An analysis of local and global solutions to address Big Data imbalanced classification: a case study with SMOTE preprocessing.

María José Basgall[1,2,3] ✉ https://orcid.org/0000-0002-7024-847X, Waldo Hasperué[2]
https://orcid.org/0000-0002-9950-1563, Marcelo Naiouf[2]
https://orcid.org/0000-0001-9127-3212, Alberto Fernández[4]
https://orcid.org/0000-0002-6480-8434, and Francisco Herrera[4]
https://orcid.org/0000-0002-7283-312X

[1] UNLP, CONICET, III-LIDI, La Plata, Argentina
mjbasgall@lidi.info.unlp.edu.ar
[2] Instituto de Investigación en Informática (III-LIDI), CIC-PBA
Facultad de Informática - Universidad Nacional de La Plata, Argentina
[3] University of Granada, Granada, Spain
[4] DaSCI Andalusian Institute of Data Science and Computational Intelligence,
University of Granada, Granada, Spain

Abstract. Addressing the huge amount of data continuously generated is an important challenge in the Machine Learning field. The need to adapt the traditional techniques or create new ones is evident. To do so, distributed technologies have to be used to deal with the significant scalability constraints due to the Big Data context.

In many Big Data applications for classification, there are some classes that are highly underrepresented, leading to what is known as the imbalanced classification problem. In this scenario, learning algorithms are often biased towards the majority classes, treating minority ones as outliers or noise.

Consequently, preprocessing techniques to balance the class distribution were developed. This can be achieved by suppressing majority instances (undersampling) or by creating minority examples (oversampling). Regarding the oversampling methods, one of the most widespread is the SMOTE algorithm, which creates artificial examples according to the neighborhood of each minority class instance.

In this work, our objective is to analyze the SMOTE behavior in Big Data as a function of some key aspects such as the oversampling degree, the neighborhood value and, specially, the type of distributed design (local vs. global).

Keywords: big data, imbalanced classification, preprocessing techniques, SMOTE, scalability

1 Introduction

Currently, Data Science has an essential role on analyzing the enormous amount of data being generated in every moment. This is known as Big Data, and the

more volume of available information, the more knowledge could be discovered [1].

However, it is known that the "small data" or standard size problems implementations are not directly applicable to Big Data due to the scalability constraints [2]. For this reason, the traditional techniques have to be adapted to the "divide-and-conquer" approach proposed by "the facto" MapReduce [3] framework for Big Data. In this direction, two alternatives are known, the local and the global design approaches [4]. The former works with each data partition separately and the results of each Map process is put together on a single Reduce process. And the latter, generates global results by distributing data and models across the Map processes. This is considered as an exact model because the final results are obtained through a more complete insight of the data.

It has to be considered that this huge amount of data does not imply all of it will be useful. Indeed, most of the times, a subset of the data will be the real source of the knowledge discovery process. As it happens with "small data", the results of this process are directly related to the quality of the data used. Thus, to obtain high quality data (also known as Smart Data [5]), preprocessing techniques have to be applied.

In order to study the data quality in a Big Data context, the focus is set on a common situation when a classification problem is faced: imbalanced (or uneven) data distribution. Imbalanced data classification is a meaningful topic due to the large amount of real problems in which the key concept is represented by the minority class (e.g., medical diagnosis of rare diseases).

In this research area, the existent methods for balancing data are undersampling and oversampling. They work eliminating or creating, majority or minority class instances, respectively. With respect to the oversampling methods, the SMOTE ("Synthetic Minority Oversampling TEchnique") [6, 7] algorithm is one of the most widespread. SMOTE creates artificial examples by interpolation according to the neighborhood of each minority class instance.

In this work, an analysis of the current preprocessing solutions for imbalanced Big Data behavior is carried out. A performance comparison of the solutions related to parameters of interest, such as the number of partitions and oversampling final ratio is shown. In addition, and regarding the SMOTE algorithm, the focus is on contrasting the behavior between the global and the local scheme implementations. The main objective is to analyze the obtained results to determine the dependency of the imbalance preprocessing or the data intrinsic quality. Another aspect to evaluate is if their performance behave as in traditional datasets context.

The article continues organized as follows. In Sect. 2, the current solutions for imbalanced Big Data classification are described. Section 3 details the experimental environment used in this work. Then, in Sect. 4, the comparative results are shown. Finally, in Sect. 5, conclusions and future works are described.

2   Big Data and the imbalanced classification problem

In this section, a brief introduction to the most used Big Data frameworks is presented in Sect. 2.1. Furthermore, a quick review about imbalanced classification and a description of its methods for Big Data are depicted in Sect. 2.2.

2.1   Big Data technologies

Due to Big Data, new technologies appeared in order to cope with it. Among them, in 2003 and developed by Google, the most significant was born: MapReduce [3]. This framework was design based on a "divide-and-conquer" scheme in order to process Big Data on a cluster using parallel and distributed implementations. MapReduce model presents two stages called Map and Reduce. The former receives data and performs operations in order to transform them. The latter process the results of the previous phase to summarize them. This model works with key-value pairs. In order to process them in parallel, all the pairs of the same key are distributed to the same node.

The most popular open-source frameworks based on MapReduce model programming are Apache Hadoop [8] and Apache Spark [9, 10]. The main difference between them is that Hadoop performs an intensive disk usage, and Spark an intensive memory usage. This generates that Spark outperforms Hadoop. Also Spark provides integration with many libraries such as MLlib [11] (the Machine Learning library), Spark Streaming [12] (to work with streams of data), among others. These are some of the reasons which make Spark the current widespread Big Data framework.

In Sec. 1 two design methods related to the use of data and models distribution were depicted: the local and the global [4]. Depending on which model is applied, the results of the developed algorithm will be approximated or exact.

2.2   Imbalanced classification in Big Data

In a classification task, poor quality or not optimal data, will imply that the results will neither be. A previous process of adaptation has to be applied on data to carry out a good learning. A common scenario to apply classifiers is when the dataset class distribution is imbalanced. The simplest preprocessing techniques to achieve a balanced dataset are ROS (Random Over Sampling) and RUS (Random Under Sampling) algorithms [13]. On one hand, ROS works replicating the minority instances in a random way in order to achieve the desired ratio balance between both classes. On the other hand, RUS creates a balanced dataset by random deletion of the majority instances.

As was introduced in Sec. 1, the SMOTE algorithm [6, 7] is one of the most applied on the oversampling preprocessing cases. SMOTE works over the minority class instances (also known as positive instances) by calculating the k nearest neighbors (kNN) of each of them. This technique creates a balanced dataset interpolating each of the positive examples with its neighbors as can be seen in Fig. 1.
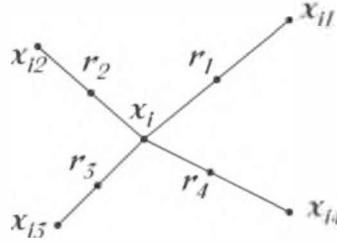
Fig. 1. Interpolation between a minority instance and its k nearest neighbors ($k = 4$).

In the MapReduce approach, the data partitioning task may lead to lack of data when processing local models. Furthermore, it could also cause "small-disjuncts" [14]. This extreme lower number of minority instances in each Map gives an incomplete representation of the dataset information. In particular, regarding the real neighborhood of each example. "Small-disjuncts" are tiny groups of very local data and with low density, surrounded by the majority class instances. Minority instances created from them may enter in the exclusive zones of the majority class inducing noise or an over-generalization.

To the best of our knowledge, no other solutions than the ROS, RUS and the SMOTE are currently available as preprocessing techniques for imbalanced Big Data problems.

In [15], authors present a work in which these methods have been adapted to the MapReduce programming style, making them suitable for Big Data. They are called RUS-BigData and ROS-BigData respectively, and both of them have been designed using the local approach. Each Map process adjusts the class distribution for the data that belongs to it by performing the under or the over sampling. Thus, these are solutions independent of the partitions number. Then, to obtain the balanced dataset, a single Reduce process collects the results generated by each Map.

Regarding the SMOTE algorithm, in [16] a global SMOTE fully scalable solution was described, called SMOTE-BD. In order to cope with the potential data partitioning problems, the whole neighborhood of each minority class instance is taken into account. That is achieved by the use of scalable data structures. The source code of SMOTE-BD can be found as a package in the Spark-packages repository [17]. The k nearest neighbors (kNN) calculation was based on [18]. Furthermore, a local SMOTE version called SMOTE-MR is available in [19]. The k nearest neighbors for each minority instance are obtained from data belonging to the same instance's partition. Working independently on each Map, gives approximated final results. This is the reason why methods are called global (or exact) and local (or approximated) as mentioned before.

Some aspects to remark are that the exact approach requires more effort in the solution development than the approximated one (which is straightforward to the MapReduce programming model). And the main advantage of the exact

design is the learning of more robust models due to the capacity of sharing data and models [4, 20].

All of the mentioned preprocessing techniques in this section were developed using Apache Spark.

3   Experimental framework

In this section, the experimental environment is detailed. First, in Sect. 3.1 the datasets and the algorithms parameters configuration used in the tests are enumerated. Then, the classifier and the evaluation metrics are described in Sec. 3.2. Finally, in Sect. 3.3 the infrastructure used for the experiments is mentioned.

3.1   Datasets and algorithms parameters

In order to compare the performance of the four preprocessing algorithms for Big Data, three imbalanced datasets were selected from the UCI Machine Learning repository [21], each one with a very different imbalanced ratio.

Table 1 shows the datasets summary, where the number of examples (*#Ex.*), number of attributes (*#Atts.*), number of instances for each class (*#(maj; min)*), class distribution (*%(maj; min)*) and imbalance ratio (*IR*) are included.

Table 1. Datasets summary

Datasets	#Ex.	#Atts.	#(maj; min)	%(maj; min)	IR
covtype7	464,677	54	(448,421; 16,256)	(96.5; 3.5)	27.58
higgs	4,954,754	28	(4,663,298; 291,456)	(94.12; 5.88)	16
susy	2,212,186	18	(2,169,299; 542,435)	(80; 20)	4

The parameters used for the methods according to their authors' specifications are shown in Table 2. As can be seen, a division was made in order to show the parameters in common for all the algorithms and the specific for the SMOTEs.

The percentage of oversampling (*% oversampling*) represents the final desired distribution between classes, that is, the final ratio. For instance, if $\% \, oversampling = 100$, both classes will have the same quantity of instances, in other words, a 1:1 ratio; and if $\% \, oversampling = 150$, the minority class will end up with a 50% more of instances than the majority class, this mean, a 1.5:1 ratio, and so on. There are several studies where the percentage of oversampling had influenced significantly over the results [22] as they increase. In consequence, three values were selected for this parameter with the purpose to test each of them.

The number of partitions (*# partitions*) sets the amount of Map process to be used. That means the number in which input data will be split.

Regarding both SMOTE versions, as they calculate the k Nearest Neighbors of each minority instance, different values of the k parameter have been proposed. The euclidean distance function was used.

Table 2. Algorithms and parameters

Algorithms	Parameters	Values
All	% oversampling	100 / 150 / 200
	# partitions	32 / 64 / 128
SMOTEs	k Nearest Neighbors	3 / 5 / 7
	Distance function	Euclidean

3.2   Classifier and evaluation metrics

The behavior of the resultant preprocessed datasets was tested using the Decision Trees classifier (DT), implemented in the Spark's MLlib library [11]. The Apache Spark and MLlib version used for this work was the 2.2.0.

Table 3 shows a confusion matrix for a binary problem from which the classification quality metrics are obtained. This matrix organizes the samples of each class according to their correct or incorrect identification. Thus, the prediction quality for each individual class are represented by the True Positive (TP) and True Negative (TN) values. These measures indicate if the preprocessing is favoring a single class or concept.

Table 3. Confusion matrix for performance evaluation of a binary classification problem

Actual	Predicted	
	Positive	Negative
Positive	True positive (TP)	False negative (FN)
Negative	False positive (FP)	True negative (TN)

Also, four metrics that describe both classes independently are obtained from it:

True Positive Rate, defined as $TPR = \dfrac{TP}{TP + FN}$, is the percentage of positive instances correctly classified.

True Negative Rate, defined as $TNR = \dfrac{TN}{FP + TN}$, is the percentage of negative instances correctly classified.

False Positive Rate, defined as $FPR = \dfrac{FP}{FP + TN}$, is the percentage of negative instances misclassified.

False Negative Rate, defined as $FNR = \dfrac{FN}{TP + FN}$, is the percentage of positive instances misclassified.

In order to evaluate the performance in imbalanced classification scenarios, more robust metrics which make use of these rates exist. Two of the most widely used are the Geometric Mean (GM) [23] and the area under the ROC curve (AUC) [24]. The former is defined in Eq. 1 and it attempts to maximize the accuracy of each one of the two classes at the same time. The latter is defined by the area under the curve given by the Eq. 2 and it evaluates which model is better on average, with a single measure.

$$GM = \sqrt{TPR * TNR} \tag{1}$$

$$AUC = \frac{1 + TPR - FPR}{2} \tag{2}$$

3.3 Infrastructure

Concerning the infrastructure used to perform the experiments, the Hadoop cluster at University of Granada was used. The cluster consists of fourteen nodes connected via a Gigabit Ethernet network. Each node has a Intel Core i7-4930K microprocessor at 3.40GHz, 6 cores (12 threads) and 64 GB of main memory working under Linux CentOS 6.9. The infrastructure works with Hadoop 2.6.0 (Cloudera CDH5.8.0), where the head node is configured as NameNode and ResourceManager, and the rest are DataNodes and NodeManagers.

4   Experimental results

In this section, the performances achieved by a Decision Tree classifier after applying independently each of the preprocessing techniques are presented.

The following tables show the average results of applying all the preprocessing methods (ROS, RUS, SMOTE-BD and SMOTE-MR)[1] on the three datasets for each oversampling percentage and number of partitions values (shown on table 2). Where boldface indicates the highest value, and the best result for each dataset is underlined.

---

[1] The SMOTE variants are abbreviated as "SMT-BD" or "SMT-MR" in all tables.

Tables 4, 5 and 6 present the obtained values considering the GM, TPR and TNR performance measures, respectively. In Table 4, no significant differences were found between the results for the same parameters configuration for each method. In general, it can be seen that all of techniques are scalable in quality regarding the partition numbers. This behavior was not expected for SMOTE-MR because the kNN of each positive instance is calculated over the data of its partition. Further investigation is ongoing to fully understand this result.

As mentioned in Sec. 3, we expected a performance improvement with the increase in the oversampling percentage. In our current datasets of study, no major variations in performance are seen, and if so, a small variation goes in the opposite direction, giving worse results (e.g. higgs dataset) for larger oversampling percentage.

Comparing with previous experience on small data [14], the different obtained results may be due to the data quality, which has not been assessed yet. The selected large datasets may have high redundancy and, therefore, the zones which need to be more strengthened, are being neglected or underestimated.

Table 4. GM average results for the four methods over the datasets for 32, 64 and 128 partitions and for 100, 150 and 200 oversampling percentage.

Part.		32				64				128			
Method		SMT-BD	SMT-MR	ROS	RUS	SMT-BD	SMT-MR	ROS	RUS	SMT-BD	SMT-MR	ROS	RUS
Perc.	Dataset												
100	higgs	0.6462	0.6477	**0.6580**	0.6560	0.6468	0.6474	**0.6562**	0.6505	0.6479	0.6476	0.6561	**0.6568**
	covtype7	0.9249	0.9242	**0.9363**	**0.9363**	0.9251	**0.9279**	0.9232	0.9271	**0.9306**	0.9274	0.9234	0.9258
	susy	0.7650	0.7671	**0.7675**	0.7633	0.7671	0.7681	**0.7685**	0.7666	0.7643	**0.7671**	0.7670	0.7655
150	higgs	0.6172	0.6157	**0.6271**	0.6223	0.6213	0.6153	**0.6261**	0.6225	0.6182	0.6135	0.6253	**0.6266**
	covtype7	**0.9250**	0.9232	0.9123	0.9123	**0.9341**	0.9249	0.9235	0.9221	0.9285	0.9273	0.9200	**0.9298**
	susy	0.7528	0.7578	**0.7667**	0.7389	0.7595	0.7646	**0.7666**	0.7432	0.7607	0.7645	**0.7646**	0.7274
200	higgs	**0.6006**	0.5973	0.5334	0.5245	**0.6011**	0.5976	0.5337	0.5196	0.5907	**0.5984**	0.5339	0.5472
	covtype7	**0.9236**	0.9190	0.9186	0.9110	0.9196	0.9195	0.9187	**0.9236**	0.9229	0.9206	0.9187	0.9080
	susy	0.7350	0.7357	**0.7364**	0.7191	0.7335	0.7323	**0.7378**	0.7293	**0.7432**	0.7334	0.7391	0.7411

Regarding runtimes, RUS and ROS have the best outperform due to the simplicity of their algorithms. Then follows understandably SMOTE-MR, due to the local nature of its approach. The partitioned data is processed locally in each Map, resulting in lower times. The SMOTE-BD presents the highest times of all the algorithms tested, using a distributed data approach, but giving an exact result. For instance, the runtime for SMOTE-MR versus SMOTE-BD for the higgs dataset, with 32 partitions, 150 % of oversampling and k equals to 5, is four times faster. It is evident that a compromise between the runtimes and the model approximation has to be considered. At last but not least, another important factor to point out is related with the data intrinsic quality, that has not been evaluated here, but it is possible to be affecting our results with redundancy and noise.

Table 5. TPR average results for the four methods over the datasets for 32, 64 and 128 partitions and for 100, 150 and 200 oversampling percentage.

Part.		32				64				128			
Perc.	Method Dataset	SMT-BD	SMT-MR	ROS	RUS	SMT-BD	SMT-MR	ROS	RUS	SMT-BD	SMT-MR	ROS	RUS
100	higgs	0.6126	0.6181	0.7534	**0.7608**	0.6114	0.6150	0.7610	**0.7783**	0.6209	0.6158	**0.7601**	0.7517
	covtype7	0.9396	0.9627	**0.9691**	0.9687	0.9622	0.9511	**0.9784**	0.9499	0.9416	0.9423	0.9773	**0.9812**
	susy	0.8511	0.8241	0.8131	**0.8536**	**0.8282**	0.8220	0.8045	**0.8282**	0.8153	**0.8221**	0.7825	0.8086
150	higgs	0.8074	0.8071	**0.8323**	0.4822	0.7909	0.8067	**0.8336**	0.4777	0.8024	0.8172	**0.8348**	0.4929
	covtype7	0.9802	0.9815	**0.9963**	0.8655	0.9835	0.9822	**0.9940**	0.8952	0.9750	0.9777	**0.9961**	0.9599
	susy	0.7069	0.7225	0.7553	**0.9154**	0.7333	0.7555	0.7445	**0.9087**	0.7378	0.7598	0.7375	**0.9294**
200	higgs	0.8527	0.8559	**0.9236**	0.3003	0.8520	0.8532	**0.9234**	0.2940	0.8549	0.8538	**0.9231**	0.3347
	covtype7	0.9922	0.9930	**0.9947**	0.8657	0.9901	0.9910	**0.9942**	0.8979	0.9844	0.9853	**0.9944**	0.8541
	susy	0.6411	0.6443	0.6456	**0.9387**	0.6376	0.6375	0.6489	**0.9240**	0.6706	0.6467	0.6526	**0.9063**

Table 6. TNR average results for the four methods over the datasets for 32, 64 and 128 partitions and for 100, 150 and 200 oversampling percentage.

Part.		32				64				128			
Perc.	Method Dataset	SMT-BD	SMT-MR	ROS	RUS	SMT-BD	SMT-MR	ROS	RUS	SMT-BD	SMT-MR	ROS	RUS
100	higgs	**0.6817**	0.6787	0.5747	0.5655	**0.6844**	0.6816	0.5658	0.5437	0.6764	**0.6811**	0.5663	0.5738
	covtype7	**0.9120**	0.8874	0.9046	0.9051	0.8896	**0.9054**	0.8712	0.9048	**0.9202**	0.9135	0.8724	0.8734
	susy	0.6878	0.7140	**0.7244**	0.6826	0.7105	0.7178	**0.7342**	0.7096	0.7173	0.7158	**0.7519**	0.7247
150	higgs	0.4721	0.4698	0.4725	**0.8031**	0.4882	0.4694	0.4703	**0.8111**	0.4763	0.4606	0.4685	**0.7965**
	covtype7	0.8729	0.8684	0.8354	**0.9617**	0.8874	0.8709	0.8580	**0.9499**	0.8844	0.8795	0.8497	**0.9006**
	susy	**0.8031**	0.7954	0.7783	0.5965	0.7873	0.7738	**0.7893**	0.6079	0.7847	0.7693	**0.7927**	0.5692
200	higgs	0.4230	0.4169	0.3081	**0.9161**	0.4242	0.4185	0.3085	**0.9183**	0.4085	0.4195	0.3087	**0.8946**
	covtype7	0.8598	0.8505	0.8484	**0.9586**	0.8541	0.8532	0.8489	**0.9501**	0.8654	0.8603	0.8488	**0.9654**
	susy	**0.8430**	0.8402	0.8398	0.5510	**0.8439**	0.8415	0.8388	0.5756	0.8237	0.8319	**0.8371**	0.6061

5   Conclusions and future works

In this work, a behavior analysis of the current preprocessing techniques for balancing Big Data was presented. Each solution use the MapReduce programming model through the Apache Spark framework, one of the most popular to deal with Big Data nowadays. Three of them were developed with a local approach and one with a global one. The main reason that motivated us was to evaluate if those methods perform as in the traditional data size problems.

Usually, in "small data" scenarios the performance of the SMOTE is better than the ROS and RUS proposals. Even though, our experiments results in Big Data do not show the same behavior. One cause could be the data quality. The current available big datasets may have presence of redundancy, noise, dispersion, among others factors which deteriorate the quality of the experimental results. Our intuition is that applying preprocessing techniques to balance the dataset is not enough, in which a previous data cleansing stage, may be necessary.

As future work, the development of hybrid models focused on the areas where resampling is specially needed will be carried out. In particular, the zones of interest are those with the presence of small disjuncts and overlapping. The proposal will prioritize the analysis of the local neighborhood of each minority class instance.

References

1. C.L. Philip Chen and Chun-Yang Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275:314 – 347, 2014.
2. Ronaldo C. Prati, Gustavo E. A. P. A. Batista, and Diego F. Silva. Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. *Knowledge and Information Systems*, 45(1):247–270, 2015.
3. Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.
4. Sergio Ramírez-Gallego, Alberto Fernández, Salvador García, Min Chen, and Francisco Herrera. Big data: Tutorial and guidelines on information and process fusion for analytics algorithms with mapreduce. *Information Fusion*, 42:51–61, 2018.
5. Diego García-Gil, Julián Luengo, Salvador García, and Francisco Herrera. Enabling smart data: Noise filtering in big data classification. *Inf. Sci.*, 479:135–152, 2019.
6. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over–sampling technique. *Journal of Artificial Intelligent Research*, 16:321–357, 2002.
7. A. Fernandez, S. Garcia, F. Herrera, and N.V. Chawla. Smote for learning from imbalanced data: Progress and challenges. marking the 15-year anniversary. *Journal of artificial intelligence research*, 61:863–905, 2018.
8. Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, 4th edition, 2015.
9. Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 15–28, San Jose, CA, 2012. USENIX.
10. H. Karau, A. Konwinski, P. Wendell, and M. Zaharia. *Learning Spark: Lightning-Fast Big Data Analytics*. O'Reilly Media, 1st edition, 2015.
11. Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Zadeh, Matei Zaharia, and Ameet Talwalkar. MLlib: Machine learning in apache spark. *Journal of Machine Learning Research*, 17(34):1–7, 2016.
12. Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, and Ion Stoica. Discretized streams: Fault-tolerant streaming computation at scale. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 423–438, New York, NY, USA, 2013. ACM.
13. Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, June 2004.

14. Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250(20):113–141, 2013.
15. A. Fernandez, S. del Rio, N. V. Chawla, and F. Herrera. An insight into imbalanced big data classification: Outcomes and challenges. *Complex and Intelligent Systems*, 3(2):105–120, 2017.
16. María José Basgall, Waldo Hasperué, Marcelo Naiouf, Alberto Fernández, and Francisco Herrera. Smote-bd: An exact and scalable oversampling method for imbalanced classification in big data. *Journal of Computer Science and Technology*, 18(03):e23, Dec. 2018.
17. SMOTE-BD Spark Package. https://spark-packages.org/package/majobasgall/smote-bd, 2018.
18. Jesús Maillo, Sergio Ramírez-Gallego, Isaac Triguero, and Francisco Herrera. knn–is: An iterative spark-based design of the k-nearest neighbors classifier for big data. *Knowledge-Based Systems*, 117:3–15, 2017.
19. SMOTE-MR source code. https://github.com/majobasgall/smote-mr, 2018.
20. A. Fernandez, F. Herrera, O. Cordon, M. Jose del Jesus, and F. Marcelloni. Evolutionary fuzzy systems for explainable artificial intelligence: Why, when, what for, and where to? *IEEE Computational Intelligence Magazine*, 14(1):69–81, Feb 2019.
21. M. Lichman. UCI machine learning repository, 2013.
22. Pablo D. Gutierrez, Miguel Lastra, Jose Manuel Benitez, and Francisco Herrera. SMOTE-GPU: Big data preprocessing on commodity hardware for imbalanced classification. *Progress in Artificial Intelligence*, 6(4):347–354, 2017.
23. Ricardo Barandela, José Salvador Sánchez, Vicente García, and E. Rangel. Strategies for learning in class imbalance problems. *Pattern Recognition*, 36(3):849–851, 2003.
24. J. Huang and C. X. Ling. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):299–310, March 2005.