Leevy *et al. J Big Data* (2018) 5:42
https://doi.org/10.1186/s40537-018-0151-6

Journal of Big Data

SURVEY PAPER Open Access

# A survey on addressing high-class imbalance in big data

Joffrey L. Leevy[1], Taghi M. Khoshgoftaar[1], Richard A. Bauder[1*] and Naeem Seliya[2]

*Correspondence:
rbauder2014@fau.edu
[1] Florida Atlantic University, Boca Raton, USA
Full list of author information is available at the end of the article

**Abstract**

In a majority–minority classification problem, class imbalance in the dataset(s) can dramatically skew the performance of classifiers, introducing a prediction bias for the majority class. Assuming the positive (minority) class is the group of interest and the given application domain dictates that a false negative is much costlier than a false positive, a negative (majority) class prediction bias could have adverse consequences. With big data, the mitigation of class imbalance poses an even greater challenge because of the varied and complex structure of the relatively much larger datasets. This paper provides a large survey of published studies within the last 8 years, focusing on high-class imbalance (i.e., a majority-to-minority class ratio between 100:1 and 10,000:1) in big data in order to assess the state-of-the-art in addressing adverse effects due to class imbalance. In this paper, two techniques are covered which include Data-Level (e.g., data sampling) and Algorithm-Level (e.g., cost-sensitive and hybrid/ensemble) Methods. Data sampling methods are popular in addressing class imbalance, with Random Over-Sampling methods generally showing better overall results. At the Algorithm-Level, there are some outstanding performers. Yet, in the published studies, there are inconsistent and conflicting results, coupled with a limited scope in evaluated techniques, indicating the need for more comprehensive, comparative studies.

**Keywords:** Big data, High-class imbalance, Data sampling, Cost-sensitive learners

## Introduction

Any dataset with unequal distribution between its majority and minority classes can be considered to have class imbalance, and in real-world applications, the severity of class imbalance can vary from minor to severe (high or extreme). A dataset can be considered imbalanced if the classes, e.g., fraud and non-fraud cases, are not equally represented. The majority class makes up most of the dataset, whereas the minority class, with limited dataset representation, is often considered the class of interest. With real-world datasets, class imbalance should be expected. If the degree of class imbalance for the majority class is extreme, then a classifier may yield high overall prediction accuracy since the model is likely predicting most instances as belonging to the majority class. Such a model is not practically useful, since it is often the prediction performance of the class of interest (i.e., minority class) that is more important for the domain experts [1]. He and Garcia [2] suggest that a popular viewpoint held by academic researchers defines imbalanced data as data with a high-class imbalance between its two classes, stating that high-class imbalance is reflected when the majority-to-minority class ratio ranges from

100:1 to 10,000:1. While this range of class imbalance may be observed in big data, it is not a strict definition of high-class imbalance. From the viewpoint of effective problem-solving, any class imbalance (e.g., 50:1) level that makes modeling and prediction of the minority class a complex and challenging task can be considered high-class imbalance by the domain experts [3]. It should be noted that we focus our survey investigation of published works on class imbalance in big data in the context of binary classification problems, since typically non-binary (i.e., multi-class) classification problems can be represented using a sequence of multiple binary classification tasks.

High-class imbalance, often observed in big data, makes identification of the minority class by a learner very complex and challenging because a high-class imbalance introduces a bias in favor of the majority class. Consequently, it becomes quite difficult for the learner to effectively discriminate between the minority and majority classes, yielding a task comparable to searching for the proverbial needle in a haystack, especially if the class imbalance is extreme. Such a biased learning process could result in the classification of all instances as the majority (negative) class and produce a deceptively high accuracy metric. In situations where the occurrence of false negatives is relatively costlier than false positives, a learner's prediction bias in favor of the majority class could have adverse consequences [4]. For example, among several patients with suspicious mole(s) pigmentation (melanocytic naevi), very few are likely to have melanoma cancer, i.e., minority class, while most are likely not to have melanoma cancer, i.e., majority class. Here, a false negative implies a patient with cancer is misclassified as not having the disease, which is a very serious error. In contrast, a false positive implies a patient without cancer is classified as having the disease, which is comparatively (to a false negative) not a serious error. In this example, the real-world class imbalance is observed to be very high, thus making the issue of class imbalance into a problem of great significance in predictive learning. Class imbalance observed in the given dataset can be considered intrinsic or extrinsic [2], where intrinsic-based class imbalance reflects the organic data distribution characteristics of the given domain and extrinsic-based class imbalance reflects influences of external factors such as time and storage associated with domain data. The problem of distinguishing between 1000 spam emails from 1,000,000 non-spam emails is an example of intrinsic class imbalance, given that most emails are non-spam. In contrast, a domain where data is transmitted and collected using a sequential stream could lead to extrinsic class imbalance if the data streaming and collection is interrupted due to external factors specific to the domain, e.g., lack of data storage capacity, time-based data collection rules, etc. It should be noted that in this survey paper, we do not distinguish between published works that focus on either intrinsic class imbalance or extrinsic class imbalance.

Specific properties are used to comprehend and define big data, including volume, variety, velocity, variability, value and complexity. Katal et al. [5] state that these properties, notably associated with big data, make modeling and analysis of such data with traditional methods difficult. More specifically, traditional methods may have difficulty coping with high volume data, the diversity of data formats, the speed of data coming from various sources, inconsistencies of data flows, the filtering of important data, and the linking and transforming of data [5]. To differentiate between non-big-data and big-data, we refer to the former as traditional data in this paper. An example of traditional

**Table 1 Categories of methods addressing class imbalance**

| Data-Level methods | | | Algorithm-Level methods | |
|---|---|---|---|---|
| Data-sampling | | Feature selection | Cost-sensitive | Hybrid/ensemble |
| Over-sampling | Under-sampling | | | |

data could be a dataset of 5000 instances collected over a period of 1 month for a small company, where each instance represents the front door access record of an employee of the company. In contrast, an example of big data could be a dataset of several million or more weather forecast reference points for collecting real-time weather data, or Medicare claims records collected from providers over several years [6]. The increasing reliance on big data applications worldwide makes a case for developing effective and efficient ways to extract knowledge from this type of data. Class imbalance plagues traditional data as well as big data; however, with the latter, the mal-effects can be felt much more severely due to the extreme degrees of class imbalance [7].

The goal of our survey paper is to summarize current research work on high-class imbalance issues in big data. The scope of our study is investigating works conducted within the past 8 years (i.e., 2010–2018) that focus on the problem junction of big data and class imbalance, and the consequent solutions developed by researchers. Moreover, in the interest of our focus on big data only, we only consider relevant works that analyze (class imbalance in big data) at least one dataset consisting of 100,000 instances or higher. In addition to analyzing the surveyed papers, we also provide our own insights into likely gaps in current research in the area and discuss avenues for future work for the community. To the best of our knowledge, we have included all published articles that fall within our survey study's scope. We believe such a large-scale survey of works addressing, and developing solutions for, high-class imbalance problems in big data is unique in the data mining and machine learning domain.

In related literature, the strategies for tackling class imbalance problems are similar for both traditional data and big data. Ali et al. [8] categorize approaches for addressing class imbalance into those conducted at the Data-Level or at the Algorithm-Level, where both categories include approaches used for both traditional data and big data (see Table 1), i.e., data sampling (under-sampling and over-sampling), feature selection, cost-sensitive methods, and hybrid/ensemble techniques. Data-Level methods include data sampling and feature selection approaches, while Algorithm-Level methods include cost-sensitive and hybrid/ensemble approaches. This categorization is further explained in the next section.

We observed some interesting trends/results based on our investigation of the surveyed works, and some key findings are summarized next. Among the Data-Level methods, empirical results of relevant works generally suggest that Random Over-Sampling (ROS) yields better classification performance than Random Under-Sampling or the Synthetic Minority Over-Sampling Technique (SMOTE). Moreover, with the MapReduce environment for big data analysis with data sampling, the process of determining the preferred balance between the data over-sampling percentage and classification performance is an empirical parameter/process, instead of a formulaic solution. At the Algorithm-Level, there are a variety of methods that seemingly provide good classification

performance for big data with high-class imbalance. For the cost-sensitive techniques, our discussion includes a fuzzy rule-based classification approach [9, 10] and an online learner scheme [11, 12]. For the hybrid/ensemble techniques, our discussion includes a Bayesian Optimization Algorithm that maximizes Matthew's Correlation Coefficient by learning optimal weights for the positive and negative classes [13], and an approach that combines Random Over-Sampling (ROS) and Support Vector Machines (SVMs) [14].

One of the primary problems we encountered during our surveyed works was that the MapReduce big data framework was observed to be quite sensitive to high-class imbalance [15], primarily due to the adverse effects of creating multiple partitions within the already very small minority class space. Hence, we suggest a greater focus on a more flexible computational environment for big data analysis, such as Apache Spark [16], for addressing the high-class imbalance problem. Another key issue plaguing big data is small disjuncts (described later in the paper) of data points within the overall dataset or within each of the two classes, and based on our survey, we note that this issue has not been given enough focus in the context of high-class imbalance in big data. In addition, given the problem's relatively poor maturity in developed effective solutions, considerably more research and empirical investigation still remain to be conducted. Many studies we investigated in this paper generally lacked sufficient depth in the scope of their empirical investigation of the high-class imbalance problem in big data. This finding makes it difficult to conclude whether one approach is more effective and efficient than another.

The remainder of this article is organized as follows. In "Methods addressing class imbalance in traditional data" section, we provide an overview of strategies and methods for handling traditional data with the class imbalance problem. While the primary focus of this paper is on high-class imbalance in big data, we present "Methods addressing class imbalance in traditional data" section to provide the reader with a more complete picture of existing approaches for class imbalance, since similar methods are generally used for both traditional data and big data. In "Methods addressing class imbalance in big data" section, we discuss the Data-Level methods and Algorithm-Level techniques for handling big data defined by high degrees of class imbalance. In "Discussion summary of surveyed works" section, we provide our insights into existing problems that still need focus (or more focus) in the context of effective solutions for big data with high-class imbalance. In "Conclusion" section, we conclude with the main points of our paper and suggest some directions for future work.

## Methods addressing class imbalance in traditional data

The two general categories of methods that address the class imbalance problem are Data-Level methods and Algorithm-Level methods, as noted by Ali et al. [8]. Each of the two categories can be further sub-divided into groups, as shown in Table 1.

Data-Level methods can be further sub-grouped into data-sampling methods and feature selection methods. Over-sampling methods and under-sampling methods form the two sub-groups of the data-sampling methods group, where data sampling from the given dataset is done either randomly or using a specified formulaic/algorithmic approach [17, 18]. During the over-sampling process, instances from the minority class are added (via replication) to the given dataset, where the replication is done either

randomly or using an intelligent algorithm. In contrast, during the under-sampling process, instances from the majority class are removed from the given dataset, where the removal is largely done randomly (as seen in related literature). Feature Selection Methods, while largely used only (without consideration to class imbalance) to improve classification performance [19, 20], may also help select the most influential features (or attributes) that can yield unique knowledge for inter-class discrimination. This leads to a reduction of the adverse effects of class imbalance on classification performance [21–23].

Algorithm-Level methods can be further sub-grouped into cost-sensitive methods and hybrid/ensemble methods. The former works on the general principal of assigning more weight to an instance or learner in the event of a misclassification, e.g., a false negative prediction may be assigned a higher cost (i.e., weight) compared to a false positive prediction, given the latter is the class of interest. Ensemble methods can also be used as cost-sensitive methods, where the classification outcome is some combination of multiple classifiers built on the dataset; Bagging and Boosting are two common types of ensemble learners [24, 25]. Bagging minimizes the predictive variance by producing several training sets from the given dataset, with a classifier being generated for each training set and then their individual models combined for the final classification. Boosting also uses several training sets from the given dataset, and after iteratively assigning different weights to each classifier based on their misclassifications, a weighted approach combining the individual classifier's results yields the final classification. Hybrid methods are designed to remedy known problems arising from the data-sampling methods, feature selection methods, cost-sensitive methods, and basic learning algorithms such as Naive Bayes [26]. In some instances, sub-groups of Data-Level methods or Algorithm-Level methods may be combined into an overall approach to address the class imbalance problem. For example, the popular Random Forest (RF) classifier is a version of the original Random Decision Forest [27] algorithm, and is an ensemble learner which also implements Bagging. In contrast, the original Random Decision Forest is not considered an ensemble learner [28].

#### Data-sampling methods for class imbalance

The common data-sampling approaches for handling class imbalance include: Random Over-Sampling (ROS), Random Under-Sampling (RUS), and Synthetic Minority Over-Sampling Technique (SMOTE) [29, 24, 30]. While ROS and RUS are relatively simpler methods, their usefulness in addressing class imbalance should not be overlooked, in lieu of more complex methods such as SMOTE [31]. It is our opinion that different methods should be investigated for the given domain's dataset to address class imbalance, as there is no one universal best approach. SMOTE is an intelligent over-sampling technique that forms new minority class instances by interpolating between several minority class instances that lie relatively close to each other [32]. The creation of new minority class instances leads to a reduction in the class imbalance degree compared to the original majority-to-minority class ratio. In over-sampling techniques, an important concern is overfitting by the learner (i.e., leading to poor generalization performance) and also an increase of the training dataset size. Fernandez et al. [32] suggest that overfitting is not a serious issue for SMOTE since it synthetically creates new instances, compared to

duplication of existing instances. In the case of under-sampling techniques, an issue of concern is the deletion of valuable information if a relatively large number of instances are removed from the majority class. A substantial removal of instances from the majority class may also alter the distribution of the majority class (and thus overall dataset) that is representative of the given domain. It is therefore important to consider different data sampling methods for class imbalance.

#### Feature-selection methods for class imbalance

In our literature review for this survey paper, it was observed that feature-selection methods for addressing class imbalance is a largely unexplored research area. Ali et al. [8], while also noting the research gap in using feature selection for class imbalance, warn that extra computational cost can be an issue of concern. Mladenic and Grobelnik [22] utilized a feature-subset selection approach developed for a Naive Bayes classifier on imbalanced text data from multiple domains. A unique characteristic of the Naive Bayes learner is that it inherently assumes inter-attribute independence in the context of the given class attribute [26]. Their empirical investigation compared 11 different feature scoring measures, and determined that the odds ratio (and its variants) produce the best results [22]. The odds ratio is commonly used for information retrieval tasks, where documents are ranked according to positive or negative associations between words. The proposed Naive Bayes classifier adopts the same conditional probability technique used by the Odds ratio for scoring features. The authors also conclude that considering domain and algorithm characteristics significantly improves classification results.

Zheng et al. [23] investigate feature selection for text categorization with imbalanced data. Their approach selected the positive features and negative features separately using feature selection techniques (including Information Gain, Chi Square, correlation coefficient, and odds ratio), and then explicitly combined. They also present variations of the odds ratio and Information Gain metrics to especially address class imbalance. Their study used the Naïve Bayes and Regularized Logistic Regression as classifiers, and the proposed approach yielded good results.

Yin et al. [21] demonstrated that both decomposition-based and Hellinger's distance-based methods can outperform existing feature-selection methods for imbalanced data. Decomposition methods partition the majority class into smaller pseudo-subclasses and assign the pseudo-class labels, while Hellinger's distance is a measure of distribution divergence. There are three phases in a decomposition-based framework. In Phase 1, K-means clustering is used to decompose the majority class into relatively balanced pseudo-subclasses. Class instance labels are then changed to subclass labels provided by the clustering step. In Phase 2, the goodness of each feature is measured with the pseudo-labels and the traditional measurement of the goodness of each feature. The features are then ranked according to the goodness based on the calculated scores. The top $k$ good features are selected, where $k$ is a user-defined parameter, and the pseudo-labels are released back to the original labels. In Phase 3, the classification task is performed. The Hellinger's distance allows the notion of "affinity" to be captured between the probability measures on a finite event space. For example, let P and Q denote two probability measures that are continuous distributions with respect to a third probability measure. If $P = Q$, then $distance = 0$ (maximal affinity) and if P and Q are completely disjoint then

$distance = \sqrt{2}$ (zero affinity). A minimal affinity means that the given feature is most discriminative between classes. The higher the *distance* (i.e., lower affinity) value, the better the corresponding feature. Thus, the Hellinger's distance can be used to measure the prediction power of features to classify instances.

#### Cost-sensitive methods for class imbalance

Several studies using cost-sensitive learning for minimizing class imbalance have been performed, and here we present some of the most relevant ones. Cao et al. [33] introduced an effective wrapper framework incorporating classification performance measures, such as area under the receiver operating characteristic curve (AUC) and geometric mean (GM), directly into the objective function of a cost-sensitive SVM. SVMs find the hyperplane that maximizes the margin used to separate two classes, and the vectors defining the hyperplane are called support vectors. The introduction of the wrapper framework improved the performance of classification by simultaneously optimizing the best feature subset, intrinsic parameters, and misclassification cost parameters [33]. In a related study, Cao et al. [34] used an objective function of cost-sensitive artificial neural networks (PSOCS-NN) instead of a cost-sensitive SVM [33], where the optimization is based on Particle Swarm Optimization (PSO) [35]. Neural networks are data-driven, self-adaptive models that attempt to mimic the operations of neurons in the human brain [36]. Their empirical results showed that PSOCS-NN was superior in most cases compared to using a basic neural network learner with RUS, SMOTE [29], and SMOTEBoost [30]. The proposed learner was also generally better than SMOTE combined with a non-optimized cost-sensitive neural network (SMOTE + CS-NN). In addition, among all methods examined, RUS had the worst classification performance.

Lopez et al. [37] conducted a study on class imbalance with a comparison of cost-sensitive learning with over-sampling, and it was determined that neither technique outperformed the other. The study used two over-sampling techniques, SMOTE and SMOTE + ENN, where ENN is the Wilson's Edited Nearest Neighbor rule. For the cost-sensitive learners, the authors study several modifications to the C4.5, SVMs, k-NN, and Fuzzy Hybrid Genetics-Based Machine Learning methods, the combination of which is carried out through a wrapper classifier that uses these cost-sensitive approaches. In SMOTE + ENN, after SMOTE is performed, ENN is used to remove any instances misclassified by its three nearest neighbors in the training dataset. It should be noted that if the real cost matrix cannot be obtained for the cost-sensitive learners, an artificial cost matrix may have to be generated, leading to computational and empirical process overhead. However, it is often the case that the real cost matrix is unavailable because there are a significant number of factors to consider [8].

#### Hybrid/ensemble methods for class imbalance

A hybrid method for addressing class imbalance may include two or more individual methods used for addressing the class imbalance problem, or may use multiple algorithms for a specific part of the overall solution. Among the hybrid methods in published works, many are centered around SVM, Artificial Neural Networks, and Decision Trees [8]. A decision tree is a classifier that is modeled on a tree-like structure of internal nodes, branches, and terminal nodes (class labels) [38]. Hybrid approaches have the

burden to ensure that the differences in the individual approaches properly complement each other as a whole, and together yield better performance compared to the individual methods alone.

Akbani et al. [39] present an algorithm that combines SVM with a variant of SMOTE combined with an error cost algorithm to overcome the sub-optimal performance of regular SVM with severely imbalanced data. The authors compare their hybrid approach with RUS, SMOTE, and SVM, and conclude that it outperforms them all. Tang et al. [40] present GSVM-RU, which is a variation of their previously proposed GSVM learner (granular computing-based learning framework that uses SVM), that incorporates under-sampling to address the problems SVM faces with working with severely imbalanced data. The authors compare their approach to three other SVM-based hybrid learners: SVM-Weight, SVM-SMOTE, and SVM-RANDU, and demonstrate that, on the average, classification performance of GSVM-RU is better than the other methods. Ahumada et al. [41] propose a clustering-based technique combined with an SVM classifier to address class imbalance. The clustering phase recursively splits the majority class instances into two groups, until the resulting datasets (sub-problem) are balanced or are relatively easy to discriminate. The outcome of clustering is a directed acyclic graph (decision tree). An SVM classifier is subsequently fitted to each sub-problem. The authors demonstrate that their approach outperforms ROS in most cases. The two works presented above are hybrid approaches that use a cost-sensitive learning aspect in their approach.

In the case of Ensemble Methods designed to address the class imbalance problem, generally Bagging, AdaBoost, and Random Forest are popular approaches [42, 43]. Several variants of Bagging have arisen from the original technique, such as Asymmetric Bagging, SMOTEBagging, ROSBagging, and RUSBagging. Adaptive Boosting (or AdaBoost) takes an iterative Boosting approach with the general goal of improving the classification performance of weak learners; some of its variants include RUSBoost [24] and ROSBoost [42]. A commonly used classifier, Random Forest is comprised of a bagging technique and random feature subspace selection that grows each tree in a Decision Forest [27]. Balanced Random Forest and Weighted Random Forest are two variants of Random Forest that have been proposed to use RF to address the problem of class imbalance [44]. Galar et al. [45] suggest that under class imbalance conditions ensemble-based techniques generally produce better classification results than data sampling methods. However, the authors also conclude that SMOTEBagging, RUSBoost, and UnderBagging outperform other ensemble classifiers. Among these three methods there was no statistical difference; however, SMOTEBagging yielded slightly better classification performance. As mentioned in the case of hybrid methods, it should be noted that ensemble methods (especially variants of basic ensemble learners) also have the burden to ensure that the differences in the individual approaches properly complement each other, and together yield better performance compared to the individual methods alone.

### Methods addressing class imbalance in big data

The strategies for tackling class imbalance are generally the same for traditional data and big data, and as noted in Table 1, these strategies are exercised at either the Data-Level or Algorithm-Level in their solutions. The key differences are influenced by the unique

characteristics of big data, as discussed in "Introduction" section. We reiterate that our survey focuses on published works addressing the high-class imbalance problem(s) often observed in big data domains. Prior to discussing the relevant works, a brief discussion on big data processing is presented next.

The processing and analysis of big data often requires specialized computational frameworks and environments that utilize computing clusters and parallel algorithms. The more popular computational frameworks for working with big data include Apache Spark [16], MapReduce (original), and Apache Hadoop [46]. MapReduce generally divides the original dataset into subsets which are relatively easier to process, and then combines the multiple partial solutions that are obtained in determining the final outcome. Apache Hadoop is an open source implementation and variant of MapReduce. Apache Spark performs faster distributed computing of big data by using in-memory operations instead of the divide-and-conquer approach of MapReduce [16]. Although Spark can run on Hadoop, this is generally not a requirement. Apache Mahout, which contains various implementations of classification models, is an open-source machine-learning library that can run on Apache Hadoop or Apache Spark [46, 47]. It is a distributed linear algebra framework and mathematically expressive Scala programming language designed for quick implementation of algorithms.

If real-time solutions are of importance, one may wish to consider Apache Storm or Apache Flink instead, since they offer true stream processing while Spark's use of micro-batch streaming may have a small lag associated with it in receiving results [48]. Flink offers the best of both worlds in this regard, with a combination of batch and true stream processing, but it is a very young project and needs more research into its viability. Additionally, it does not currently support nearly as many machine learning solutions as the other platforms. No distributed machine-learning libraries have the same amount of options as some of the non-distributed tools such as Weka [49] because not every algorithm lends itself well to parallelization [48]. Mahout and MLLib[1] provide the best-rounded big data libraries in terms of algorithm coverage and both work with Spark. MLlib has a wider overall selection of algorithms and a larger and more dedicated team working on it, but is relatively new and largely unproven.

The choice of tools will largely depend on the applications they are being used for as well as user preferences [48]. For example, Mahout and MLlib include options for recommendations, so if the intended application is an e-commerce site or social network, one may wish to choose from them for features such as item or user suggestions. Social Media or Internet of Things data may require real-time results, necessitating the use of Storm or Flink along with their associated machine-learning libraries. Other domains such as Healthcare often produce disparate datasets that may require a mix of batch and streaming processing, in which case Flink or Spark could also be used.

#### Data-level methods for class imbalance in big data

Fernandez et al. [32] examine class imbalance in big data, focusing on problems with the MapReduce framework. The authors state that the lack of data and small disjuncts issues

[1] https://spark.apache.org/docs/latest/ml-guide.html.

observed in big data are accentuated within MapReduce. They conduct an experiment comparing RUS, ROS, and SMOTE for class imbalance in big data using MapReduce with two subsets, extracted from the ECBDL14 dataset [50] and maintaining the original class ratio. The two subsets respectively consisted of 12 million and 600,000 instances, a 98:2 class ratio, and 90 features. The original 631 features of the ECBDL14 dataset were reduced to 90 features by applying a feature selection algorithm [3, 50]. The experiment examined performances of Random Forest (RF) and Decision Tree (DT) learners, using both Apache Spark and Apache Hadoop (MapReduce) frameworks. The key conclusions are: SMOTE performed worse than RUS or ROS; RUS performed better with less partitions (Maps); ROS performed better with higher partitions; Apache Spark based RF and DT yielded better results with RUS compared to ROS; number of partitions in Hadoop had significant influence on obtained performance; models using Apache Spark generally yielded better results compared to when using Hadoop. The best overall values of geometric mean (GM) for ROS, RUS, and SMOTE were 0.706, 0.699, and 0.632, respectively (Table 3, Appendix A1). Our general analysis of Fernandez et al. [32] includes the following:

- The focus of this work was more on demonstrating limitations of MapReduce than on developing an effective solution for the high-class imbalance problem in big data.
- Different big data frameworks were used for some data-sampling methods, making comparative conclusions unreliable, i.e., SMOTE implementation is done in Apache Hadoop, while RUS and ROS implementations are done in Apache Spark.
- It is not clear what sampling ratios (90:10, 75:25, etc.) were used with RUS, ROS, and SMOTE, limiting the study without investigating the impact of various sampling ratio values of classification performance; impact of reducing the number of features to 90 (from 631) on the various experiments is not discussed.
- The experimental results lack statistical validation; limiting classification performance evaluation to GM, compared to AUC which is considered a more robust performance metric.
- The empirical conclusions were based on analyzing only one dataset, i.e., ECDBL14.

Moreover, the problems associated with MapReduce have been known to the big data community [15, 16], as well as the knowledge that in comparison Apache Spark is significantly more efficient. Some key problems of MapReduce include: the division of the already very small minority class space reduces the number of positive class instances available in each partition for effective analysis; and, it seems to be adversely sensitive to the presence of disjunct data spaces in the overall data or within each class.

Rio et al. [51] utilized the Apache Hadoop framework to investigate MapReduce versions of RUS and ROS with the MapReduce version of the RF classifier (from the Apache Mahout library). The ECBDL14 dataset is used as the big data case study (from the bioinformatics domain), and the MapReduce approach for Differential Evolutionary Feature Weighting (DEFW-BigData) algorithm was utilized for detecting the most important features [3]. The dataset consisted of approximately 32 million instances, a class ratio of 98:2, and 631 features. The RF parameters were set to: 192 number of trees, 10 (and 25) number of features to build the trees, and unlimited depth of the trees.

Their initial empirical results demonstrate that ROS performed slightly better than RUS, based on using the product of true positive rate (TPR) and true negative rate (TNR), i.e., TPR × TNR. The best overall values for ROS and RUS were 0.489 and 0.483, respectively (Table 3, Appendix A1). However, the authors noted that ROS suffered from very low TPR compared to TNR. Hence, they experimented with a range of higher over-sampling ratios for ROS combined with the DEFW-BigData algorithm to select the top 90 features based on the weight-based ranking obtained. An increase in the over-sampling rate increased the TPR rate and lowered the TNR, and the best results for this experimentation were obtained with an over-sampling rate of 170%. Rio et al. [51] generally has similar limitations that were discussed for Fernandez et al. [32], but in addition, it has the following issues: the number of features for building trees seem too small compared to available 631 features; there is no clear indication why the authors selected the top 90 features; since the ECBDL14 data is used, a comparison with the other studies using that data would add value to the findings presented; inclusion of the popular SMOTE data-sampling algorithm is missing; MapReduce, which is known to be sensitive to high-class imbalance is used (as noted by multiple works surveyed in our study), instead of the more efficient Apache Spark; and, the study seems like a subset of Fernandez et al. [51].

Tsai et al. [15] compared the performance of a cloud-based MapReduce framework with a distributed learning framework based on the data parallelism paradigm, as well as with a baseline single machine framework. The authors evaluated four datasets, with two of them being binary class datasets and the other two being non-binary class datasets. We summarize their results on the two binary class datasets, Breast Cancer and Protein Homology. The former dataset consisted of 102,294 instances, 117 features, and an approximate class ratio of 99:1. The Protein Homology dataset consisted of 145,751 instances, 74 features, and an unstated class ratio value. SVMs were used by all three approaches, and the given dataset was divided into training and test datasets using a 90:10 split. The cloud-based MapReduce framework has a mapping function that handles filtering and sorting, a shuffle function that reassigns data based on the keys outputted by the mapping operation, and a reduce function that parallelizes the computations done on each data group. The procedure was implemented by a computer server using different settings of 1, 10, 20, 30, 40, and 50 virtual machines (computer nodes) to train the SVM classifier. The distributed learning framework, using a divide-and-conquer approach, was based on dividing the training dataset into 10, 20, 30, 40, and 50 subsets, with one computer node associated with one specific subset for the training task. The baseline single machine framework simply implemented the SVM learner on one computer using centralized data. For the Breast Cancer data, classification accuracy of both the baseline and distributed frameworks was 99.39%, while the MapReduce framework yielded only 58% classification accuracy. For the Protein Homology dataset, all three approaches produced similar classification accuracy, around 99% with a relative difference of less than 0.12% among the three approaches. However, as the number of nodes was increased from 10 to 30, classification accuracy for the MapReduce and distributed frameworks fell slightly, but were relatively similar to each other with 30–50 nodes. Our analysis of Tsai et al. [15] brings out these salient points: accuracy percentage is not a good indicator of classification performance since it provided no information on the TPR and TNR values of the SVM learners. A highly imbalanced dataset, for example,

could have a 99.5% accuracy rate with a TNR of nearly 100% and TPR of almost 0%; comparing the baseline single machine and distributed learning frameworks with Apache Spark would have provided more valuable results, since Spark is significantly better than MapReduce; with classification accuracies over 99% (except for MapReduce framework with Breast Cancer data), the implementation of a cloud-based big data framework may be irrelevant for the case study datasets, perhaps since the dataset sizes were too small compared to other big data works examined in our survey; and, data-sampling, cost-sensitive learning, or other commonly used techniques for addressing class imbalance in big data were not used in the comparative study.

Triguero et al. [16] examine Evolutionary Under-Sampling (EUS) in cases of severe class imbalance in big data, based on the initial knowledge that EUS had shown promise in addressing class imbalance in traditional data [52]. The authors implement the EUS approach with the Apache Spark framework, and compare it with their previous implementation of EUS with the Apache Hadoop (MapReduce) framework [53]. The base learner in both implementations is the C4.5 decision tree learner which is incorporated into the overall class balancing and classification process. EUS provides a fitness function for a prototype selection method, where the fitness function aims to find the proper balance between reduction (under-sampling) of training data instances and classification performance [52]. The authors recognized that divide-and-conquer methods based on MapReduce can potentially be affected by a lack of density from the minority class in the subsets created. Consequently, the in-memory operation of Apache Spark is modified such that the majority and minority class instances can be independently managed. This enables a relatively higher number of minority class instances to be retained in each subset created. The case study data was comprised of variants of two big data sets, i.e., ECDBL'14 and KDD Cup 1999 datasets. Two subsets, 50% and 25%, of the ECBDL'14 data were used, each with 631 features and a class ratio of 98:2. Three variants of the KDD Cup 1999 data were used, based on different binary class combinations (DOS vs. PRB; DOS vs. R2L; and, DOS vs. U2R), each with 41 features and the approximate class ratio values of 95:1, 3450:1, and 74,680:1 for the three datasets. The key results observed in the paper include: Apache Spark framework had shorter runtimes than Apache Hadoop; EUS performed better than RUS, but as expected, its runtime was much longer than that of RUS. The best overall values of GM classification performance metric for EUS and RUS were 0.672 and 0.662, respectively (Table 3, Appendix A1). The study unfortunately does not present a complete picture of the promising classification performance of EUS for high-class imbalance in big data. Regardless of its expected slow runtime, the author's EUS implementations with Apache Spark and Apache Hadoop could have been compared with other commonly used strategies, e.g., SMOTE and ROS, ensemble methods, cost-sensitive methods, etc. For example, Apache Spark may be able to execute a task 10 times faster than Apache Hadoop (MapReduce) via one strategy, but 100 times faster via another strategy. In addition, general research on increasing runtime of evolutionary computing methods should be investigated for incorporation into approaches presented by Triguero et al. [16, 53].

Park et al. [54] develop an overall big data analysis scheme for predicting traffic accidents on the road, including data collection, data processing involving data-sampling, and classification modeling using the Apache Hadoop (MapReduce) framework. Given

the context of our survey, we do not discuss their data collection approach, and instead focus on the class imbalance strategy investigated for the big data obtained from the periodic traffic reports. Within the Apache Hadoop framework, the authors implement a MapReduce modification of SMOTE for tackling the severely imbalanced traffic accidents dataset, i.e., class ratio of approximately 370:1, and a total of 524,131 instances characterized by 14 features. Subsequent to the over-sampling approach, the minority class (accident) instances in the training dataset went from 0.27 to 23.5% of the dataset. The Logistic Regression [55] learner yielded a classification accuracy of 76.35% (Table 3, Appendix A1) and the TPR was 40.83%, both of which the authors state as comparable to results presented by other investigators in the domain. Park and Ha [56] also experimented with SMOTE in a MapReduce framework (Apache Hadoop) and obtained optimal classification accuracy when the minority class reached about 30% of the training dataset (from the initial 0.14% of minority class instances). The initial training dataset consisted of 1,024,541 instances, a class ratio of 710:1, and 13 predictive features. Based on the Logistic Regression learner, classification accuracy obtained was approximately 0.806 (Table 3, Appendix A1). Our general analysis of the studies presented in [54, 56] are: MapReduce is known to have considerable sensitivity to high-class imbalance in a dataset, thus likely yielding sub-optimal classification performance; implementing the class imbalance scheme within the Apache Spark framework may yield better performance; SMOTE used within the MapReduce framework tends to perform sub-par, as descriptively pointed out by Fernandez et al. [32]; classification accuracy is generally not a dependable performance metric for classifiers; the study does not determine which (domain-specific) method is optimal for addressing high-class imbalance in contrast to SMOTE, e.g., RUS, ROS, EUS, Ensemble Methods, etc. A collective look at the works of Fernandez et al. [51], Park et al. [54], and Park and Ha [56], suggests an interesting observation in our study. The study presented in [51] obtained an optimal class balance for ROS with an over-sampling rate of 130%, which increased to 170% when ROS was modified to incorporate the DEFW-BigData algorithm [3, 50]. Whereas, experiments presented in [54, 56] indicated extremely high over-sampling percentage values for obtaining the best classification accuracy. Thus, determination of the preferred balance between classification accuracy (or performance) and the over-sampling rate requires an empirical and observational approach, instead of a formulaic approach.

Chai et al. [57] investigate high-class imbalance in big data in the context of statistical text classification within the medical records domain. The authors apply RUS to achieve equal class balance between the majority and minority classes, i.e., 50:50, with the goal of comparing classification performances between the original severely imbalanced dataset and the balanced dataset. Subsequent to some typical data cleansing and processing associated with text data, the training dataset was obtained which consisted of almost 516,000 instances, 85,650 features, and approximately 0.3% of instances forming the minority class. Regularized Logistic Regression is used as the underlying classifier, primarily because of its ability to avoid model overfitting while using a very large set of features that is typical in text classification. Empirical results indicated that the F1 score (F-measure) were relatively similar with or without under-sampling, i.e., the balanced dataset had no influence on the classification performance. However, under-sampling increased Recall and decreased Precision of the learner. The best overall value

of the F-measure was 0.99 (Table 3, Appendix A1). Our general analysis of Chai et al. [57] includes the following salient points: no clear indication is given as to why an equal class ratio with under-sampling is preferred for optimal classification performance, and this may be a significant drawback of the study; no clear explanation is provided on the selection of under-sampling over other relatively simple data-sampling methods, such as over-sampling; the case study dataset suffers from severe class imbalance, but it is not clear why the authors chose not to consider varying degrees of class ratios (90:10, 75:25, etc.) to study which imbalance ratio yields good classification performance.

#### Algorithm-Level methods for class imbalance in big data

Lopez et al. [9] propose the Chi-FRBCS-BigDataCS algorithm, a Fuzzy Rule-Based Classification System (FRBCS) that is able to deal with uncertainty involved in large volumes of data without ignoring the learning of the minority class, i.e., class imbalance in big data. Based on the classical FRBCS developed by Chi et al. [10], the proposed algorithm is a linguistic cost-sensitive FRBCS that is incorporated with the MapReduce framework using Apache Hadoop. To address class imbalance in big data, the proposed algorithm modifies the basic FRBCS approach as follows: first, the FRBCS method is adapted to follow the MapReduce principles that direct a distribution of the work on several processing nodes; and second, cost-sensitive learning modifications are applied to the FRBCS method. For the latter, a new rule weight computation is developed, i.e., Penalized cost-sensitive Certainty Factor (PCF-CS) [9]. The case study data involves class-based variants of the KDD Cup 1999 dataset, including DOS vs. Normal, DOS vs. R2L, and Normal vs. R2L binary class combinations, with each dataset consisting of 41 features. For each of these three binary class datasets, the following percentages were evaluated as training data using fivefold cross-validation: 10%, 25%, 40%, 50%, 60%, 75%, and 100%. These dataset sizes ranged between approximately 100,000 instances and 4,900,000 instances, with class ratio values varying from 80:20 to 74,680:1. The study also examines variants of the Record Linkage Comparison Patterns (RLCP) and the Poker Hand dataset, both obtained from the UCI Machine Learning Dataset Repository.

The authors of [9] compared their proposed Chi-FRBCS-BigDataCS algorithm with other FRBCS methods, including: Chi-FRBCS (the classical fuzzy rule-based classifier); Chi-FRBCS-CS (a proposed Chi-FRBCS version that introduces cost-sensitive learning by modifying some of the Chi-FRBCS operations); and Chi-FRBCS-BigData (the classical Chi-FRBCS version adapted to deal with big data, adopting a MapReduce design implemented under the Apache Hadoop framework). The empirical results indicated that Chi-FRBCS-BigDataCS produced the best classification performance with an AUC of 0.99 (Table 4, Appendix A2). Some key shortcomings of Lopez et al. [9] include: comparison of the proposed Chi-FRBCS-BigDataCS algorithm is limited only to the original and iteratively-modified versions of the classical FRBCS method, and no other existing cost-sensitive approaches for handling high-class imbalance in big data; similar to previously mentioned statements, implementing the Chi-FRBCS-BigDataCS algorithm within the Apache Spark framework may yield better performances and also demonstrate better tolerance to high-class imbalance, which is often noted in MapReduce; the computational cost and runtime of the proposed FRBCS variant is not discussed;

and, such a study could benefit from comparisons with non-cost-sensitive methods for addressing class imbalance.

In addition to high volume, high velocity, and high dimensionality, big data stream (online learning) categorization also suffers from high sparsity and high-class imbalance in the data. Utilizing batch learning for online data classification has drawbacks of not only requiring very large storage capacities, but also costly re-training for new data and the unavailability of all training data instances. This is largely because, in real-world data stream schemes the data arrives rapidly in a sequential manner. A typical online learning algorithm processes one instance at a time, making very simple updates with each newly arriving instance iteratively. Wang et al. [11] develop a General Sparse Online Learning algorithm for classification of an online data stream, and modify it to propose the cost-sensitive first-order sparse online learning (CS-FSOL) and cost-sensitive second-order sparse online learning (CS-SSOL) algorithms, to address high sparsity and high-class imbalance problems in big data stream classification. Sparse online learning generates a degree of sparsity into learned weights of online algorithms [12]. CS-SSOL is a second order learning algorithm, which enables it to exploit gradient-based learning more efficiently. A second-order online learner dynamically incorporates knowledge of observed data from an earlier iteration to perform more informative gradient-based learning. In contrast, first-order online learners (such as CS-FSOL) generally utilize a constant learning rate for all coordinates on a curve. Several benchmark datasets from web machine learning repositories are used to evaluate multiple online learners, including CS-FSOL and CS-SSOL [11]. The datasets sizes ranged between approximately 2000 and 2,400,000 instances, with feature set sizes varying between 7510 and 16,079,971, and the class ratio being as high as 99:1. The proposed CS-FSOL and CS-SSOL algorithms are compared against the following online learning algorithms: Cost-Sensitive Online Gradient Descent; Cost-Sensitive Passive-Aggressive; and, Cost-Sensitive Perceptron Algorithm with Uneven Margin. Key conclusions made are: cost-sensitive algorithms are better than non-cost-sensitive algorithms; second-order algorithms are better than first-order algorithms; and CS-SSOL yields the best classification accuracy. The best classification accuracy observed for CS-SSOL was 0.99 (Table 4, Appendix A2). Our general analysis of Wang et al. [11] includes: the proposed solutions need further validation by application across other data-centric domains (for example, software engineering, healthcare, etc.), allowing for a broader insight into their effectiveness; classification accuracy is not a suitable performance metric, especially under high-class imbalance conditions; and, investigating the effectiveness of the proposed algorithm for only highly imbalanced data (i.e., without high sparsity) may provide useful insights.

Marchant and Rubinstein [58] analyzed the issue of class imbalance created by Entity Resolution for big data and proposed the OASIS (Optimal Asymptotic Sequential Importance Sampling) algorithm. Entity Resolution (ER), also known as record linkage, is the process of linking and grouping to locate records that refer to a unique entity within a dataset. OASIS enables the convergence of F-measure, Precision, and Recall to true population values, and is an effective tool for evaluating trained binary classifiers when the class labels are not readily available. It is based on Adaptive/Sequential Importance Sampling (AIS), and at the time of writing this survey article, is available for download as an

$$\frac{TP * TN - FP * FN}{\sqrt{((TP + FP) * (TP + FN) * (TN + FP) * (TN + FN))}}$$

($TP$ = True Positive, $TN$ = True Negative, $FP$ = False Positive, $FN$ = False Negative)

**Fig. 1** Matthew's correlation coefficient (MCC)

open-source Python package.[2] Importance sampling draws from a record-pair distribution, which may depend upon previously-sampled items, and an unbiased estimate can be obtained by using a bias correction estimator. Adaptive Sampling enables the estimates to approach their true values. OASIS unites two key concepts: Stratification and a Bayesian generative model of the label distribution. Stratification is a popular statistical technique for dividing a population into homogenous sub-groups, while the Bayesian generative model partitions matching record pairs into strata.

The empirical case study of Marchant and Rubinstein [58] consisted of five ER datasets, with the pooled/sampled training datasets ranging from approximately 20,000 to 676,000 instances (record-pairs) and the imbalance ratio (IR) ranging from 0.99 to 3380. The OASIS method was compared with three baseline sampling methods: Passive Sampling, Stratified Sampling, and Non-Adaptive Importance Sampling. Passive Sampling is a relatively simple method that samples record pairs randomly with a uniform distribution from the pool with replacement. At each iteration, the F-measure is estimated only on the record pairs/labels sampled so far. Stratified Sampling is used to estimate balanced F-measures, based on pools of record pairs being partitioned into strata. Non-Adaptive Importance Sampling uses a static distribution to sample record pairs. Based on a Linear SVM classifier, the paper concludes that the performance score of OASIS outperformed the baseline methods, with an overall best value for F-measure Absolute Error of $10^{-5}$ (Table 4, Appendix A2). A further comparative investigation with the three baseline methods and OASIS was performed with other classifiers: Multilayer Perceptron with one hidden layer, AdaBoost, Logistic Regression, and SVM with a Gaussian kernel. OASIS also demonstrated superior performance compared to these four classifiers. A likely shortcoming of this work is the relatively very small feature set size of two predictive attributes, which is generally not observed in the data mining community. The ECBDL'14 Big Data Competition [3], for example, featured a dataset containing 631 features. In addition, since OASIS is a relatively new approach, further empirical investigation is needed, including a wider range of comparison with other methods and learners, to ensure performance consistency.

Maurya [13] presents the ImbalancedBayesOpt (IBO) algorithm as a unique approach based on the optimization of Matthew's Correlation Coefficient (MCC), which can be used as a measure of class imbalance in binary datasets. The value for MCC is computed from the *TP*, *TN*, *FP*, and *FN* numbers provided in a classifier's confusion matrix, and its equation is shown in Fig. 1. Based on a Gaussian process, IBO is a Bayesian algorithm that directly maximizes MCC by learning optimal weights for the positive and negative classes. The author also proposes the ImbalancedGridOpt (IGO) algorithm, which

[2] https://git.io/OASIS.

optimizes the MCC metric on an imbalanced dataset using a uniform grid search over the parameter, *w*, i.e., weight of the positive (minority) instances. The case study presented is based on a dataset representing measurements of manufactured parts as they move through the production line, with the binary class representing whether or not a part will fail quality control, where a value of 1 indicates failure (minority class) [13]. The severely imbalanced dataset, represented by a class ratio of about 171:1, consisted of approximately 1.18 million instances characterized by 6747 features. The Gradient Boosting Machine (GBM) learner [59] is used to develop classification models for both IBO and IGO. Two types of sampling methods were used to obtain much smaller training datasets, *Unbiased* and *Biased*, where the former implies sampling was done from both classes using the same sampling rate, *s*, while the latter implies sampling was done only from the majority class (i.e., minority class size was not reduced) using a sampling rate of *s*. In both cases, $s = 1\%$ and $s = 10\%$, were considered during the empirical study. A multi-perspective comparative study of the IBO, IGO, and GBM algorithms indicated these salient results: IBO and IGO perform better than GBM, especially under severely imbalanced datasets; IBO and IGO have comparable performances, with respect to classification accuracy, Recall, Precision, and MCC values; and IBO provides noticeably faster runtimes than IGO. The value of *w* for the best classification performance for IBO was 0.87 (Table 4, Appendix A2). Our general analysis of Maurya [13] is as follows: selection of GBM as the classifier over other more commonly used learners is not advisable, because a few studies show it as a better performer and classifier performances depend on multiple factors including dataset characteristics and empirical settings; the original big data is not completely utilized in the experiments—only a small fraction is used as the training dataset, making the empirical work more comparable to works on class imbalance in traditional data; proposed algorithms, IBO and IGO, are compared to only one other learner (i.e., GBM), limiting the results' generalization in the data mining community; and, only under-sampling is considered (for part of the experiments), instead of also including other data-sampling methods, such as EUS, ROS, and SMOTE.

Veeramachaneni et al. [60] developed the $AI^2$ approach, an active learning scheme for improving detection of outliers in big data in the computer networks domain. By considering both Analyst Intuition and Artificial Intelligence (thus $AI^2$), the approach combines the two common categories of information security analysis methods, i.e., an expert's analytical expertise or using unsupervised machine learning algorithms. The proposed system involves three general steps: (a) the system trains supervised and unsupervised models and uses these them to detect outliers; (b) the predicted outliers are scored by an aggregation of matrix decomposition, artificial neural networks, and joint probability methods; and (c) an analyst inspects the scored instances and then selects the true positives, and incorporates the expert-based inferences into a new model for use the following day. The dataset in this study contained approximately 3.6 billion instances and 37 features, with a class ratio of 680:1. After 3 months of training, $AI^2$ showed a higher attack detection rate of IP packets than an unsupervised learning-only outlier detection approach, and also yielded a lower false positive rate with a higher Recall. The best overall value for AUC was 0.85 (Table 4, Appendix A2). The paper presents little to no information of the unsupervised learning scheme used for comparison against the proposed approach, and to the best of our knowledge, the scheme is a basic clustering-based

method. The improved performance of $AI^2$ is not that surprising since active knowledge (via the analyst's expertise inferences) is being imparted to the model on a daily basis. In addition, $AI^2$ is a form of active learning; hence, it should also be compared against other active learning schemes, and not just a basic unsupervised learning approach. A suggested improvement in the $AI^2$ approach would be to automate or semi-automate the analyst's inferences after a substantial amount of time has passed.

Galpert et al. [14] investigated class imbalance within the problem of ortholog detection of different yeast species. Orthologs are genes in different species that originate from the same gene in the last common ancestor. Several supervised big data techniques were evaluated, such as cost-sensitive Random Forest (RF-BDCS) using MapReduce, Random Over-Sampling with Random Forest (ROS + RF-BD) using MapReduce, and the Apache Spark Support Vector Machines (SVM-BD) combined with MapReduce ROS (ROS + SVM-BD). The datasets in this study ranged from approximately 8,000,000 to 29,900,000 instances, with all of them defined by a feature set size of six. The majority-to-minority class ratios varied between 1630:1 and 10,520:1. The supervised approach was compared with unsupervised algorithms, Reciprocal Best Hits [61], Reciprocal Smallest Distance [62], and OMA [63]. The OMA ("Orthologous MAtrix") project is a method and database for the inference of orthologs among complete genomes. The supervised learning classifiers outperformed the unsupervised learners, and among the supervised classifiers, ROS + SVM-BD showed the best performance with the best overall values for GM and AUC of 0.879 and 0.885, respectively (Table 4, Appendix A2). The authors used an implementation of SVM in the scalable MLLib machine learning library of Spark, but compare it with ROS implemented within the Hadoop framework. The top-performing algorithm, ROS + SVM-BD, was only evaluated against two supervised learning classifiers. In addition, since an over-sampling method was considered in the study, including the SMOTE algorithm also would have provided a broader perspective.

In order to distinguish fraudulent banking transactions from genuine ones within a highly imbalanced environment, Wei et al. [64] presented a system called i-Alertor. The model integrates contrast pattern mining [65], cost-sensitive neural networks, and Decision Forest. Contrast pattern mining, an algorithm tailored for high-class imbalance, was proposed by Wei et al. [64] to improve the efficiency of an emerging pattern algorithm [66] called MDB-LLborder. The efficiency is improved by the indirect calculation of minimum borders, use of hash tables during the validation checking phase, and implementation of a cross-coverage test that prunes redundant patterns. The dataset in this study contained approximately 8,000,000 instances and 130 features, with a majority-to-minority class ratio of 5330:1. An experiment to gauge the performance of i-Alertor against a popular rule-based system used by many Australian banks was carried out. No other information about this rule-based system was provided. The authors found that i-Alertor had a higher detection rate of the minority class (fraud incidents). The average true positive rate was 0.66 (Table 4, Appendix A2), which suggests that a significant percentage of fraudulent transactions can still occur undetected. Unfortunately, Wei et al. [64] only tested the model against one other system, limiting generalization across domains.

Research into the hybridization of under-sampling techniques was carried out by D'Addabbo and Maglietta [67], who introduced Parallel Selective Sampling (PSS). PSS is

an under-sampling technique that is based on Tomek links [68], a modified, condensed nearest-neighbor technique that only evaluates points close to the boundary. This concept is combined with SVM and called the PSS-SVM algorithm. The datasets in this study ranged from approximately 25,000 to 1,000,000 instances, with feature set sizes between 2 and 54. Majority-to-minority class ratios varied between 93:7 and 200:1. PSS-SVM was compared with SVM, RUS-SVM, and RUSBoost [24] classifiers. In terms of prediction accuracy and processing times, PSS-SVM outperformed RUSBoost, SVM, and RUS-SVM. The best overall value for F-measure of PSS-SVM was 0.99 (Table 4, Appendix A2). The experiment performed by D'Addabbo and Maglietta [67] is limited to combining PSS with SVM. It would be useful to investigate how the hybridization of PSS with other classifiers, such as logistic regression or Naive Bayes, affects performance. In addition, a comparison of PSS-SVM against a wide range of hybrid/ensemble techniques should be considered.

For the widely publicized ECDBL'14 Big Data competition [50], the winning algorithm by Triguero et al. [3] deserves special mention. The dataset in this study contained approximately 32 million instances and 631 features, with a majority-to-minority class ratio of 98:2. The ROSEFW-RF algorithm [3] used several MapReduce techniques for potentially addressing the class imbalance problem. ROSEFW-RF stands for Random Over-Sampling and Evolutionary Feature Weighting for Random Forest, and can be described sequentially with six algorithms [3]: (1) Map phase for the ROS algorithm; (2) Reduce phase for the ROS algorithm; (3) Map phase for the RF-BigData algorithm for the building of the model phase; (4) Map phase for the RF-BigData algorithm for classifying phase; (5) Map phase for the DEFW [69] algorithm; (6) Reduce phase for the DEFW algorithm. The best overall value for (true positive*true negative) rate of ROSEFW-RF was 0.53 (Table 4, Appendix A2). The research is limited by MapReduce technologies, which may be sensitive to high-class imbalance.

There is recent research into the use of integrated Extreme Learning Machine (ELM) classifiers to address class imbalance within a MapReduce environment via Apache Hadoop [70]. An ELM classifier is designed to train single-hidden layer feed forward neural networks [70]. The algorithm chooses hidden nodes at random [71]. The datasets in this study ranged between approximately 1500 and 336,000 instances, with feature set sizes between 3 and 16. Majority-to-minority class ratios varied from 92:8 to 2140:1. The new algorithm consists of four stages: (1) Alternately over-sample between positive class instances and negative class instances; (2) construct balanced data subsets based on the generated positive class instances; (3) train component classifiers with ELM algorithm on the constructed balanced data subsets; (4) integrate the ELM classifiers with simple voting approach. Performance-wise, the proposed ELM algorithm by Zhai et al. [70] was shown to be superior to the SMOTE-Boost [30], SMOTE-Vote and SMOTE-Bagging classification techniques. The best overall value for Geometric Mean of the ELM algorithm was 0.969 (Table 4, Appendix A2). The performance of the proposed method has only been compared with SMOTE ensemble methods. A comparison done against a more diverse set of methods would be more informative.

An imbalanced class environment can also be created by rare failure events within large-scale manufacturing operations. Hebert [72] separately compared RF and XGBoost (two tree-based classification methods) with logistic regression to research this issue.

XGBoost [73] is highly scalable, having the ability to process a dataset of billions of examples with very efficient use of computing resources. It is a boosted classification tree that, along with RF, can detect non-linear relationships. The dataset in this study had approximately 1,180,000 instances and 4264 features, with a majority-to-minority class ratio of 170:1. Both tree-based classification methods were found to perform much better than logistic regression. The best overall value for Accuracy Gain of XGBoost was 0.049 (Table 4, Appendix A2). The best overall value for Gini Index Mean Decrease of RF was 0.15 (Table 4, Appendix A2). The Gini Index measures impurity between different classes in a population [72]. A lower Gini Index indicates better class separation. The use of tree-based classifiers has its drawbacks. Implementing one or a couple decision trees may make the understanding of interactions between various parameters a simple process, but a forest of trees can be quite complex to comprehend. In addition, there are other classifiers, such as neural networks and k-nearest neighbors (k-NN), which can be also be used in an experiment to test comparative performances of linear systems vs. non-linear systems.

An evaluation of the performance of several methods used to address class imbalance was performed by Rio et al. [46], where all methods were implemented within the MapReduce framework (via Apache Hadoop and Apache Mahout), with RF as the base classifier. These methods included ROS, RUS, SMOTE, and a cost-sensitive learning version of RF. The datasets in this study ranged from approximately 435,000 to 5,700,000 instances, with feature set sizes between 2 and 41. Majority-to-minority class ratios varied between 80:20 and 77,670:1. The results were inconclusive, as there was no best model among these four diverse algorithms. The authors state that the best performing algorithm depends on the number of mappers with MapReduce that are chosen to run the experiment. This is contrary to what was found in [32, 51], where ROS was the top performing method within the MapReduce environment. This inconsistency of results opens up an avenue for future work. For Geometric Mean, the best overall values of ROS, RUS, SMOTE, and RF were 0.986, 0.984, 0.914, and 0.965, respectively (Table 4, Appendix A2). The main limitation of this study relates to the relevance of the obtained results. Computational frameworks such as Apache Spark appear to be more tolerant to a highly imbalanced class environment [16], and in our opinion, experimental work done on highly imbalanced datasets within these flexible big data frameworks could produce more meaningful results.

Baughman et al. [74] examined a modification to DeepQA, the technology that powered IBM Watson on the *Jeopardy!* game show. DeepQA is a question-and-answer, natural language processing system that can help professionals make critical and timely decisions [75]. The authors combined a Data-Level approach (manual question and answer vetting, over-sampling) with an Algorithm-Level approach (logistic regression with a regularization term) to address the issue of high-class imbalance. Adding a regularization term to logistic regression during the training phase has been shown to reduce the effect of class imbalance [74]. The dataset in this study contained approximately 720,000 instances and 400 features, with a majority-to-minority class ratio of 6930:1. The results show that regularized logistic regression with over-sampling outperformed unregularized logistic regression with over-sampling in terms of accuracy, which increased by 0.0158–0.28 (Table 4, Appendix A2). This best value of accuracy is low. The results

also show that the Data-Level techniques of vetting and over-sampling increased the recall rate. The skill level of the person doing the vetting is potentially a limiting factor for the method proposed by Baughman et al. [74]. For example, vetting done by a professional would most likely yield higher accuracy values than vetting done by a college student. Furthermore, apart from the logistic regression classifier, other classifiers should also be evaluated.

### Performance scores of different approaches surveyed

In the Appendices (A1 and A2) sections, Tables 3 and 4 collectively show performance scores that represent the best experimental values for the Data-Level and Algorithm-Level approaches in the papers covered by the survey. It should be noted that comparisons between performance measures from different datasets may not be valid. These datasets may vary in the number instances, number of features, and the type of computational framework selected, if any is used. Additionally, variations of an original experiment may be carried out on the same dataset. However, providing these performance scores may be valuable for future comparative research.

### Summarized highlights of surveyed works

| Data-Level approaches |
|---|
| Fernandez et al. [32] |
| The use of ROS and RUS produce better classification results than SMOTE. When ROS was compared with RUS, the better sampler was ROS. It was also shown that the performance of SMOTE is negatively affected by MapReduce partitioning |
| Rio et al. [51] |
| ROS produced better classification results than RUS within a MapReduce framework |
| Triguero et al. [16] |
| A shorter runtime was obtained when Apache Spark was used instead of Apache Hadoop (MapReduce). The authors also compared simple RUS with evolutionary under-sampling. RUS had a faster runtime, as expected, but the latter is a better classifier |
| Park et al. [54] |
| SMOTE was implemented on Apache Hadoop (MapReduce) to determine the optimum balance between over-sampling and classification accuracy for big data. Results indicate that the greatest increase in classification accuracy occurred when the minority class instances were increased to about 25% (11,200% over-sampling) of the original dataset |
| Park and Ha [56] |
| SMOTE was also implemented on Apache Hadoop (MapReduce) in this study. After over-sampling, the optimum balance between over-sampling and classification accuracy was 30% (28,700% over-sampling) of the original dataset |
| Chai et al. [57] |
| Research on the use of under-sampling to balance a dataset (majority class instances reduced to 50% of the original dataset) shows that classification performance is unaffected |
| **Algorithm-Level approaches** |
| Lopez et al. [9] |
| The experiments conducted for this work are centered on a linguistic cost-sensitive fuzzy rule-based classifier using Apache Hadoop (MapReduce). The algorithm is called Chi-FRBCS-BigDataCS |
| Wang et al. [11] |
| CS-SSOL is a second-order, sparse classifier designed for use on an online data stream |
| Marchant and Rubinstein [58] |
| OASIS is a novel algorithm based on adaptive importance sampling. The algorithm is essentially a tool for evaluating trained binary classifiers when class labels are not readily available |

| **Algorithm-Level approaches** |
| --- |
| Maurya [13] |
| ImbalancedBayesOpt is a Bayesian optimization algorithm that maximizes MCC by learning optimal weights for the positive and negative classes |
| Veeramachaneni et al. [60] |
| $AI^2$ combines artificial intelligence (matrix decomposition, neural networks, joint probability methods) with analyst intuition |
| Galpert et al. [14] |
| This method combines Apache Hadoop (MapReduce) ROS with Apache Spark SVM (ROS + SVM-BD) |
| Wei et al. [64] |
| The i-Alertor model was introduced in this research work. It integrates contrast pattern mining, cost-sensitive neural networks, and Decision Forest |
| D'Addabbo and Maglietta [67] |
| This method combines Parallel Selective Sampling with SVM (PSS-SVM) |
| Triguero et al. [3] |
| ROSEFW-RF, a combination of ROS and evolutionary feature weighting using Random Forest with Apache Hadoop (MapReduce), was the winning algorithm in the ECDBL'14 Big Data competition |
| Zhai et al. [70] |
| This research is focused on the use of ELM classifiers with Apache Hadoop (MapReduce) |
| Hebert [72] |
| Experimental results show that certain tree-based classifiers, such as RF and XGBoost, are better learners than logistic regression |
| Rio et al. [46] |
| The results of this study indicate that the classification performance of sampling methods, such as ROS, RUS, and SMOTE, is strongly related to the number of mappers used within a MapReduce environment (Apache Hadoop). The type of Data-Level sampling technique does not appear to matter |
| Baughman et al. [74] |
| The research work discusses the gamification of the DeepQA system for real-world use. The proposed solution is a combination of algorithm level approaches (logistic regression with a regularization term) and data level approaches (question and answer modifications, over-sampling) |

## Discussion summary of surveyed works

The papers discussed in the previous section cover a wide range of solutions for class imbalance at the Data-Level and Algorithm-Level. However, we recognize that there are significant gaps in the current research on high-class imbalance in a big data environment. The literature is lacking in research methods on topics such as within-class imbalance, small disjuncts, feature selection, stacked ensembles, advanced over-sampling techniques, and one-class learners. We expound on these topics in the following paragraphs.

Where minority class instances are very limited, the lack of representative data can make learning difficult regardless of the imbalance between classes [2]. Moreover, the minority target concept may also contain a sub-concept with limited instances, which can result in classification difficulty for the learner. This condition is known as within-class imbalance and is defined by the distribution of representative data for sub-concepts within a class. For big data, there seems to be a knowledge gap concerning the effects of within-class imbalance on sampling, particularly for ROS [8]. In one of the papers covered by our survey, ROS was shown to outperform SMOTE and RUS within a MapReduce framework [32]. If the class imbalance problem of a dataset is caused by within-class concepts, ROS may over-duplicate samples on some regions more than on

others [8, 76]. A more favorable resampling process should, first, detect the sub-concepts constituting the class, then oversample each concept respectively to balance the overall distribution [76].

The problem of small disjuncts is related to the problem of within-class imbalances [2]. The classifier frequently creates large disjuncts, which essentially are the rules that govern a significant portion of instances related to the target concept under consideration. However, there are also underrepresented sub-concepts with small portions of instances that are governed by rules or small disjuncts. The validity of clusters corresponding to small disjuncts can be problematic, since these disjuncts may be influenced by factors such as noise. Furthermore, the amount of noise associated with big data may generally be higher than that of traditional data. The problem of small disjuncts may also worsen during data partitioning within the MapReduce (Apache Hadoop) big data analysis framework [32], and in our opinion, the small disjuncts problem in big data analysis presents itself as a relatively open research question.

For classification tasks in a highly imbalanced class environment, feature selection helps to suggest the most influential features that can provide additional information for class discrimination, and the added benefits of improving classification performance and reducing computational cost in some cases. The increasing use of the Internet has greatly contributed to the data sharing and the use of applications for big data with high dimensionality. For example, Wang et al. [11] proposed an online learning classifier in a study with datasets containing as many as 16,000,000 features. It should be noted that we were unable to locate research papers published within the last 8 years that exclusively used feature selection to address class imbalance for big data. Admittedly, we discussed ROSEFW-RF [3], an algorithm that combines ROS and Evolutionary Feature Weighting, but there are certain feature selection methods [21–23] that specifically target class imbalance.

Although variations of the bagging and boosting ensemble techniques were included in our survey, we did not encounter any research that involved stacked ensembles being used to address class imbalance for big data. Stacking is a hybrid technique that consists of two phases [25]. The first phase involves different models being learned, with the output from each model combined to create a new dataset. In the second-phase, the new dataset is used with a meta-learner to provide the final output. Stacked ensembles are commonly used in data science competitions.[3]

With regard to intelligent over-sampling techniques at the Data-Level, our survey only found published research on SMOTE, which is commonly used as a benchmark. However, there are several other advanced over-sampling techniques that SMOTE could be evaluated against within a big data context. The popular ones include Mega-Trend Diffusion Function (MTDF), Adaptive Synthetic Sampling Approach (ADASYN), Majority Weighted Minority Oversampling Technique (MWMOTE), Immune Centroids Oversampling Technique (ICOTE) and Couples Top-N Reverse k-Nearest Neighbor (TRkNN) [77].

[3] https://blogs.sas.com/content/subconsciousmusings/2017/05/18/stacked-ensemble-models-win-data-science-competitions/.

As a final point, we also did not encounter any one-class learners during our exhaustive search for Algorithm-Level techniques. One-class learning algorithms, also known as recognition-based methods, work by modelling the classifier on the representation of the minority class, instead of both classes [8]. One-class learning is particularly useful when used on very highly imbalanced data sets composed of a high-dimensional noisy feature space. For big data, this technique is certainly worth researching. However, it should be noted that some machine learning algorithms, such as decision tree and k-NN, do not function with instances from only one class.

## Conclusion

Big data with a class imbalance can introduce a bias in favor of the majority class. In situations where the occurrence of false negatives is costlier than false positives, this bias can have undesirable consequences. Our survey paper focuses on big data and high-class imbalance, with particular emphasis on big-data appropriate techniques for research work that uses at least one dataset of over 100,000 instances in a published study.

Data-Level and Algorithm-Level solutions can be implemented to address high-class imbalance. The Data-Level approach covers sampling and feature selection techniques. Sampling techniques consist of over-sampling and under-sampling solutions. The Algorithm-Level approach covers cost-sensitive and hybrid/ensemble techniques. For Data-Level sampling techniques, ROS had a better classification accuracy than RUS and SMOTE in most studies (using a MapReduce framework). Additionally, the determination of the optimum balance between over-sampling percentage and classification accuracy in a MapReduce environment was shown to be an empirical and not a formulaic process. Furthermore, Rio et al. [46] observed that the performance superiority of SMOTE, ROS, RUS, (Data-Level methods) or cost-sensitive RF (Algorithm-Level method) may essentially depend on the number of MapReduce mappers used. With both Data-Level and Algorithm-Level approaches, a recurring limitation in the studies appears to be the use of MapReduce technology as a computational framework. A more flexible framework such as Apache Spark could provide more valuable results. Another general limitation is the narrow scope of evaluated techniques for each study. We recommend that evaluations should be performed on a diverse range of classifiers presented in the surveyed works, as listed in Table 2.

**Table 2 Surveyed works categorized based on method type**

| Data-Level methods | Algorithm-Level methods | |
|---|---|---|
| **Data-sampling methods** | **Cost-sensitive methods** | **Hybrid/ensemble methods** |
| Fernandez et al. [32] | Lopez et al. [9] | Marchant and Rubinstein [58] |
| Rio et al. [51] | Wang et al. [11] | Maurya [13] |
| Triguero et al. [16] | | Veeramachaneni et al. [60] |
| Park et al. [54] | | Galpert et al. [14] |
| Park and Ha [56] | | Wei et al. [64] |
| Chai et al. [57] | | D'Addabbo and Maglietta [67] |
| | | Triguero et al. [3] |
| | | Zhai et al. [70] |
| | | Hebert [72] |
| | | Rio et al. [46] |
| | | Baughman et al. [74] |

Detailed tables are provided in the Appendices A1, A2 section

Future work is needed to settle apparently conflicting conclusions about big data and high-class imbalance. For example, Fernandez et al. [32] and Rio et al. [51] vouch for the outstanding performance of ROS. However, Rio et al. [46] state that the performance of sampling techniques such as ROS, RUS, and SMOTE is closely related to the number of mappers used within a MapReduce environment, and that there was essentially no performance difference between the three approaches. For experiments performed solely within a MapReduce environment, it would be interesting to see the results if those same experiments were conducted within a more flexible big data computational framework, such as Apache Spark.

Clearly, many more evaluations need to be done before a particular method can be judged a top performer. For example, Lopez et al. [9] introduced a cost-sensitive classifier, Chi-FRBCS-BigDataCS, for fuzzy rule-based systems, but there is no published work on how this technique compares with other cost-sensitive classifiers that are not fuzzy rule-based. Additionally, for some of the newer algorithms such as OASIS [58], evaluations on a diverse range of big data and classifiers should be carried out to validate performance consistency. Lastly, future work should specifically address the research gaps discussed previously, including within-class imbalance, small disjuncts, feature selection, stacked ensembles, advanced over-sampling techniques, and one-class learners.

**Authors' contributions**
JLL, RAB, and NS performed the primary literature review and analysis for this work, and also drafted the manuscript. TMK worked with JLL to develop the article's framework and focus. TMK introduced this topic to JLL, and helped to complete and finalize this work. All authors read and approved the final manuscript.

**Author details**
[1] Florida Atlantic University, Boca Raton, USA. [2] Ohio Northern University, Ada, USA.

**Acknowledgements**
We would like to thank the reviewers in the Data Mining and Machine Learning Laboratory at Florida Atlantic University. Additionally, we acknowledge partial support by the NSF (CNS-1427536). Opinions, findings, conclusions, or recommendations in this paper are solely of the authors' and do not reflect the views of the NSF.

**Competing interests**
The authors declare that they have no competing interests.

**Availability of data and materials**
Not applicable.

**Consent for publication**
Not applicable.

**Ethics approval and consent to participate**
Not applicable.

**Funding**
Not applicable.

## Appendix A1

### Data-Level approach

The performance evaluators in Table 3 show score results that represent the best or optimal experimental values. Comparisons between results for separate works of research or separate experiments within the same paper may not be valid. However, providing these performance scores may be valuable for future comparative research.

**Table 3 Performances' summary of Data-Level methods**

| Technique | GM | TPR*TNR | AUC | Accuracy | F-measure | Big Data framework |
|---|---|---|---|---|---|---|
| Data-Sampling methods | | | | | | |
| Fernandez et al. [32] | | | | | | Apache Hadoop and Apache Spark |
| ROS | 0.71 | – | – | – | – | |
| RUS | 0.70 | – | – | – | – | |
| SMOTE | 0.63 | – | – | – | – | |
| Rio et al. [51] | | | | | | Apache Hadoop |
| ROS | – | 0.49 | – | – | – | |
| RUS | – | 0.48 | – | – | – | |
| Triguero et al. [16] | | | | | | Apache Hadoop and Apache Spark |
| EUS | 0.67 | – | 0.67 | – | – | |
| RUS | 0.66 | – | 0.66 | – | – | |
| Park et al. [54] | | | | | | Apache Hadoop |
| SMOTE | – | – | – | 0.76 | – | |
| Park and Ha [56] | | | | | | Apache Hadoop |
| SMOTE | – | – | – | 0.81 | – | |
| Chai et al. [57] | | | | | | – |
| RUS | – | – | – | – | 0.99 | |

## Appendix A2

### Algorithm-Level approach

The performance evaluators in Table 4 show score results that represent the best or optimal experimental values. Comparisons between results for separate works of research or separate experiments within the same paper may not be valid. However, providing these performance scores may be valuable for future comparative research.

**Table 4 Performances' summary of algorithm-level methods**

| Technique | GM[a] | TP*TN[b] | AUC[c] | A[d] | AG[e] | F[f] | FAE[g] | *W*[h] | TP[i] | GD[j] | BDF[k] |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Cost-sensitive | | | | | | | | | | | |
| Lopez et al. [9] | | | | | | | | | | | Apache Hadoop |
| Chi-FRBCS-Big DataCS | – | – | 0.99 | – | – | – | – | – | – | – | |
| Wang et al. [11] | | | | | | | | | | | – |
| CS-SSOL | – | – | – | 0.99 | – | – | – | – | – | – | |
| Hybrid/ensemble | | | | | | | | | | | |
| Marchant and Rubinstein [58] | | | | | | | | | | | – |
| OASIS | – | – | – | – | – | – | $10^{-5}$ | – | – | – | |
| Maurya [13] | | | | | | | | | | | – |
| IBO | – | – | – | – | – | – | – | 0.87 | – | – | |
| Veeramachaneni et al. [60] | | | | | | | | | | | – |
| $AI^2$ | – | – | 0.85 | – | – | – | – | – | – | – | |
| Galpert et al. [14] | | | | | | | | | | | Apache Hadoop and Apache Spark |
| ROS + SVM-BD | 0.88 | – | 0.89 | – | – | – | – | – | – | – | |
| Wei et al. [64] | | | | | | | | | | | – |
| i-Alertor | – | – | – | – | – | – | – | – | 0.66 | – | |
| D'Addabbo and Maglietta [67] | | | | | | | | | | | – |
| PSS-SVM | – | – | – | – | – | 0.99 | – | – | – | – | |
| Triguero et al. [3] | | | | | | | | | | | Apache Hadoop |
| ROSEFW-RF | – | 0.53 | – | – | – | – | – | – | – | – | |
| Zhai et al. [70] | | | | | | | | | | | Apache Hadoop |
| ELM ensemble | 0.97 | – | – | – | – | – | – | – | – | – | |
| Hebert [72] | | | | | | | | | | | – |
| RF | – | – | – | – | – | – | – | – | – | 0.15 | |
| XGBoost | – | – | – | – | 0.05 | – | – | – | – | – | |
| Rio et al. [46] | | | | | | | | | | | Apache Hadoop |
| ROS | 0.99 | – | – | – | – | – | – | – | – | – | |
| RUS | 0.98 | – | – | – | – | – | – | – | – | – | |
| SMOTE | 0.91 | – | – | – | – | – | – | – | – | – | |
| RF | 0.97 | – | – | – | – | – | – | – | – | – | |
| Baughman et al. [74] | | | | | | | | | | | – |
| DeepQA | – | – | – | 0.28 | – | – | – | – | – | – | |

[a] Geometric mean

[b] True positive rate * true negative rate

[c] Area under the ROC curve

[d] Accuracy

[e] Accuracy gain

[f] F-measure

[g] F-measure absolute error

[h] Positive datapoints weight

[i] True positive rate

[j] Gini index mean decrease

[k] Big Data framework

## Publisher's Note



Received: 29 August 2018 Accepted: 19 October 2018
Published online: 01 November 2018

### References

1. Bauder RA, Khoshgoftaar TM. The effects of varying class distribution on learner behavior for medicare fraud detection with imbalanced Big Data. Health Inf Sci Syst. 2018;6:9 **(14 pages)**.
2. He H, Garcia E. Learning from imbalanced data. IEEE Trans Knowl Data Eng. 2009;21(9):1263–84.
3. Triguero I, Rio S, Lopez V, Bacardit J, Benítez J, Herrera F. ROSEFW-RF: the winner algorithm for the ECBDL'14 big data competition: an extremely imbalanced big data bioinformatics problem. Knowl Based Syst. 2015;87:69–79.
4. Seliya N, Khoshgoftaar TM, Van Hulse J. A study on the relationships of classifier performance metrics. In: 21st international conference on tools with artificial intelligence (ICTAI 2009). IEEE. 2009. pp. 59–66.
5. Katal A, Wazid M, Goudar R. Big data: issues, challenges, tools, and good practices. In: Sixth international conference on contemporary computing. 2013.
6. Herland M, Khoshgoftaar TM, Bauder RA. Big Data fraud detection using multiple medicare data sources. J Big Data. 2018;5:29 **(21 pages)**.
7. Bauder RA, Khoshgoftaar TM. Medicare fraud detection using random forest with class imbalanced Big Data. In: 2018 IEEE international conference on information reuse and integration (IRI), IEEE. 2018. pp. 80–7.
8. Ali A, Shamsuddin SM, Ralescu AL. Classification with class imbalance problem: a review. Int J Adv Soft Comput Appl. 2015;7(3):176–204.
9. Lopez V, Rio S, Benitez J, Herrera F. Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced Big Data. Fuzzy Sets Syst. 2015;258:5–38.
10. Chi Z, Yan H, Pham T. Fuzzy algorithms with applications to image processing and pattern recognition. Singapore: World Scientific; 1996.
11. Wang D, Wu P, Zhao P, Hoi S. A framework of sparse online learning and its applications. Comput Sci. 2015.
12. Langford J, Li L, Zhang T. Sparse online learning via truncated gradient. J Mach Learn Res. 2009;10:777–801.
13. Maurya A. Bayesian optimization for predicting rare internal failures in manufacturing processes. In: IEEE international conference on Big Data. 2016.
14. Galpert D, del Río S, Herrera F, Ancede-Gallardo E, Antunes A, Agüero-Chapin G. An effective Big Data supervised imbalanced classification approach for ortholog detection in related yeast species. BioMed Res Int. 2015;2015:748681. https://doi.org/10.1155/2015/748681.
15. Tsai C, Lin W, Ke S. Big Data mining with parallel computing: a comparison of distributed and MapReduce methodologies. J Syst Softw. 2016;122:83–92.
16. Triguero I, Galar M, Merino D, Maillo J, Bustince H, Herrera F. Evolutionary undersampling for extremely imbalanced Big Data classification under Apache Spark. In: IEEE congress on evolutionary computation (CEC). 2016.
17. Khoshgoftaar TM, Seiffert C, Van Hulse J, Napolitano A, Folleco A. Learning with limited minority class data. In: Sixth international conference on machine learning and applications (ICMLA 2007), IEEE. 2007. pp. 348–53.
18. Van Hulse J, Khoshgoftaar TM, Napolitano A. Experimental perspectives on learning from imbalanced data. In: Proceedings of the 24th international conference on machine learning, ACM. 2007. pp. 935–42.
19. Malhotra R. A systematic review of machine learning techniques for software fault prediction. Appl Soft Comput. 2015;27:504–18.
20. Wang H, Khoshgoftaar TM, Napolitano A. An empirical investigation on Wrapper-Based feature selection for predicting software quality. Int J Softw Eng Knowl Eng. 2015;25(1):93–114.
21. Yin L, Ge Y, Xiao K, Wang X, Quan X. Feature selection for high-dimensional imbalanced data. Neurocomputing. 2013;105:3–11.
22. Mladenic D, Grobelnik M. Feature selection for unbalanced class distribution and Naïve Bayes. In: International conference on machine learning. 1999.
23. Zheng Z, Wu X, Srihari R. Feature selection for text categorization on imbalanced data. Explor Newsletter. 2014;6(1):80–9.
24. Seiffert C, Khoshgoftaar TM. RUSBoost: a hybrid approach to alleviating class imbalance. IEEE Trans Syst Man Cybern Part A. 2010;40(1):185–97.
25. Graczyk M, Lasota T, Trawinski B, Trawinski K. Comparison of bagging, boosting and stacking ensembles applied to real estate appraisal. In: Asian conference on intelligent information and database systems. 2010. pp. 340–50.
26. McCallum A, Nigam K. A comparison of event models for Naive Bayes text classification. In: AAAI-98 workshop on learning for text categorization. 1998.
27. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32.
28. Ho T. Random decision forests. In: Proceedings of the third international conference on document analysis and recognition. 1995.
29. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: synthetic minority over-sampling technique. J Artif Intell Res. 2002;16:321–57.
30. Chawla N, Lazarevic A, Hall L, Bowyer K. SMOTEBoost: improving prediction of the minority class in boosting. In: 7th European conference on principles and practice of knowledge discovery in databases. 2003.
31. Rodriguez D, Herraiz I, Harrison R, Dolado J, Riquelme J. Preliminary comparison of techniques for dealing with imbalance in software defect prediction. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering. Article no. 43. 2014.

32. Fernandez A, Rio S, Chawla N, Herrera F. An insight into imbalanced Big Data classification: outcomes and challenges. Complex Intell Syst. 2017;3:105–20.
33. Cao P, Zhao D, Zaiane O. An optimized cost-sensitive SVM for imbalanced data learning. In: Pacific-Asia conference on knowledge discovery and data mining. 2013. pp. 280–92.
34. Cao P, Zhao D, Zaiane O. A PSO-based cost-sensitive neural network for imbalanced data classification. In: Pacific-Asia conference on knowledge discovery and data mining. 2013. pp. 452–63.
35. Li N, Tsang IW, Zhou Z-H. Efficient optimization of performance measures by classifier adaptation. IEEE Trans Pattern Anal Mach Intell. 2013;35(6):1370–82.
36. Zhang G, Patuwo B, Hu M. Forecasting with artificial neural networks: the state of the art. Int J Forecast. 1998;14:35–62.
37. López V, Fernandez A, Moreno-Torres J, Herrera F. Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics. Expert Syst Appl. 2012;39(7):6585–608.
38. Kaminski B, Jakubczyk M, Szufel P. A framework for sensitivity analysis of decision trees. CEJOR. 2017;26(1):135–59.
39. Akbani R, Kwek S, Japkowicz N. Applying support vector machines to imbalanced datasets. In: European conference on machine learning. 2004. pp. 39–50.
40. Tang Y, Chawla N. SVMs modeling for highly imbalanced classification. IEEE Trans Syst Man Cybern. 2009;39(1):281–8.
41. Ahumada H, Grinblat G, Uzal L, Granitto P, Ceccatto A. REPMAC: a new hybrid approach to highly imbalanced classification problems. In: Eighth international conference on hybrid intelligent systems. 2008.
42. Bekkar M, Alitouche T. Imbalanced data learning approaches review. Int J Data Mining Knowl Manag Process. 2013;3(4):15–33.
43. Khoshgoftaar TM, Golawala M, Van Hulse J. An empirical study of learning from imbalanced data using random forest. In: 19th IEEE international conference on tools with artificial intelligence (ICTAI 2007), IEEE, vol. 2, pp. 310–17. 2007.
44. Chen C, Liaw A, Breiman L. Using random forest to learn imbalanced data. Tech Report 666, University of California, Berkeley. 2004.
45. Galar M, Fernandez A, Barrenechea E, Bustince H, Herrera F. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Trans Syst Man Cybern C Appl Rev. 2012;42(4):463–84.
46. Río S, Lopez V, Benítez J, Herrera F. On the use of MapReduce for imbalanced Big Data using random forest. Inf Sci. 2014;285:112–37.
47. The Apache Software Foundation. Apache Mahout. 2017. http://mahout.apache.org/users/classification/. Accessed Apr 2018.
48. Landset S, Khoshgoftaar TM, Richter A, Hasanin T. A survey of open source tools for machine learning with big data in the Hadoop ecosystem. J Big Data. 2015;2(24):1–36.
49. Frank E, Hall MA, Witten IH. The WEKA Workbench: data mining: practical machine learning tools and techniques. 4th ed. Burlington: Morgan Kaufmann; 2016.
50. Evolutionary computation for Big Data and Big learning workshop data mining competition 2014: self-deployment track. 2014. http://cruncher.ico2s.org/bdcomp/. Accessed 4 Sept 2018.
51. Rio S, Benitez J, Herrera F. Analysis of data preprocessing: increasing the oversampling ratio for extremely imbalanced Big Data classification. In: IEEE Trustcom/BigDataSE/ISPA. 2015.
52. Garcıa S, Herrera F. Evolutionary under-sampling for classification with imbalanced data sets: proposals and taxonomy. Evol Comput. 2009;17(3):275–306.
53. Triguero I, Galar M, Vluymans S, Cornelis C, Bustince H, Herrera F, Saeys Y. Evolutionary under sampling for imbalanced big data classification. In: IEEE congress on evolutionary computation (CEC), May 2015, pp. 715–22.
54. Park S, Kim S, Ha Y. Highway traffic accident prediction using VDS big data analysis. J Supercomput. 2016;72(7):2815–31.
55. Ng A, Jordan M. On discriminative vs. generative classifiers: a comparison of logistic regression and Naïve Bayes. Adv Neural Inf Process Syst. 2002;14:605–10.
56. Park S, Ha Y. Large imbalance data classification based on MapReduce for traffic accident prediction. In: Eighth international conference on innovative mobile and internet services in ubiquitous computing. 2014.
57. Chai K, Anthony S, Coiera E, Magrabi F. Using statistical text classification to identify health information technology incidents. J Am Med Inform Assoc. 2013;20(5):980–5.
58. Marchant NG, Rubinstein BIP. In search of an entity resolution OASIS: optimal asymptotic sequential importance sampling. Proc VLDB Endow. 2017;10(11):1322–33.
59. Friedman J. Greedy function approximation: a gradient boosting machine. Ann Stat. 2001;29(5):1189–232.
60. Veeramachaneni K, Arnaldo I, Korrapati V, Bassias C, Li K. AI$^2$: training a Big Data machine to defend. In: IEEE 2nd international conference on Big Data security on cloud. 2016.
61. Hirsh A, Fraser H. Protein dispensability and rate of evolution. Nature. 2001;411(6841):1040–9.
62. Wall D, Fraser H, Hirsh A. Detecting putative orthologs. Bioinformatics. 2003;19(13):1710–1.
63. Roth A, Gonnett G, Dessimoz C. Algorithm of OMA for large-scale orthology inference. BMC Bioinform. 2008;9:518.
64. Wei W, Li J, Cao L, Ou Y, Chen J. Effective detection of sophisticated online banking fraud on extremely imbalanced data. World Wide Web. 2013;16(4):449–75.
65. Wang L, Zhao H, Dong G, Li J. On the complexity of finding emerging patterns. Theor Comput Sci. 2005;335(1):15–27.
66. Jong D, Li J. Efficient mining of emerging patterns: discovering trends and differences. In: Fifth ACMSIGKDD international conference on knowledge discovery and data mining. 1999. pp. 43–52.
67. D'Addabbo A, Maglietta R. Parallel selective sampling method for imbalanced and large data classification. Pattern Recogn Lett. 2015;62:61–7.
68. Tomek I. Two modifications of CNN. IEEE Trans Syst Man Cybern. 1976;6(11):769–72.
69. Triguero I, Derrac J, García S, Herrera F. Integrating a differential evolution feature weighting scheme into prototype generation. Neurocomputing. 2012;97:332–43.

70. Zhai J, Zhang S, Wang C. The classification of imbalanced large data sets based on MapReduce and ensemble of ELM classifiers. Int J Mach Learn Cybern. 2017;8(3):1009–17.
71. Huang G, Zhu Q, Siew C. Extreme learning machine: theory and applications. Neurocomputing. 2006;70(1–3):489–501.
72. Hebert J. Predicting rare failure events using classification trees on large scale manufacturing data with complex interactions. In: IEEE international conference on Big data. 2016.
73. Chen T, Guestrin C. XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016. pp. 785–94.
74. Baughman A, Chuang W, Dixon K, Benz Z, Basilico J. Deepqa jeopardy! gamification: a machine-learning perspective. IEEE Trans Comput Intell AI Games. 2014;6(1):55–66.
75. Ferrucci D, Brown E, Chu-Carroll J, Fan J, Gondek D, Kalyanpur A, Lally A, Murdock J, Nyberg E, Prager J, Schlaefer N, Welty C. Building Watson: an overview of the DeepQA project. AI Mag. 2010;31(3):59–79.
76. Sun Y, Wong A, Kamel M. Classification of imbalanced data: a review. Int J Pattern Recogn Artif Intell. 2009;23(4):687–719.
77. Amin A, Anwar S, Adnan A, Nawaz M, Howard N, Qadir J, Hawalah A, Hussain A. Comparing oversampling techniques to handle the class imbalance problem: a customer churn prediction case study. IEEE Access. 2016;4:7940–57.