considered better or not. For example, in BAG and GBT the use of Gini was preferred for MCC (by a narrow margin), whereas the opposite was true for AUC (by a much larger margin). RANF performed in a more consistent way, for all metrics usually the use of resampling or WGini showed itself to be better than the Gini variant. For the second set of comparisons, WGini *vs* ROS, RUS, and SMOTE, Right and Left values were very close, thus no clear winner could be named. Likewise, the comparisons between ROS and RUS were not conclusive and were highly dependent on the ensemble method used and the metric under consideration. The bad performance of ROSE as a resampling method was clearly worse than other resampling techniques (RUS, SMOTE, and ROS) and all non-resampling techniques (Gini and WGini). It must be noted that the ROPE region was almost zero for most of the comparisons, which means that the methods hardly performed equally well, but that one method would be better than the other depending on the specific dataset.

5.2. Resampling within the ensemble

This subsection collects the results of the ensemble methods that perform the resampling within the ensemble; each base classifier in the ensemble will be trained on a dataset obtained after performing a specific resampling for that classifier. Table 7.a details the results of the experiments for MCC, and Table 7.b for AUC. As in the previous section, the boxes and the blueness intensity of the cells, are used to highlight the results and provide an enriched representation. As might be expected, for this strategy, GBT was also shown to be better than BAG. Focusing on the results involving resampling (*i.e.*, excluding Gini and WGini variants), for GBT, ROS could be more beneficial, while RUS apparently performed better for BAG. Nevertheless, looking closely at the tables, row-wise blueness intensity is almost the same for RUS, ROS, and RB, which suggests that all the resampling methods performed in a similar way. As stated before, depending on the chosen metric, pre-processing techniques could be discouraged because the Gini variants of the ensembles, apparently performed better. This happens specifically when looking at the MCC metric, but its unacceptable performance with some datasets such as ECBDL and HIGGS should not be forgotten.

In Table 8, the average ranks computed for each of the performance metrics are shown. GBT+ROS showed the best results for both metrics, being statistically better than any BAG variant. Looking at AUC, the worst methods were the Gini variants of GBT and BAG. The best methods were GBT variants when resampling was used (ROS, RB and RUS), with no differences between all three. On the other hand, for MCC the worst method was BAG+RB and BAG using the weighted Gini index, leaving it apparently quite clear that resampling techniques performed better than the non-resampling (using Gini or weighted Gini) alternatives.

The results of the Friedman–Nemenyi test are shown in Fig. 2. As regards the MCC, GBT+ROS and GBT (Gini) performed better than any BAG variant, on the contrary, BAG+RB and BAG (WGini) performed worse than any GBT variant. Thus, it is not clear whether resampling was beneficial or not. Another interesting insight for this metric, is that all GBT variants were considered equivalent and the same could be said for all BAG variants. Regarding the AUC metric, the use of resampling improved the performance. GBT+ROS and GBT+RB were better than any BAG variant and the Gini variant of GBT. Moreover, GBT (Gini) and BAG (Gini) performed worse than any other GBT variant. For both metrics, five groups of equivalent-performing classifiers were shown, also, the size of the groups ranged from three to five methods so, the statistical differences between most of the methods were pretty small.

Finally, regarding the one-to-one comparisons, Table 9 shows the Bayesian hierarchical sign test for each pair of methods belonging to this part of the study. It should be noted that the Gini *vs* WGini comparison was not reported here, because already appears in Table 6. In view of the comparisons between Gini *vs* resampling (RUS, ROS, and RB), a clear discrepancy is shown between the MCC and the AUC metrics for both ensemble families (BAG and GBT). In view of MCC, the Gini variants performed better than the resampling-based variants by a narrow margin. On the contrary, the resampling-based variants clearly performed better for the AUC metric (*i.e.*, Right region values were close

Table 6
One to-one comparison using the Bayesian hierarchical sign test for each metric and classifier variant. The best results for each comparison (metric and classifier) appear within black boxes. Generally, it was unclear whether one variant was better than other. It depended on the specific dataset, metric, and ensemble method. ROSE was the only variant that showed itself to be clearly worse than the rest.

Comparison	Metric	BAG			RANF			GBT		
		Left	ROPE	Right	Left	ROPE	Right	Left	ROPE	Right
Gini *vs* WGini	MCC	0.6105	0.0000	0.3895	0.4058	0.0000	0.5942	0.6665	0.0000	0.3335
	AUC	0.1180	0.0000	0.8820	0.0165	0.0000	0.9835	0.0648	0.0002	0.9350
Gini *vs* RUS	MCC	0.7355	0.0000	0.2645	0.4263	0.0000	0.5737	0.7737	0.0000	0.2263
	AUC	0.0695	0.0000	0.9305	0.0030	0.0000	0.9970	0.0845	0.0002	0.9153
Gini *vs* ROS	MCC	0.6868	0.0000	0.3132	0.3832	0.0000	0.6168	0.6155	0.0000	0.3845
	AUC	0.0628	0.0002	0.9370	0.0105	0.0000	0.9895	0.0695	0.0000	0.9305
Gini *vs* SMOTE	MCC	0.7072	0.0000	0.2928	0.4433	0.0000	0.5567	0.6073	0.0000	0.3927
	AUC	0.0985	0.0000	0.9015	0.0255	0.0000	0.9745	0.2112	0.0003	0.7885
Gini *vs* ROSE	MCC	0.9145	0.0000	0.0855	0.9455	0.0000	0.0545	0.9768	0.0000	0.0232
	AUC	0.4535	0.0000	0.5465	0.2827	0.0000	0.7173	0.6402	0.0000	0.3598
WGini *vs* RUS	MCC	0.6723	0.0005	0.3272	0.5642	0.0000	0.4358	0.6230	0.0000	0.3770
	AUC	0.5273	0.0235	0.4492	0.5252	0.0003	0.4745	0.4925	0.0037	0.5038
WGini *vs* ROS	MCC	0.5465	0.0000	0.4535	0.5072	0.0000	0.4928	0.4635	0.0000	0.5365
	AUC	0.5122	0.0000	0.4878	0.4990	0.0005	0.5005	0.4783	0.0195	0.5022
WGini *vs* SMOTE	MCC	0.5302	0.0000	0.4698	0.5002	0.0000	0.4998	0.5042	0.0000	0.4958
	AUC	0.5325	0.0008	0.4667	0.5360	0.0013	0.4627	0.6653	0.0000	0.3347
WGini *vs* ROSE	MCC	0.9657	0.0000	0.0343	0.9585	0.0000	0.0415	0.9550	0.0000	0.0450
	AUC	0.9417	0.0003	0.0580	0.9490	0.0000	0.0510	0.9480	0.0000	0.0520
RUS *vs* ROS	MCC	0.3820	0.0012	0.6168	0.4263	0.0002	0.5735	0.3465	0.0000	0.6535
	AUC	0.5155	0.0092	0.4753	0.5172	0.0248	0.4580	0.4810	0.0210	0.4980
RUS *vs* SMOTE	MCC	0.4173	0.0000	0.5827	0.3957	0.0035	0.6008	0.3200	0.0003	0.6797
	AUC	0.5317	0.0003	0.4680	0.5410	0.0043	0.4547	0.7295	0.0113	0.2592
RUS *vs* ROSE	MCC	0.8760	0.0000	0.1240	0.9698	0.0000	0.0302	0.9515	0.0000	0.0485
	AUC	0.8872	0.0000	0.1128	0.9660	0.0000	0.0340	0.9680	0.0000	0.0320
ROS *vs* SMOTE	MCC	0.4567	0.0000	0.5433	0.6167	0.1293	0.2540	0.4675	0.0000	0.5325
	AUC	0.4940	0.0307	0.4753	0.0813	0.8862	0.0325	0.6912	0.0828	0.2260
ROS *vs* ROSE	MCC	0.9278	0.0000	0.0722	0.9865	0.0000	0.0135	0.9862	0.0000	0.0138
	AUC	0.9172	0.0000	0.0828	0.9828	0.0005	0.0167	0.9888	0.0000	0.0112
SMOTE *vs* ROSE	MCC	0.8992	0.0000	0.1008	0.9597	0.0000	0.0403	0.9908	0.0000	0.0092
	AUC	0.8665	0.0000	0.1335	0.9475	0.0008	0.0517	0.9413	0.0022	0.0565

to 1). Using weighted Gini, for BAG ensembles could be sufficient as it has shown itself to be marginally better than applying any kind of resampling. Nevertheless, it was unclear whether resampling might be beneficial for GBT ensembles, as the differences were minimal and opted for one option or the other depending on the metric in use. Finally, regarding the comparisons between resampling methods (RUS, ROS, and RB), no clear winner was found either for the BAG or for the GBT ensembles. The differences that could led to the choice of one method or the other were very small, and thus, the decision will depend on the specific dataset and metric.

5.3. Comparing the two strategies

Having separately tested the performance of the two strategies, a comparison of all the previous methods will be reported in this subsection.

In Table 10, the average ranks of all the algorithms of the study are shown. GBT+ROS was the best method, followed by most of the other alternatives of GBT which were considered statistically equivalent according to the Hochberg procedure at a confidence interval of 95% (for MCC, RUS+GBT was statistically worse; while for AUC, SMOTE+GBT was statistically worse). The

Table 7
Experimental results performing resampling for each ensemble base classifier for MCC (a) and AUC (b). The best results for each classifier and the best overall results appear within gray and black boxes, respectively. GBT showed the best performance for both metrics. For MCC, Gini variants showed the best performance for the majority of the datasets. With regard to the AUC, RUS and ROS were preferable for BAG and GBT, respectively.

(a) MCC

Dataset	BAG					GBT				
	Gini	WGini	RUS	ROS	RB	Gini	WGini	RUS	ROS	RB
COVTYPE 1vs4	0.9522	0.9438	0.9457	0.9467	0.9445	0.9906	0.9776	0.9690	0.9789	0.9703
COVTYPE 1vs3	0.6753	0.5789	0.5783	0.5786	0.5791	0.8548	0.7897	0.7860	0.8016	0.7801
COVTYPE 1vs2	0.9801	0.9756	0.9753	0.9785	0.9769	0.9975	0.9938	0.9932	0.9952	0.9933
COVTYPE 0vs4	0.7408	0.6848	0.6853	0.6863	0.6839	0.9051	0.8036	0.7934	0.8157	0.7941
COVTYPE 0vs3	0.8993	0.7829	0.8042	0.7845	0.7958	0.9751	0.9463	0.9385	0.9571	0.9391
COVTYPE 0vs2	0.8852	0.8033	0.8039	0.8059	0.8179	0.9317	0.8858	0.8852	0.8956	0.8892
ECBDL'14 1M	0.0000	0.1275	0.1279	0.1276	0.1249	0.0602	0.1555	0.1466	0.1613	0.1469
SUSY IR16	0.4385	0.3061	0.3070	0.3073	0.3100	0.4970	0.3537	0.3513	0.3538	0.3514
SUSY IR4	0.5345	0.4685	0.4698	0.4686	0.4504	0.5796	0.5295	0.5312	0.5314	0.5314
KDDCUP dos vs r2l	0.9757	0.9596	0.4164	0.8699	0.9141	0.9955	0.9885	0.4266	0.9982	0.4388
KDDCUP dos vs nor.	0.9987	0.9991	0.9988	0.9993	0.9992	0.9999	0.9999	0.9998	0.9998	0.9998
HEPMASS IR16	0.6565	0.4319	0.4282	0.4311	0.3847	0.6706	0.4742	0.4751	0.4762	0.4743
HIGGS IR16	0.1158	0.1597	0.1608	0.1604	0.1545	0.1944	0.2306	0.2344	0.2352	0.2344
HEPMASS IR4	0.6920	0.6153	0.6149	0.6160	0.5771	0.7188	0.6636	0.6685	0.6684	0.6686
HIGGS IR4	0.2004	0.2663	0.2683	0.2676	0.2543	0.3440	0.3743	0.3785	0.3793	0.3785
ECBDL'14 10M	0.0000	0.1252	0.1253	0.1253	0.1219	0.0717	0.1620	0.1619	0.1650	0.1619

(b) AUC

Dataset	BAG					GBT				
	Gini	WGini	RUS	ROS	RB	Gini	WGini	RUS	ROS	RB
COVTYPE 1vs4	0.9694	0.9933	0.9935	0.9937	0.9944	0.9941	0.9973	0.9971	0.9979	0.9972
COVTYPE 1vs3	0.7939	0.8902	0.8924	0.8900	0.8973	0.9055	0.9629	0.9652	0.9666	0.9637
COVTYPE 1vs2	0.9930	0.9958	0.9957	0.9963	0.9962	0.9988	0.9989	0.9989	0.9992	0.9989
COVTYPE 0vs4	0.8366	0.9595	0.9606	0.9599	0.9611	0.9396	0.9806	0.9806	0.9825	0.9807
COVTYPE 0vs3	0.9221	0.9750	0.9772	0.9765	0.9754	0.9828	0.9939	0.9941	0.9953	0.9942
COVTYPE 0vs2	0.9339	0.9630	0.9633	0.9630	0.9646	0.9614	0.9825	0.9827	0.9839	0.9831
ECBDL'14 1M	0.5000	0.7043	0.7066	0.7049	0.7030	0.5030	0.7270	0.7288	0.7338	0.7293
SUSY IR16	0.6264	0.7701	0.7710	0.7702	0.7695	0.6641	0.7934	0.7949	0.7953	0.7949
SUSY IR4	0.7200	0.7689	0.7697	0.7695	0.7653	0.7433	0.7941	0.7951	0.7952	0.7951
KDDCUP dos vs r2l	0.9761	0.9984	0.9993	0.9976	0.9979	0.9953	0.9973	0.9989	0.9994	0.9990
KDDCUP dos vs nor.	0.9997	0.9998	0.9997	0.9998	0.9998	1.0000	1.0000	1.0000	1.0000	1.0000
HEPMASS IR16	0.7550	0.8343	0.8357	0.8355	0.8316	0.7645	0.8626	0.8643	0.8645	0.8642
HIGGS IR16	0.5102	0.6669	0.6683	0.6675	0.6577	0.5313	0.7265	0.7299	0.7304	0.7300
HEPMASS IR4	0.8006	0.8342	0.8344	0.8344	0.8318	0.8237	0.8620	0.8639	0.8640	0.8641
HIGGS IR4	0.5381	0.6663	0.6677	0.6672	0.6474	0.6119	0.7265	0.7286	0.7290	0.7287
ECBDL'14 10M	0.5000	0.7000	0.7006	0.7004	0.6990	0.5040	0.7462	0.7476	0.7499	0.7476

worst results were for ROSE (for all the ensembles) and RANF (without resampling) that were ranked lowest in the tables.

Fig. 3 shows the performance of the methods by using average ranks with one-to-one comparisons: one metric on each axis.

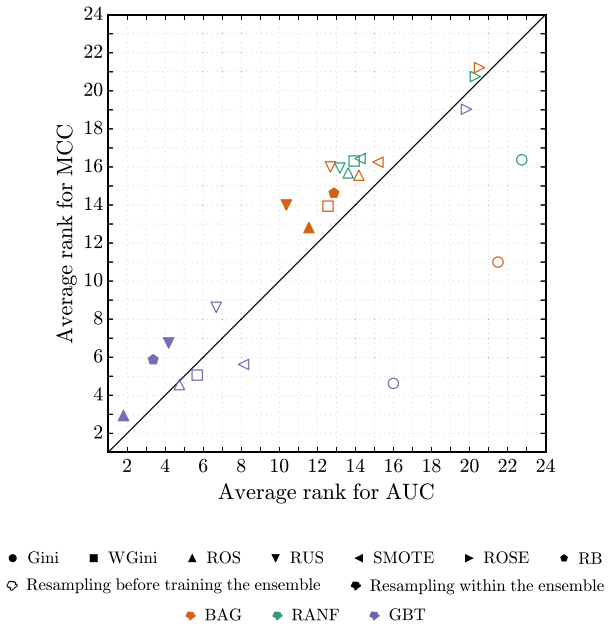


Fig. 3. Average ranks for all ensemble methods according to MCC and AUC metrics. The shape of the marker represents the resampling strategy, the color of the marker represents the ensemble method, and the fill of the marker represents the two resampling strategies: before training (unfilled), and within the ensemble (filled). In general, GBT variants showed better performance than RANF and BAG variants.

The marker aspect represents: the resampling strategy (shape), the ensemble method (color), and the two resampling strategies (unfilled means resampling before training, and filled means resampling within the ensemble). Two clusters could be found, one contained most of the GBT variants on the best positions (left lower corner), while the other contained RANF and BAG variants, which were located farther to the right. Differences between RANF and BAG were not so clear, but in general the BAG variants tended to perform better than the RANF ones. The differences between the metrics were remarkable for some methods, especially for the Gini variants (represented with circular markers), which were positioned far away from the diagonal line. This finding offers an interesting insight, insofar as the use of resampling may be not beneficial for the performance of some classifiers according to some metrics. For this reason, the use of several metrics is advisable and may even be crucial when drawing proper conclusions on imbalance within Big Data environments.

The rankings suggested to us that resampling before training had a fairly similar performance to resampling for each base classifier within the ensemble. This intuitive evaluation may be contrasted in Fig. 4, which represents several Bayesian hierarchical sign tests compared to the performance of resampling before training the ensemble (L) with the performance of resampling within the ensemble (R). The application of RUS (Fig. 4.a) before training, for BAG ensembles, showed a slightly better performance. On the contrary, for GBT ensembles, applying RUS at each iteration of the ensemble marginally outperformed its application once before training. Finally, ROS (Fig. 4.b) showed that it was moderately better when used for each classifier within the ensemble than when used to obtain a balanced dataset with which to train all the classifiers in the ensemble. Irrespective of the detailed analysis presented above, the overall idea, as the clouds of points were almost centered and situated outside the ROPE region, was that no clear winner could be named. Neither could a similar performance between strategies be noted. Therefore, depending on the specific dataset one approach will be better than the other and vice-versa.

Table 8
Average ranks of the ensemble classifiers performing resampling for each base classifier, according to the MCC (a) and the AUC (b) metrics. The methods below the dashed line were statistically different from the best one according to the Hochberg procedure at a confidence level of 95%. GBT+ROS was the best method, being statistically better than any BAG variant.

Algorithm	Avg. rank
GBT+ROS	2.1875
GBT (Gini)	2.5625
GBT (WGini)	3.6250
GBT+RB	3.9375
GBT+RUS	4.6875
BAG (Gini)	6.1250
BAG+ROS	7.3125
BAG+RUS	7.9375
BAG (WGini)	8.3125
BAG+RB	8.3125

(a) MCC

Algorithm	Avg. rank
GBT+ROS	1.1875
GBT+RB	2.3750
GBT+RUS	3.0625
GBT (WGini)	4.0000
BAG+RUS	5.8125
BAG+ROS	6.6250
BAG+RB	7.0000
BAG (WGini)	7.3750
GBT (Gini)	7.5625
BAG (Gini)	10.0000

(b) AUC

5.4. Execution time analysis

In this study, two popular families of ensembles were evaluated: Bagging and Boosting. Their intrinsic differences make their execution times significantly different. As is well known, execution times are crucial in Big Data environments. With this in mind, the training and prediction times of the different ensemble algorithms and resampling techniques for the ECBDL14'10M dataset (see Table 3) are presented in this section. Fig. 5 shows a bar plot with the training and prediction times for all ensemble methods that were tested. On the left hand-side of the plot, orange bars depict prediction times in microseconds (μs). On the right-hand side, purple bars depict training times in seconds (s). BAG, RANF, and GBT ensembles (from top to bottom) were grouped with black horizontal lines.

In training, the fastest method was Random Forest trained with a dataset subsampled by RUS, followed by Random Forest without resampling (Gini and WGini variants). This result was predictable, because RUS decreases the number of instances in the dataset (speeding up the training process), and Random Forest builds decision trees using feature subsets. The approaches that apply resampling within the ensemble (BAG+RUS, BAG+ROS, BAG+RB; and GBT+RUS, GBT+ROS, GBT+RB) were computationally more expensive than those that apply it before training, which was therefore reflected by their execution times. The slowest method was SMOTE+GBT. It is well known that SMOTE is a time-consuming method, because of the kNN computation, and its combination with boosting makes it a less recommendable alternative for Big Data in terms of training time.

In prediction, the differences between the tested methods were much smaller. This result is because, once all the trees of an ensemble are built, voting and prediction is almost the same, regardless of the ensemble family. The resampling before training methods were the fastest, followed by the methods that involved training each base classifier with a different resample.

5.5. Discussion

The experimental results revealed that GBT outperformed Bagging and Random Forest for imbalanced Big Data classification, which is in line with what happens with normal-sized datasets, where Boosting has traditionally outperformed Bagging. More specifically, the combination of ROS within the GBT training (*i.e.*, GBT+ROS) was the best method for all the measures. The main drawback of GBT is the much lengthier time taken for training than Bagging approaches (*e.g.*, RUS+GBT was around 13 times slower than RUS+BAG). Also, GBT scalability in Big Data was limited, because Boosting performs an iterative process, that cannot be fully parallelized.

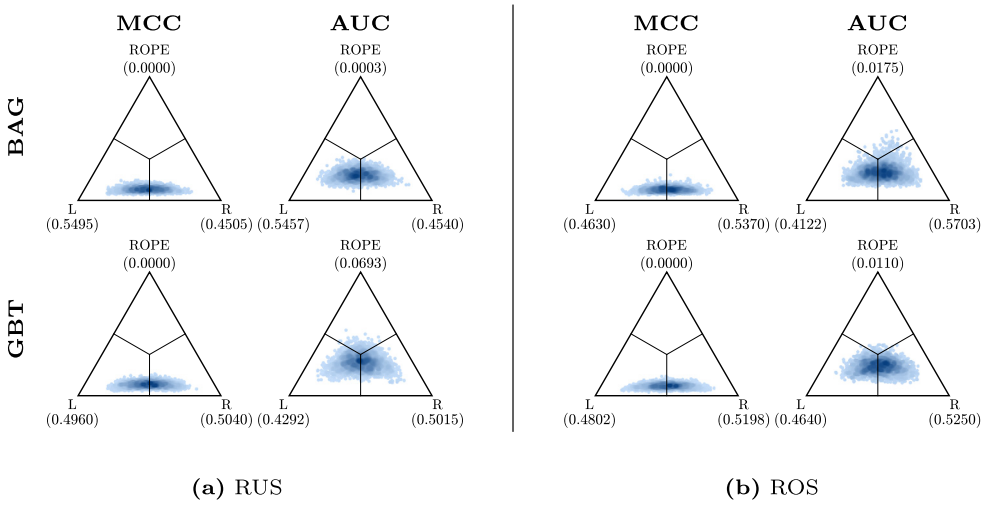


Fig. 4. Bayesian hierarchical sign test heatmaps showing the influence of RUS and ROS resampling before training BAG (top row) and GBT (bottom row) ensembles (L), compared with the influence of RUS and ROS resampling on each iteration of the ensemble (R). Each column represents one metric. There was no clear winner, as the each approach performed better than the others as a function of the specific dataset.

Table 9
Bayesian hierarchical sign tests comparing different resampling methods (RUS, ROS, and RB) applied within the ensemble (BAG and GBT), according to the MCC and AUC metrics. Resampling methods were also compared to base ensembles using Gini and Weighted Gini impurities. Depending on the specific metric, resampling-based ensembles would be recommended or not. Which resampling method is better, is unclear.

Comparison	Metric	BAG			GBT		
		Left	ROPE	Right	Left	ROPE	Right
Gini *vs* RUS	MCC	**0.6950**	0.0000	0.3050	**0.7278**	0.0000	0.2722
	AUC	0.1430	0.0000	**0.8570**	0.0575	0.0000	**0.9425**
Gini *vs* ROS	MCC	**0.6218**	0.0000	0.3782	**0.5615**	0.0000	0.4385
	AUC	0.0725	0.0002	**0.9273**	0.0755	0.0000	**0.9245**
Gini *vs* RB	MCC	**0.6160**	0.0000	0.3840	**0.7155**	0.0000	0.2845
	AUC	0.0833	0.0000	**0.9167**	0.1235	0.0005	**0.8760**
WGini *vs* RUS	MCC	**0.6557**	0.0000	0.3443	**0.5727**	0.0000	0.4273
	AUC	**0.5027**	0.0015	0.4958	0.4460	0.0027	**0.5513**
WGini *vs* ROS	MCC	0.4765	0.0000	**0.5235**	0.4840	0.0000	**0.5160**
	AUC	**0.4998**	0.0232	0.4770	**0.4830**	0.0350	0.4820
WGini *vs* RB	MCC	**0.5242**	0.0000	0.4758	**0.5865**	0.0000	0.4135
	AUC	**0.5458**	0.0025	0.4517	0.4835	0.0095	**0.5070**
RUS *vs* ROS	MCC	0.3788	0.0000	**0.6212**	0.3357	0.0000	**0.6643**
	AUC	**0.5135**	0.0002	0.4863	0.4677	0.0225	**0.5098**
RUS *vs* RB	MCC	0.4283	0.0000	**0.5717**	**0.5125**	0.0000	0.4875
	AUC	**0.5222**	0.0000	0.4778	**0.5092**	0.0005	0.4903
ROS *vs* RB	MCC	**0.5057**	0.0000	0.4943	**0.6233**	0.0000	0.3767
	AUC	**0.5157**	0.0000	0.4843	0.4907	0.0005	**0.5088**

Overall, resampling for each base classifier in the ensemble had a better performance than resampling only once before training. Nevertheless, the differences were not as clear as might be expected, in view of the Hochberg procedure that showed their statistical equivalence, and the Bayesian tests that showed quite balanced distributions for both alternatives, revealing that one strategy could outperform the other on around 50% of occasions.

SMOTE is a powerful but computationally expensive method, which appears not to be as good in Big Data as it is in normal-sized datasets. Something similar happens with ROSE that, although it has proven its validity with normal-sized datasets, clearly showed a worse performance with datasets of greater size. In other words, methods that generate synthetic examples were

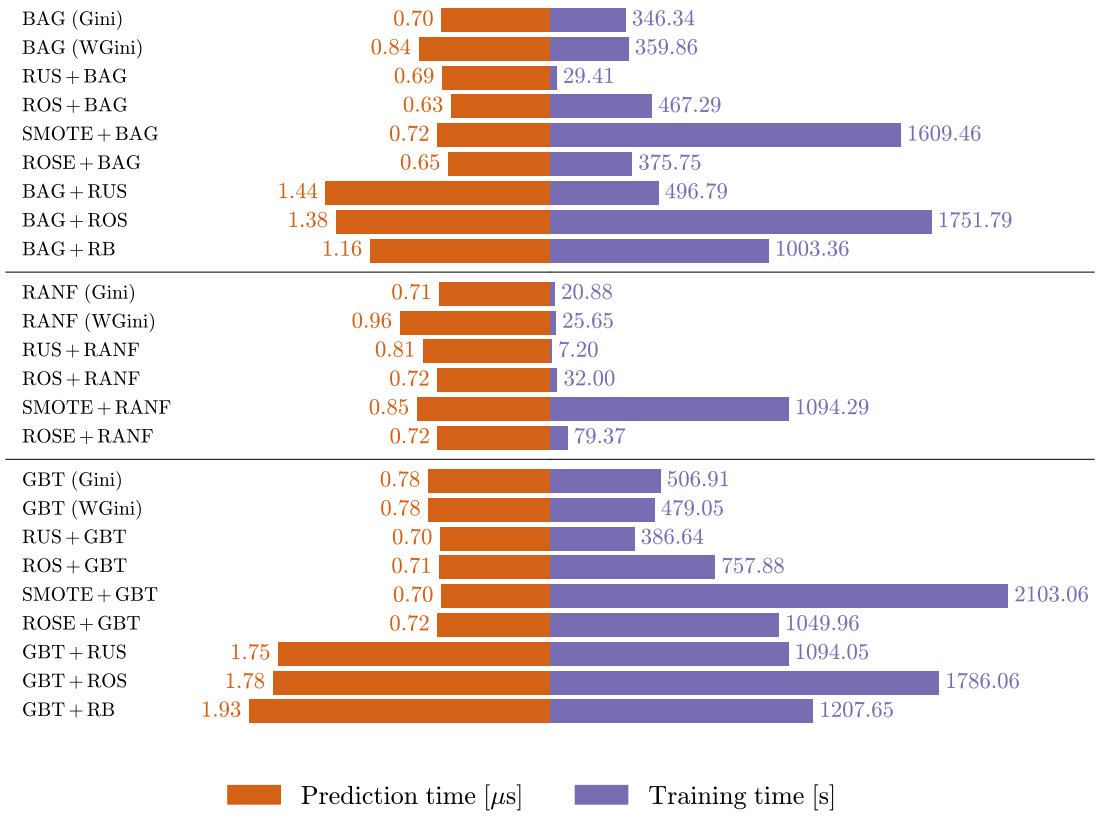


Fig. 5. Comparison of execution times for ECBDL14'10M dataset. Orange bars (on the left) depict the prediction time in microseconds (μs). Purple bars (on the right) depict the training time in seconds (s). RANF showed itself to be the fastest ensemble method, and RUS, the fastest resampling strategy. GBT ensembles and SMOTE resampling strategies, were the two slowest choices. Also, resampling before training the ensemble was faster than resampling within the ensemble.

unable to outperform ROS, which is simpler and less computationally intensive. Our intuitive understanding of this unexpected behavior is that, although the proportion of instances in minority classes remains unfavorable in Big Data, it seems that the increase in these instances in large datasets is enough to make introducing new synthetic instances no beneficial.

It is worth noting that algorithm performance differs greatly depending on the metric that is used. Special attention should be paid to studies and experimentation, because the use of one metric or another might lead to completely different conclusions. For this reason, the use of several imbalance metrics is highly recommended within Big Data environments, just as it is for normal-sized datasets.

With regard to the MCC, ensemble methods without balancing the datasets (*i.e.*, using the Gini index) were quite competitive, specially BAG, which was ranked better than its combination with any resampling technique. Nevertheless, it has to be noted that for some datasets, such as ECBDL14 and HIGGS, an ensemble trained without any balancing technique is the alternative that clearly performed worst of all. Bearing this in mind, and knowing that Gini variants achieved very poor results for AUC metric, we can affirm that balancing techniques can actually contribute functional methods to mitigate the problem of imbalance within Big Data. Whenever no resampling techniques are used, we highly encourage the use of impurity indexes, at the least, that take into account the imbalance, such as the weighted Gini index. The use of weighted Gini has demonstrated itself in this study to be a reasonably good solution for imbalanced Big Data: it is fast and straightforward to apply by instance weighting.

6. Conclusions and future work

Although there are numerous studies on the classification of imbalanced data, these studies are practically non-existent in the context of imbalanced Big Data classification. This paper has shed light on the impact of using data-level approaches within Big Data ensemble classification. Those approaches mainly involve the use of pre-processing techniques such as RUS, ROS, SMOTE, or ROSE for transforming an imbalanced dataset into a balanced one. Whether rebalancing is better performed once only before training the ensemble, or as many times as there are base classifiers contained in the ensemble, was also evaluated. All the experiments were performed on Bagging-based and Boosting-based ensembles, highlighting how resampling techniques specifically affect both ensemble families, in terms of performance. The conclusions of the study, although some might appear ambiguous, can be very useful to help guide future research work into imbalance in Big Data classification.

Within the experimental framework, Boosting ensembles, although requiring more computational power, clearly outperformed Bagging-based alternatives. The use on the trees construction of an impurity index that takes into account the imbalance, such as weighted Gini, offered roughly equivalent results to the use of resampling techniques. Surprisingly, the training of the ensembles on the original datasets without any change (using the standard Gini index), offered quite good results overall (for MCC and F_1-score metrics). However, this procedure is not advisable, because the results were dreadful for some datasets (clearly visible when AUC or G-mean were used), but still an accurate indicator of a lack of robust solutions for improving the performance for all the metrics. Regarding the resampling

Table 10
Average ranks of all ensemble classifiers evaluated in this study, according to the MCC and AUC metrics. The statistical differences between the methods below the dashed line and the best method were significant, according to the Hochberg procedure at a confidence level of 95%. GBT-based ensembles showed themselves to be statistically better than BAG and RANF ensembles.

Algorithm	Avg. rank
GBT+ROS	2.9375
ROS+GBT	4.5625
GBT (Gini)	4.6250
GBT (WGini)	5.0625
SMOTE+GBT	5.6250
GBT+RB	5.8750
GBT+RUS	6.7500
RUS+GBT	8.6250
BAG (Gini)	11.0000
BAG+ROS	12.8125
BAG (WGini)	13.9375
BAG+RUS	14.0000
BAG+RB	14.6250
ROS+BAG	15.5625
ROS+RANF	15.6875
RUS+RANF	15.9375
RUS+BAG	16.0000
SMOTE+BAG	16.2500
RANF (WGini)	16.3125
RANF (Gini)	16.3750
SMOTE+RANF	16.4375
ROSE+GBT	19.0312
ROSE+RANF	20.7500
ROSE+BAG	21.2188

(a) MCC

Algorithm	Avg. rank
GBT+ROS	1.8125
GBT+RB	3.3750
GBT+RUS	4.1875
ROS+GBT	4.7500
GBT (WGini)	5.6875
RUS+GBT	6.6875
SMOTE+GBT	8.1250
BAG+RUS	10.3750
BAG+ROS	11.5625
BAG (WGini)	12.5625
RUS+BAG	12.6875
BAG+RB	12.8750
RUS+RANF	13.1875
ROS+RANF	13.6250
RANF (WGini)	13.9375
ROS+BAG	14.1875
SMOTE+RANF	14.2500
SMOTE+BAG	15.1875
GBT (Gini)	16.0000
ROSE+GBT	19.8438
ROSE+RANF	20.3125
ROSE+BAG	20.5312
BAG (Gini)	21.5000
RANF (Gini)	22.7500

(b) AUC

methods, ROS generally achieved better results, but with only a minimal advantage, closely followed by RUS and SMOTE with no statistically significant differences. In contrast, ROSE was clearly the worst alternative. Our conclusion is therefore that complex methods that involve the generation of synthetic instances are not as effective for Big Data as they are for normal-sized datasets. Although they could, depending on the dataset, be the best option, in general we discourage their use in favor of simpler and faster methods such as ROS.

Ensembles specifically designed to overcome the imbalance problem (*i.e.*, resampling before training each base classifier), achieved better performance than resampling a dataset once and then training a conventional ensemble with it. Nevertheless, the differences between the two strategies were very small, suggesting that whether one strategy is actually better than the other will strongly depend on the data set to which it is applied. Therefore, whenever execution times are considered critical, as it is often the case with Big Data, the general recommendation would be to use the faster strategies based on a single initial resampling or the use of impurity indexes that take into account imbalance (*e.g.* weighted Gini).

An interesting insight is the importance of using different evaluation metrics when dealing with imbalance problems. This is preferable, because each metric uses the values within the confusion matrix in a specific way and therefore has its own strengths and weaknesses. Hence, using more than one metric yields a more informed view of the results and a better evaluation of the performance of a single classifier. The conclusions that can be drawn by using MCC and F_1-score are very different than using AUC or G-mean, thus at least one metric of each group should be used for future Big Data imbalance studies (in addition, it is enough to use one from each group, since the conclusions that can be obtained are similar for the two metrics within each group).

Despite the advances relating to the classification of Big Data in recent years, research into imbalanced Big Data is still scarce and more studies and surveys are needed to unify the publications that periodically emerge. More precisely, the presence of the pathologies that are traditionally associated with imbalanced datasets, such as overlapping, noisy examples, small disjuncts, and borderline instances [32], have yet to be studied in Big Data.

The resampling in this study perfectly balanced the datasets with 50% of the examples belonging to each class (with the exception of RB, for which the imbalanced ratios were random for each base classifier). Another interesting future line of research would be the evaluation of how different resampling ratios affect different classifiers.

In view of the results given by the data-level approaches for imbalanced Big Data learning, we place the focus on the exploration of algorithm-level approaches, which should assist more fruitful advances for these kinds of problems. An interesting research line could be the evaluation of novel forest-based approaches that have recently emerged, such as Random Forest quantile classifier [55] and the adaptation of Oblique Random Forest [56] for imbalanced learning, specially for large datasets. The design and implementation of new classifiers for Big Data frameworks, such as Apache Spark, is a promising research line nowadays, as is the adaptation and revalidation of any popular proposal for normal-sized datasets.

CRediT authorship contribution statement

Mario Juez-Gil: Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Visualization, Funding acquisition. **Álvar Arnaiz-González:** Conceptualization, Software, Validation, Resources, Writing - original draft, Supervision, Project administration. **Juan J. Rodríguez:** Conceptualization, Writing - review & editing, Supervision, Project administration, Funding acquisition. **César García-Osorio:** Conceptualization, Writing - review & editing, Supervision, Project administration, Funding acquisition.

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.

Acknowledgments

The project leading to these results has received funding from "la Caixa" Foundation, Spain, under agreement LCF/PR/PR18/51130007. This work was supported by the *Junta de Castilla y León, Spain* under project BU055P20 (JCyL/FEDER, UE) co-financed through European Union FEDER funds, and by the *Consejería de Educación* of the *Junta de Castilla y León* and the European Social Fund, Spain through a pre-doctoral grant (EDU/1100/2017). This material is based upon work supported by Google Cloud, United States.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.asoc.2021.107447.

References

[1] D. Laney, 3-d data management: controlling data volume, velocity and variety, Technical report, META Group Research Note, 2001.
[2] J. Gantz, D. Reinsel, Extracting value from chaos, IDC iview 1142 (2011) (2011) 1–12.
[3] A. Jain, The 5 v's of big data, 2016, [Online; accessed 17-July-2020].
[4] R.H. Hariri, E.M. Fredericks, K.M. Bowers, Uncertainty in big data analytics: survey, opportunities, and challenges, J. Big Data 6 (1) (2019) 44.
[5] N.V. Chawla, N. Japkowicz, A. Kotcz, Editorial: Special issue on learning from imbalanced data sets, SIGKDD Explor. Newsl. 6 (1) (2004) 1–6.

[6] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284.
[7] J.F. Díez-Pastor, J.J. Rodríguez, C.I. García-Osorio, L.I. Kuncheva, Diversity techniques improve the performance of the best imbalance learning ensembles, Inform. Sci. 325 (2015) 98–117.
[8] A. Fernández, S. del Río, N.V. Chawla, F. Herrera, An insight into imbalanced big data classification: outcomes and challenges, Complex Intell. Syst. 3 (2) (2017) 105–120.
[9] B. Krawczyk, Learning from imbalanced data: open challenges and future directions, Prog. Artif. Intell. 5 (4) (2016) 221–232.
[10] X. Gao, Z. Chen, S. Tang, Y. Zhang, J. Li, Adaptive weighted imbalance learning with application to abnormal activity recognition, Neurocomputing 173 (2016) 1927–1935.
[11] A. Azaria, A. Richardson, S. Kraus, V.S. Subrahmanian, Behavioral analysis of insider threat: A survey and bootstrapped prediction in imbalanced data, IEEE Trans. Comput. Soc. Syst. 1 (2) (2014) 135–155.
[12] J.F. Diez-Pastor, A. Gil Del Val, F. Veiga, A. Bustillo, High-accuracy classification of thread quality in tapping processes with ensembles of classifiers for imbalanced learning, Measurement 168 (2021) 108328.
[13] E. Ramentol, I. Gondres, S. Lajes, R. Bello, Y. Caballero, C. Cornelis, F. Herrera, Fuzzy-rough imbalanced learning for the diagnosis of high voltage circuit breaker maintenance: The smote-frst-2t algorithm, Eng. Appl. Artif. Intell. 48 (2016) 134–139.
[14] Z. Gao, L.-f. Zhang, M.-y. Chen, A. Hauptmann, H. Zhang, A.-N. Cai, Enhanced and hierarchical structure algorithm for data imbalance problem in semantic extraction under massive video dataset, Multimedia Tools Appl. 68 (3) (2014) 641–657.
[15] B. Krawczyk, M. Galar, Łukasz. Jeleń, F. Herrera, Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy, Appl. Soft Comput. 38 (2016) 714–726.
[16] J.L. Leevy, T.M. Khoshgoftaar, R.A. Bauder, N. Seliya, A survey on addressing high-class imbalance in big data, J. Big Data 5 (1) (2018) 42.
[17] J. Luengo, D. García-Gil, S. Ramírez-Gallego, S. García, F. Herrera, Big Data Preprocessing: Enabling Smart Data, Springer Nature, 2020.
[18] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, F. Herrera, A review on ensembles for the class imbalance problem: Bagging-boosting-, and hybrid-based approaches, IEEE Trans. Syst. Man Cybern. C 42 (4) (2012) 463–484.
[19] H. Ghaderi Zefrehi, H. Altınçay, Imbalance learning using heterogeneous ensembles, Expert Syst. Appl. 142 (2020) 113005.
[20] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, G. Bing, Learning from class-imbalanced data: Review of methods and applications, Expert Syst. Appl. 73 (2017) 220–239.
[21] A. Fernández, S. García, M. Galar, R.C. Prati, B. Krawczyk, F. Herrera, Learning from Imbalanced Data Sets, Springer, 2018.
[22] S. González, S. García, J. Del Ser, L. Rokach, F. Herrera, A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities, Inf. Fusion 64 (2020) 205–237.
[23] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[24] R.E. Schapire, The strength of weak learnability, Mach. Learn. 5 (2) (1990) 197–227.
[25] L.I. Kuncheva, J.J. Rodriguez, Classifier ensembles with a random linear oracle, IEEE Trans. Knowl. Data Eng. 19 (4) (2007) 500–508.
[26] J. Maudes, J.J. Rodríguez, C. García-Osorio, Disturbing neighbors diversity for decision forests, in: Applications of Supervised and Unsupervised Ensemble Methods, Springer, 2009, pp. 113–133.
[27] J. Maudes, J.J. Rodríguez, C. García-Osorio, Disturbing neighbors ensembles for linear SVM, in: International Workshop on Multiple Classifier Systems, Springer, 2009, pp. 191–200.
[28] J. Maudes, J.J. Rodríguez, C. García-Osorio, N. García-Pedrajas, Random feature weights for decision tree ensemble construction, Inf. Fusion 13 (1) (2012) 20–30.
[29] C. Pardo, J.J. Rodríguez, J.F. Díez-Pastor, C. García-Osorio, Random oracles for regression ensembles, in: Ensembles in Machine Learning Applications, Springer, 2011, pp. 181–199.
[30] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.
[31] G. Menardi, N. Torelli, Training and assessing classification rules with imbalanced data, Data Min. Knowl. Discov. 28 (1) (2014) 92–122.
[32] J.F. Díez-Pastor, J.J. Rodríguez, C. García-Osorio, L.I. Kuncheva, Random balance: Ensembles of variable priors classifiers for imbalanced data, Knowl.-Based Syst. 85 (2015) 96–111.
[33] J. Tanha, Y. Abdi, N. Samadi, N. Razzaghi, M. Asadpour, Boosting methods for multi-class imbalanced data classification: an experimental review, J. Big Data 7 (1) (2020) 1–47.
[34] X. Liu, J. Wu, Z. Zhou, Exploratory undersampling for class-imbalance learning, IEEE Trans. Syst. Man Cybern. B 39 (2) (2009) 539–550.
[35] Y. Jeon, D. Lim, PSU: Particle stacking undersampling method for highly imbalanced big data, IEEE Access 8 (2020) 131920–131927.
[36] S. del Río, V. López, J.M. Benítez, F. Herrera, On the use of mapreduce for imbalanced big data using random forest, Inform. Sci. 285 (2014) 112–137.
[37] T. Hasanin, T. Khoshgoftaar, The effects of random undersampling with simulated class imbalance for big data, in: 2018 IEEE International Conference on Information Reuse and Integration (IRI), 2018, pp. 70–79.
[38] A. Fernández, S. Garcia, F. Herrera, N.V. Chawla, Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary, J. Artif. Intell. Res. 61 (2018) 863–905.
[39] M.J. Basgall, W. Hasperué, M. Naiouf, A. Fernández, F. Herrera, SMOTE-BD: An exact and scalable oversampling method for imbalanced classification in big data, J. Comput. Sci. Tech. 18 (03) (2018) 203–209.
[40] F. Hu, H. Li, A novel boundary oversampling algorithm based on neighborhood rough set model: NRSBoundary-SMOTE, Math. Probl. Eng. (2013) 2013.
[41] F. Hu, H. Li, H. Lou, J. Dai, A parallel oversampling algorithm based on nrsboundary-smote, J. Inf. Comput. Sci. 11 (13) (2014) 4655–4665.
[42] I. Triguero, S. del Río, V. López, J. Bacardit, J.M. Benítez, F. Herrera, ROSEFW-RF: The winner algorithm for the ECBDL'14 big data competition: An extremely imbalanced big data bioinformatics problem, Knowl.-Based Syst. 87 (2015) 69–79, Computational Intelligence Applications for Data Science.
[43] I. Triguero, M. Galar, S. Vluymans, C. Cornelis, H. Bustince, F. Herrera, Y. Saeys, Evolutionary undersampling for imbalanced big data classification, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 715–722.
[44] T. Liu, A.W. Moore, A. Gray, K. Yang, An investigation of practical approximate nearest neighbor algorithms, in: Proceedings of the 17th International Conference on Neural Information Processing Systems, in: NIPS'04, MIT Press, Cambridge, MA, USA, 2004, pp. 825–832.
[45] N. Lunardon, G. Menardi, N. Torelli, ROSE: A package for binary imbalanced learning, R J. 6 (1) (2014).
[46] H. Liu, M. Zhou, X.S. Lu, C. Yao, Weighted gini index feature selection method for imbalanced data, in: 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC), 2018, pp. 1–6.
[47] C. Chen, A. Liaw, L. Breiman, Using Random Forest to Learn Imbalanced Data, Vol. 110, (1–12) University of California, Berkeley, 2004, p. 12.
[48] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 1–27.
[49] D. Dua, C. Graff, UCI machine learning repository, 2017.
[50] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[51] A. Benavoli, G. Corani, J. Demšar, M. Zaffalon, Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis, J. Mach. Learn. Res. 18 (77) (2017) 1–36.
[52] M. Juez-Gil, 2020, mjuez/baycomp_plotting.
[53] D. Brzezinski, J. Stefanowski, R. Susmaga, I. Szczech, On the dynamics of classification measures for imbalanced and streaming data, IEEE Trans. Neural Netw. Learn. Syst. 31 (8) (2020) 2868–2878.
[54] M. Bekkar, H.K. Djemaa, T.A. Alitouche, Evaluation measures for models assessment over imbalanced data sets, J. Inf. Eng. Appl. 3 (10) (2013).
[55] R. O'Brien, H. Ishwaran, A random forests quantile classifier for class imbalanced data, Pattern Recognit. 90 (2019) 232–249.
[56] R. Katuwal, P. Suganthan, L. Zhang, Heterogeneous oblique random forest, Pattern Recognit. 99 (2020) 107078.