Journal of Big Data

RESEARCH Open Access

Severely imbalanced Big Data challenges: investigating data sampling approaches

Tawfiq Hasanin, Taghi M. Khoshgoftaar, Joffrey L. Leevy* and Richard A. Bauder

*Correspondence:
jleevy2017@fau.edu
Florida Atlantic University,
777 Glades Road, Boca Raton,
FL 33431, USA

Abstract

Severe class imbalance between majority and minority classes in Big Data can bias the predictive performance of *Machine Learning* algorithms toward the majority (negative) class. Where the minority (positive) class holds greater value than the majority (negative) class and the occurrence of false negatives incurs a greater penalty than false positives, the bias may lead to adverse consequences. Our paper incorporates two case studies, each utilizing three learners, six sampling approaches, two performance metrics, and five sampled distribution ratios, to uniquely investigate the effect of severe class imbalance on Big Data analytics. The learners (*Gradient-Boosted Trees, Logistic Regression, Random Forest*) were implemented within the Apache Spark framework. The first case study is based on a Medicare fraud detection dataset. The second case study, unlike the first, includes training data from one source (SlowlorisBig Dataset) and test data from a separate source (POST dataset). Results from the Medicare case study are not conclusive regarding the best sampling approach using *Area Under the Receiver Operating Characteristic Curve* and *Geometric Mean* performance metrics. However, it should be noted that the *Random Undersampling* approach performs adequately in the first case study. For the SlowlorisBig case study, Random Undersampling convincingly outperforms the other five sampling approaches (*Random Oversampling, Synthetic Minority Over-sampling TEchnique, SMOTE-borderline1 , SMOTE-borderline2 , ADAptive SYNthetic*) when measuring performance with *Area Under the Receiver Operating Characteristic Curve* and *Geometric Mean* metrics. Based on its classification performance in both case studies, *Random Undersampling* is the best choice as it results in models with a significantly smaller number of samples, thus reducing computational burden and training time.

Keywords: Big Data, Class imbalance, Machine Learning, Medicare fraud, Oversampling, SlowlorisBig, Undersampling

Introduction

The exponential increase of raw data in recent years has been associated with technological advances in the fields of *Data Mining* (DM) and *Machine Learning* (ML) [1, 2]. These advances have significantly improved the efficiency and effectiveness of Big Data applications in a diverse range of areas, such as knowledge discovery and information processing. Big Data is identified by various data-related properties, and for this reason, an exact definition of Big Data remains elusive. One definition, presented by Senthilkumar et al. [3], relates Big Data to six V's: Volume, Variety, Velocity, Veracity, Variability, and Value. Volume is associated with the reams of data produced by an organization.

Variety is concerned with the different formats of data, e.g., organized, partially organized, or unorganized. Velocity covers how rapidly data is manufactured, provided, and handled. Veracity reflects the correctness of the data. Variability pertains to data fluctuations. Value, also known as Big Data analytics, is the method of data extraction for effective decision-making.

Class imbalance is the term used for a dataset containing a majority and minority class. The spectrum of class imbalance ranges from "slightly imbalanced" to "rarity." Dataset rarity is associated with insignificant numbers of positive instances [4], e.g., the occurrence of 25 fraudulent transactions among 1,000,000 normal transactions within a financial security dataset of a reputable bank. Since many multi-class problems can be simplified by binary classification, data scientists frequently take the binary approach for analytics [5]. The minority (positive) class, which accounts for a smaller percentage of the dataset, is often the class of interest in real-world problems [5]. The majority (negative) class constitutes the larger percentage.

Machine learning algorithms generally outperform traditional statistical techniques at classification [6–8], but these algorithms cannot effectively distinguish between majority and minority classes if the dataset suffers from severe class imbalance or rarity. Severely imbalanced data, also known as high-class imbalance, is often defined by majority-to-minority class ratios between 100:1 and 10,000:1 [5]. The failure to sufficiently distinguish between majority and minority classes is akin to searching for a proverbial polar bear in a snowstorm and could cause the classifier to label almost all instances as the majority (negative) class, thereby producing an accuracy performance metric value that is deceptively high. When the occurrence of a false negative incurs a higher penalty than a false positive, a classifier's prediction bias in favor of the majority class may lead to adverse consequences [9]. For example, if defective flight-control software for a jetliner is classified as defect-free (false negative), the end result of greenlighting the production of this software could be catastrophic. Conversely, if the software is defect-free but was flagged as defective, the outcome would most certainly not pose an imminent threat to human life.

One strategy for addressing class imbalance involves the generation of one or more datasets, each with a different class distribution than the original. To achieve this, the two main categories of data sampling are utilized: undersampling and oversampling. Undersampling discards instances from the majority class, and if the process is random, the approach is known as *Random Undersampling* (RUS) [10]. Oversampling adds instances to the minority class, and if the process is random, the approach is known as *Random Oversampling* (ROS) [10]. *Synthetic Minority Over-sampling TEchnique* (SMOTE) [11] is a type of oversampling that generates new artificial instances between minority instances in close proximity to each other. Among ROS, RUS, SMOTE, and SMOTE variants, it has been shown that RUS imposes the lowest computational burden and registers the shortest training time [12].

Our work evaluates six data sampling approaches for addressing the effect that severe class imbalance has on Big Data analytics. To accomplish this, we compare results from two case studies involving imbalanced Big Data from different application domains. For the processing of Big Data, we use the Apache Spark [13] and Apache Hadoop frameworks [14–16].

The first case study is based on a Medicare fraud detection dataset (Combined dataset) [17], which is a combination of three Medicare datasets, with fraud labels derived from the *Office of Inspector General* (OIG) *List of Excluded Individuals/Entities* (LEIE) dataset [18]. The Combined dataset contains 759,740 instances (759,267 negatives and 473 positives) and 102 features. About 0.06% of instances are in the minority class. Results from the Medicare case study are not conclusive as to the best sampling approach, where *Area Under the Receiver Operating Characteristic Curve* (AUC) and *Geometric Mean* (GM) are concerned. For the AUC metric, the best sampled distribution ratios were obtained by RUS at 90:10, SMOTE at 65:35, and RUS at 90:10 for *Gradient-Boosted Trees* (GBT), *Logistic Regression* (LR), and *Random Forest* (RF), respectively. With regards to the GM metric, the best sampled distribution ratios were obtained by RUS at 50:50, SMOTE at 50:50, and RUS at 50:50 for GBT, LR, and RF, respectively. It is worth pointing out that RUS performed satisfactorily in this case study. For the AUC metric with LR, SMOTE (best value in sub-table) was labeled as group 'a' and RUS as group 'b' by Tukey's *Honestly Significant Difference* (HSD) test [19]. For the GM metric with LR, both SMOTE (best value in sub-table) and RUS were labeled as group 'a' by Tukey's HSD test.

The second case study, unlike the first, includes training data from one source (SlowlorisBig Dataset) [20] and test data from a separate source (POST dataset) [21]. Slowloris and POST are two types of Denial of Service (DOS) attacks. The SlowlorisBig Dataset contains 1,579,489 instances (1,575,234 negatives and 4,255 positives) and 11 features. About 0.27% of instances are in the minority class. The POST dataset contains 1,697,377 instances (1,694,986 negatives and 2,391 positives) and 13 features. About 0.14% of instances are in the minority class. In this study, RUS decisively outperforms the other five sampling approaches for the SlowlorisBig case study when measuring the performance with AUC and GM. For the AUC metric, the best sampled distribution ratios achieved with RUS were 90:10, 65:35, and 50:50 for GBT, LR, and RF, respectively. With regards to the GM metric, the best sampled distribution ratios achieved with RUS were 50:50, 65:35, and 50:50 for GBT, LR, and RF, respectively.

RUS is the best choice for both case studies based on its classification performance and the fact that it generates models with a significantly smaller number of samples, leading to a reduction in computational burden and training time. Our contribution involves the investigation of severe class imbalance with six data sampling approaches, and to the best of our knowledge, demonstrates a unique approach. Furthermore, the comparison of Big Data from different application domains enables us to better understand the extent to which our contribution is generalizable.

The remainder of this paper is organized as follows: "Related work" section provides an overview of literature related to data sampling methods that address severe class imbalance in Big Data; "Case studies datasets" section presents the details of the Medicare, SlowlorisBig, and POST datasets; "Methodologies" section describes the different aspects of the methodologies used to develop and implement our approach, including the Big Data processing framework, one-hot encoding, sampling ratios, sampling techniques, learners, performance metrics, and framework design. "Approach for case studies experiments" section provides additional information on the case studies; "Results and discussion" section presents and discusses our empirical results; and "Conclusion" section concludes our paper with a summary of the work presented and suggestions for related future work.

Related work

Two main categories for tackling class imbalance are data-level techniques and algorithm-level techniques [22]. Data-level techniques cover both data sampling and feature selection approaches. Data sampling approaches commonly include ROS, RUS, and SMOTE. In this section, we focus on related works associated with data sampling techniques that address severe class imbalance in Big Data.

In [23], Fernández et al. provide an insight into imbalanced Big Data classification outcomes and challenges. They compared RUS, ROS, and SMOTE using MapReduce with two subsets of the *Evolutionary Computation for Big Data and Big Learning* (ECBDL'14) dataset [24], while maintaining the original class ratio. The two subsets, one with 12 million instances and the other with 0.6 million, were both defined by a 98:2 class ratio. The original 631 features of the ECBDL'14 dataset were reduced to 90 features by the application of a feature selection algorithm [24, 25]. The authors examined the performance of RF and *Decision Tree* (DT) learners, using both Apache Spark in-memory computing (used with the MLlib library [26]) and Apache Hadoop MapReduce (used with the Mahout library [27]) frameworks. Some interesting conclusions emerged from the analysis: (1) Models using Apache Spark generally produced better classification results than models using Hadoop; (2) RUS performed better with less MapReduce partitions, while ROS performed better with more, indicating that the number of partitions in Hadoop impacts performance; (3) Apache Spark-based RF and DT produced better results with RUS compared to ROS. The best overall values of GM for ROS, RUS, and SMOTE were 0.706, 0.699, and 0.632, respectively. We note that the focus of [23] leaned toward demonstrating limitations of MapReduce rather than developing an effective solution for the high-class imbalance problem in Big Data. Secondly, different Big Data frameworks were used for some data sampling methods, making comparative conclusions unreliable. For example, the SMOTE implementation was done in Apache Hadoop, whereas RUS and ROS implementations were done in Apache Spark. Finally, the study does not indicate the sampling ratios (90:10, 75:25, etc.) used with RUS, ROS, and SMOTE, which means there is no means of assessing the impact of using various sampling ratio values on classification performance.

The experimentation by Del Río et al. in [28] analyzed the effect of increasing the oversampling ratio for extremely imbalanced Big Data. Their work relied on the Apache Hadoop framework for evaluating the MapReduce versions of RUS, ROS, and RF. The ECBDL'14 dataset served as the case study, and the MapReduce approach for Differential Evolutionary Feature Weighting (DEFW-BigData) algorithm was used to select the most influential features [25]. The full ECBDL'14 dataset was used, which contained approximately 32 million instances, a class ratio of 98:2, and 631 features. The authors showed that ROS slightly outperformed RUS with regards to the product of *True Positive Rate* (TP_{rate}) and *True negative Rate* (TN_{rate}). The best values for ROS and RUS were 0.489 and 0.483, respectively. The authors also observed that ROS had a very low TP_{rate} compared to TN_{rate}, which motivated further experimentation with a range of higher oversampling 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 oversampling rate was found to increase the TP_{rate} and lower the TN_{rate}, and the best overall results for [28] were obtained with an oversampling rate of 170%. This related work has limitations that are similar to those of [23]. However, there are additional issues such as the use of MapReduce, which is sensitive to severe class imbalance [29], as the only framework, and also the lack of inclusion of the popular SMOTE technique for comparison.

An analytical approach for predicting highway traffic accidents was proposed by Park et al. in [30], which involved classification modeling using the Apache Hadoop MapReduce framework. The authors implemented a modification of SMOTE for addressing a dataset of severely imbalanced traffic accidents, i.e., a class ratio of approximately 370:1, and a total of 524,131 instances defined by 14 features. After oversampling was performed, the minority class (accident) instances in the training dataset increased from 0.27% to 23.5%. A classification accuracy of 76.35% and a TP_{rate} of 40.83% were obtained by a LR classifier. In a similar experiment, the authors also experimented with SMOTE in a MapReduce framework (Apache Hadoop) [31] and obtained the best overall classification accuracy of 80.6% when the minority class reached about 30% of the training dataset, from the initial 0.14% of minority class instances. The original training dataset contained 1,024,541 instances, a class ratio of 710:1, and 13 features. For the studies presented in [30, 31], we point out that MapReduce is particularly sensitive to high-class imbalance in datasets, thus likely yielding sub-optimal classification performance. Second, we believe that the use of the Apache Spark framework may outperform the Apache Hadoop (MapReduce) framework. Finally, we remind the user of the main limitation of the accuracy classification metric. It is not a dependable metric because a severely imbalanced dataset with a 99.9% accuracy could have TP_{rate} and TN_{rate} values of approximately 0% and 100%, respectively.

In [32], Chai et al. examined severe class imbalance within the context of using statistical text classification to identify information technology health incidents. RUS was used to balance the majority and minority classes, i.e., 50:50, with the aim of comparing classification performances between the original, imbalanced dataset and the balanced dataset. The training dataset contained approximately 516,000 instances and 85,650 features, with about 0.3% of instances constituting the minority class. Regularized LR was selected as the classifier mainly due to its ability to avoid overfitting while using a very large set of features that is typical in text classification. Experimental results show that the F-measure scores were relatively similar with or without under-sampling, i.e., the balanced dataset did not affect classification performance. However, undersampling increased recall and decreased precision of the classifier. The best value of the F-measure was 0.99. One limitation of [32] relates to the question of why the authors only used a balanced ratio in their study, with no other ratios considered. Furthermore, no clear explanation was provided for the use of undersampling as the only data sampling technique in the study.

It should be noted that research on Big Data sampling techniques for addressing severe class imbalance is still in an embryonic state. As a result, literature searches on this narrow topic are not expected to yield prolific results.

Case studies datasets

Our work includes two case studies. The dataset used in the first case study came from a different application domain than the datasets used in the second case study. In the first case study, *Cross Validation* (CV) was performed on the Medicare dataset. In the second case study, the SlowlorisBig Dataset was used for training and the POST dataset for testing. The Medicare dataset is considered high dimensional (102 features), whereas the SlowlorisBig and POST datasets are not (11 and 13 features, respectively).

Medicare

To construct ML models for detecting Medicare fraud, we first combined three datasets: Medicare Physician and Other Supplier (Part B), years 2012 to 2015; Prescriber (Part D), years 2013 to 2015; and Durable Medical Equipment, Prosthetics, Orthotics and Supplies (DMEPOS) datasets from the *Centers for Medicare and Medicaid Services* (CMS) [33], years 2013 to 2015. The Part B dataset includes claims information for each procedure a physician/provider performs in a specified year. The Part D dataset provides claims information on prescription drugs provided through the Medicare Part D Prescription Drug Program in a specified year. The DMEPOS dataset includes claims for medical equipment, prosthetics, orthotics, and supplies that physicians/providers referred patients to for purchase or rent from a supplier in a specified year. The three Medicare datasets were joined into a Combined dataset, with fraud labels derived from the OIG's LEIE dataset. The Combined dataset contains 759,740 instances (759,267 negatives and 473 positives) and 102 features. About 0.06% of instances are in the minority class.

SlowlorisBig and POST

DOS attacks are carried out through various methods designed to deny network availability to legitimate users [34]. *Hypertext Transfer Protocol* (HTTP) contains several exploitable vulnerabilities and is often targeted for DOS attacks [35, 36]. During a Slowloris attack, numerous HTTP connections are kept engaged for as long as possible. Only partial requests are sent to a web server, and since these requests are never completed the available connections for legitimate users becomes zero. During a Slow HTTP POST attack, legitimate HTTP headers are sent to a target server. The message body of the exploit must be the correct size for communication between the attacker and the server to continue. Communication between the two hosts becomes a drawn-out process as the attacker sends messages that are relatively very small, tying up server resources. This effect is worsened if several POST transmissions are done in parallel.

Data collection for the Slowloris and POST attacks was performed within a real-world network setting. An ad hoc Apache web server, which was set up within a campus network environment, served as a viable target. A Slowloris.py attack script [37] and the Switchblade 4 tool from *Open Web Application Security Project* (OWASP) were used to generate attack packets for Slowloris and POST, respectively. Attacks were launched from a single host computer in hourly intervals. Attack configuration settings, such as connection intervals and number of parallel connections, were varied, but the same PHP form element on the web server was targeted during the attack. The resulting

SlowlorisBig Dataset contains 1,579,489 instances (1,575,234 negatives and 4255 positives) and 11 features. About 0.27% of instances are in the minority class. The resulting POST dataset contains 1,697,377 instances (1,694,986 negatives and 2391 positives) and 13 features. About 0.14% of instances are in the minority class.

Methodologies

This section provides insight into the methodologies for this experiment. It covers the Big Data framework, one-hot encoding, sampling ratios, sampling techniques, learners, performance metrics, and framework design.

Big Data framework

The processing and analysis of Big Data frequently requires specialized computational frameworks that benefit from the use of clusters and parallel algorithms. Two such frameworks are Apache Spark and MapReduce [27]. Apache Spark, referred to as Spark herein, is a framework for Big Data and ML that uses in-memory operations instead of the divide-and-conquer approach of MapReduce. Compared to MapReduce, the data processing speed of Spark is exponentially faster because MapReduce writes to and reads from hard drives. For this reason, we decided to use the in-memory implementation of Spark in our study.

In addition to Spark, we use the Apache Hadoop framework, which consists of several tools and technologies for Big Data, two of which are used in our work. *Hadoop Distributed File System* (HDFS) [38] can store large files across a large cluster of nodes, while *Yet Another Resource Negotiator* (YARN) [39] is used for job management and scheduling.

One-hot encoding

Through one-hot encoding, all categorical features in this work were converted into dummy variables for several reasons. One primary reason relates to the fact that some ML algorithms do not deal with categorical features in their raw form. Another reason is due to the high quantity of instances with missing values in the original datasets. Imputing these values, discarding instances with such values, and converting categorical features are three traditional solutions for addressing this issue. Because the number of instances with missing values is very high, imputing could change the nature of the data. Furthermore, discarding instances could result in the loss of valuable information. Hence, we decided against imputing values and discarding instances.

As an example of categorical feature conversion, a gender feature that is missing male and female categorical values will generate two new features, where the record with missing gender is filled with zeroes for both features. A drawback is that a feature with C distinct categories will generate C-1 new features, and this may increase the dimensionality of the feature space where the categorical values are too many. Another challenge may arise if the test set contains categorical values that do not exist in the training set and vice versa.

Sampling ratios

Unequal proportions of majority and minority instances are responsible for class imbalance issues, which may cause the ML algorithm to be biased toward the majority class during model training. In some cases, the positive class (class of interest) is completely ignored. For our work, we use six data sampling methods, generating five class ratios (distributions) for each method (i.e. 99:1, 90:10, 75:25, 65:35, and 50:50) in a majority to minority format of representation. The selected ratios were chosen to provide a good range of class distribution from perfectly balanced with a 50:50 ratio, through moderately balanced, to highly imbalanced with a 99:1 ratio. The inclusion of the highly imbalanced ratio facilitates the construction of a generalized curve and provides empirical information that aids in the selection of optimal ratios for this study.

Sampling techniques

This section is an overview of the six data sampling techniques used in our study. We selected one undersampling technique and five oversampling techniques, three of which are variants that focus on the boundary between the majority and the minority class.

1. RUS: This approach randomly discards instances from the majority class, resulting in a reduction of dataset size. Reducing the size of the majority class decreases computational burden, making analysis on very large datasets more manageable. The obvious disadvantage with RUS is the loss of potentially useful information, because instances of the majority class are randomly discarded [10]
2. ROS: This approach adds to the instances of the minority class by randomly duplicating observations belonging to that class with replacement. Oversampling increases the size of the dataset, potentially increasing computational costs. Since this technique duplicates minority class instances, it is susceptible to data overfitting [40].
3. SMOTE: This oversampling approach generates artificial instances [11], increasing the size of the minority class instances via *k*-nearest neighbors and sampling with replacement. SMOTE interpolates from original minority instances instead of just duplicating them. This method initially finds the *k*-nearest neighbors of the minority class for each minority instance. New instances are then generated in the direction of some or all of the nearest neighbors, depending on the oversampling percentage goal, by calculating the difference between the original minority example and its nearest neighbors and multiplying this difference by a random number (between 0 and 1).
4. *borderline-SMOTE* (SMOTEb): This approach [41] modifies the SMOTE algorithm by selecting the minority instances on the border of the minority decision region in the feature-space, only performing SMOTE on these instances. The number of majority neighbors of each minority instance is used to divide the minority class instances into three categories: SAFE, DANGER, or NOISE. Only the minority instances in the DANGER category are used to generate artificial instances. There are two types of SMOTEb, type 1 and type 2. Type 1 or *SMOTE-borderline1* (SMOTEb1) generates new, synthetic instances that belong to a class different from the original minority examples. Type 2 or *SMOTE-borderline2* (SMOTEb2) generates instances that can belong to any class.
5. *ADAptive SYNthetic* (ADASYN): This approach [42] is similar to SMOTE except that it focuses on generating instances adjacent to original minority examples that were

misclassified by a *k*-nearest neighbors classifier. As a result, more artificial instances will be generated in regions where the nearest neighbor rule is ignored.

RUS and ROS were implemented within the scalable libraries of Spark. SMOTE and its variants were implemented within imbalanced-learn [43], a toolbox with many predefined imbalanced solutions, including sampling.

Learners

We use three learners built for Apache Spark from MLlib (machine learning library). For our study, we use LR [44], RF [45], and GBT [46]. The default configurations are assumed, unless otherwise stated.

LR uses a sigmoidal, or logistic, function to generate values from [0,1] that can be interpreted as class probabilities. LR is similar to linear regression but uses a different hypothesis class to predict class membership. The bound matrix parameter was set to match the shape of the data so the algorithm knows the number of classes and features the dataset contains. The bound vector size was set to 1 for binomial regression, with no thresholds applied for binary classification.

RF is an ensemble approach building multiple decision trees. The classification results are calculated by combining the results of the individual trees, typically using majority voting. RF generates random datasets via sampling with replacement to build each tree, and selects features at each node automatically based on entropy and information gain. In this study, we set the number of trees to 100 and the max depth to 16. Additionally, the parameter that caches node IDs for each instance, was set to true and the maximum memory parameter was set to 1,024 megabytes in order to minimize training time. The setting that manipulates the number of features to consider for splits at each tree node was set to one-third, since this setting provided better results upon initial investigation. The maximum bins parameter, which is for discretizing continuous features, was set to 2 since we use one-hot encoding on categorical variables to avoid converting any numerical values as categorical.

GBT is an ensemble approach that trains each Decision Tree iteratively in order to minimize loss determined by the algorithm's loss function. During each iteration, the ensemble is used to predict the class for each instance in the training data. The predicted values are evaluated with the actual values allowing for the identification and correction of previously mislabeled instances. The parameter that caches node IDs for each instance was set to TRUE, and the maximum memory parameter was set to 1,024 MB to minimize training time.

Performance metrics

Our work records the confusion matrix for a binary classification problem, where the class of interest is usually the minority class and the opposite class is the majority class, i.e. positives and negatives, respectively. A related list of simple performance metrics [9] is explained as follows:

- *True positive* (TP) is the number of positive samples correctly identified as positive.

- *True negative* (TN) is the number of negative samples correctly identified as negative.
- *False positive* (FP), also known as Type I error, is the number of negative instances incorrectly identified as positive.
- *False negative* (FN), also known as Type II error, is the number of positive instances incorrectly identified as negative.
- $\mathrm{TP}_{\mathrm{rate}}$, also known as Recall or Sensitivity, is equal to TP / (TP + FN).
- $\mathrm{TN}_{\mathrm{rate}}$, also known as Specificity, is equal to TN / (TN + FP).

We used more than one performance metric to better understand the challenge of evaluating ML with severely imbalanced data. The first metric is AUC [47, 48], where an ROC curve depicts a learner's performance across all classifier decision thresholds. From this curve, the AUC obtained is a single value that ranges from 0 to 1, with a perfect classifier having a value of 1. AUC indicates the predictive potential of a binary classifier and seeks to maximize the joint performance of the classes via true positive rate (sensitivity/recall) and true negative rate (specificity). Additionally, due to the class imbalance in the datasets included in our work, we consider AUC a good metric for assessing classification performance. The second performance metric included in our study is GM, which indicates how well the model performs at the threshold where $\mathrm{TP}_{\mathrm{rate}}$ and $\mathrm{TN}_{\mathrm{rate}}$ are equal. GM is equal to $\sqrt{\mathrm{TP}_{\mathrm{rate}} \times \mathrm{TN}_{\mathrm{rate}}}$.

Framework design

The evaluation of the learners is performed using two approaches based on our case studies. The approach for the Medicare dataset uses *k*-fold CV. With this method, the model is trained and tested *k* times, where it is trained on *k*-1 folds each time and tested on the remaining fold. This is to ensure that all data are used in the classification. More specifically, we use stratified CV which tries to ensure that each class is approximately equally represented across each fold. In our study, we assigned a value of 5 to *k*: four folds for training and one fold for testing. Note that Spark does not support *k*-fold CV and thus we implemented our own version of CV for use with Spark scalable processing. The approach for the SlowlorisBig and POST datasets used the Training/Test method, with the former dataset utilized to train the model and the latter used to test.

We repeated the process of building and evaluating the models 10 times for each learner and dataset. The use of repeats helps to reduce bias due to bad random draws when generating the samples. The final performance result is the average over all 10 repeats.

Approach for case studies experiments

Case 1: Medicare

In this case study, statistics obtained after the application of sampling techniques on the Medicare dataset, i.e. undersampling and oversampling, are presented in Table 1. The number of positives and negatives when sampling has not been performed ("None" method) are also included in the table. The count of 379 positives in the table represents the quantity of minority instances within the four folds of training data, out of a total of 473 (positives within both test and training folds) in the dataset. Likewise, the count of

Table 1 Medicare sampling

Ratio	No sampling		Undersampling			Oversampling		
	Negatives	Positives	Negatives	Positives	Negatives %	Negatives	Positives	Positives%
(All:all)	607,414	379	–	–	–	–	–	–
(99:1)	–	–	37,521	379	6.18	607,414	6,135	1,618.86
(90:10)	–	–	3,411	379	0.56	607,414	67,490	17,807.51
(75:25)	–	–	1,137	379	0.19	607,414	202,471	53,422.52
(65:35)	–	–	704	379	0.12	607,414	327,069	86,297.91
(50:50)	–	–	379	379	0.06	607,414	607,414	160,267.55

Table 2 SlowlorisBig sampling

Ratio	No sampling		Undersampling			Oversampling		
	Negatives	Positives	Negatives	Positives	Negatives %	Negatives	Positives	Positives%
(All:all)	1,575,234	4255	–	–	–	–	–	–
(99:1)	–	–	421,245	4255	26.74	1,575,234	15,911	373.95
(90:10)	–	–	38,295	4255	2.43	1,575,234	175,026	4113.42
(75:25)	–	–	12,765	4255	0.81	1,575,234	525,078	12,340.26
(65:35)	–	–	7902	4255	0.50	1,575,234	848,203	19,934.26
(50:50)	–	–	4255	4255	0.27	1,575,234	1,575,234	37,020.78

607,414 negatives represents the quantity of majority instances within the four folds of training data, out of a total of 759,267 (negatives within both test and training folds). We can also see from Table 1 that oversampling with the 50:50 class ratio increases the original count of positives by over 160,000% due to the severe class imbalance in this dataset.

Case 2: SlowlorisBig and POST

In this case study, we built models using the SlowlorisBig Dataset and tested them on the POST dataset. These two datasets are in the same application domain but come from different sources. As in the first case study, we provided statistics (shown in Table 2) based on the datasets generated after the application of various sampling techniques.

Results and discussion

In this section, we present the results of the Medicare and SlowlorisBig case studies. As explained in the previous section, we generated five class ratios (50:50, 65:35, 75:25, 90:10, and 99:1) using six sampling techniques: RUS, ROS, SMOTE, SMOTEb1, SMOTEb2, and ADASYN. We included the full datasets ("all:all"), without any data sampling performed ("None" method), to serve as a baseline comparison. As mentioned in "Methodologies" section, our results were obtained by implementing three ML classifiers, i.e. GBT, LR, RF and evaluated with the AUC and GM performance metrics.

The results of our experiment for the full datasets, prior to sampling, are included in Table 3. The table shows the two metrics: AUC and GM.

The overall results for both Medicare and SlowlorisBig Datasets are presented by averaging the AUC in part (a) of Tables 4 and 5, respectively. Similarly, part (b) of both tables reports the average results for the GM metric. For parts (a) and (b) of both tables,

Table 3 No-sampling (all:all) results

Dataset	Learner	AUC	GM
Medicare	GBT	0.7905	0.0091
	LR	0.8155	0.0000
	RF	0.7938	0.0082
SlowlorisBig	GBT	0.6868	0.2517
	LR	0.5920	0.6449
	RF	0.867	0.0000

the highest value within each column (class distribution ratio) of each sub-table is in italic type, and the highest value within each row (sampling method) of each sub-table is underlined. As discussed in "Methodologies" section, we performed 5-fold CV for the Medicare case study while we used a Training/Test method for the SlowlorisBig case study. The average values shown are derived from 50 models (5-fold CV with 10 repeats) in the Medicare case study and 10 models in the SlowlorisBig case study.

AUC values for the Medicare dataset are shown in Table 4. The best performance, on average, for the GBT model was 0.81675 with RUS and a 90:10 ratio, followed by 0.80703, which was obtained by ROS with a 50:50 ratio. The lowest score of 0.62805 was obtained with ROS and a 90:10 ratio, which was a lower value than the score recorded for the GBT model with unsampled data. For the LR model associated with the Medicare dataset, the best performance was obtained by SMOTE, with values between 0.82211 and 0.82781 for distribution ratios of 90:10, 75:25, 65:35, and 50:50. However, the LR model yielded a value of 0.82011 using RUS and a 99:1 ratio, which was better than the score of 0.81554 for the LR model with unsampled data. The lowest score of 0.6621 for the LR model was obtained with ROS and a 99:1 ratio. Finally, for the RF learner, RUS outperformed the other sampling methods with a score of 0.82793 for the 90:10 ratio.

GM values for the Medicare dataset are also shown in Table 4. The reader should note that GM records the model performance outcome of the confusion matrix, unlike AUC, which provides an overall performance. Thus, we observed that the balanced ratio of 50:50 performed the best while the performances decrease when the ratios become progressively more imbalanced. RUS yielded the best results for GBT and RF. However, with the LR model, SMOTE performed the best with a GM score of 0.75345, followed by ROS, ADASYN, and then RUS.

For the AUC metric of the SlowlorisBig Dataset, shown in Table 5, the score for the GBT model with RUS was 0.97226, corresponding to a 90:10 ratio. However, the RUS ratios of 75:25, 65:35, and 50:50 also performed well when compared to the other sampling methods. The lowest score of 0.46056 was obtained for the ADASYN method and a ratio of 50:50, which is considered a worse score than a random guess (AUC value of 0.5). With regards to the LR model, RUS with a ratio of 65:35 produced the highest value of 0.97113. ADASYN with a 65:35 ratio recorded the lowest value of 0.43311. However, the second best method after RUS was also ADASYN with a 90:10 ratio and a score of 0.77948. Lastly, with regards to the RF model, the best AUC value was obtained using RUS with a 50:50 ratio; however, two oversampling methods, ROS and SMOTE, also produced decent results.

Table 4 Case 1: Medicare results

Learner	Method	(All:all)	(99:1)	(90:10)	(75:25)	(65:35)	(50:50)
(a) AUC							
GBT	None	0.79047	–	–	–	–	–
	RUS	–	*0.80373*	*0.81675*	*0.80405*	*0.79127*	0.77587
	ROS	–	0.74328	0.62805	0.72565	0.76417	*0.80703*
	ADASYN	–	0.71368	0.69611	0.69586	0.69675	0.69351
	SMOTE	–	0.73903	0.72194	0.72634	0.72986	0.73439
	SMOTEb1	–	0.68831	0.67235	0.65831	0.65448	0.66498
	SMOTEb2	–	0.68917	0.67780	0.66209	0.66312	0.66730
LR	None	0.81554	–	–	–	–	–
	RUS	–	*0.82011*	0.81868	0.81553	0.80998	0.79415
	ROS	–	0.66210	0.68306	0.75298	0.79036	0.81547
	ADASYN	–	0.81205	0.81622	0.81758	0.81384	0.81578
	SMOTE	–	0.81306	*0.82211*	*0.82685*	*0.82781*	*0.82413*
	SMOTEb1	–	0.74471	0.73845	0.73526	0.74014	0.73484
	SMOTEb2	–	0.72167	0.71599	0.72523	0.72752	0.72426
RF	None	0.79383	–	–	–	–	–
	RUS	–	*0.81515*	*0.82793*	*0.81503*	*0.80619*	*0.79546*
	ROS	–	0.77538	0.75640	0.75728	0.76989	0.79315
	ADASYN	–	0.74537	0.73496	0.72920	0.73266	0.73577
	SMOTE	–	0.77417	0.76921	0.77629	0.77443	0.76790
	SMOTEb1	–	0.76460	0.74777	0.75695	0.75844	0.75883
	SMOTEb2	–	0.76440	0.75071	0.75155	0.75282	0.74967
(b) GM							
GBT	None	0.00907	–	–	–	–	–
	RUS	–	*0.08674*	*0.37061*	*0.60384*	*0.67830*	*0.70412*
	ROS	–	0.01234	0.14263	0.34824	0.50723	0.69501
	ADASYN	–	0.00205	0.00413	0.05390	0.12527	0.30430
	SMOTE	–	0.01027	0.06270	0.22959	0.33785	0.47255
	SMOTEb1	–	0.03254	0.20534	0.28603	0.33670	0.40159
	SMOTEb2	–	0.04371	0.18432	0.26794	0.32180	0.38951
LR	None	0	–	–	–	–	–
	RUS	–	0.13376	*0.45411*	*0.66222*	*0.72088*	0.73044
	ROS	–	0.05917	0.36425	0.58388	0.67673	0.75224
	ADASYN	–	0.06607	0.35955	0.59097	0.69207	0.74657
	SMOTE	–	0.12602	0.45052	0.64526	0.71975	*0.75345*
	SMOTEb1	–	*0.13877*	0.37785	0.50841	0.55796	0.59091
	SMOTEb2	–	0.10953	0.35552	0.50170	0.54910	0.58911
RF	None	0.00823	–	–	–	–	–
	RUS	–	*0.09315*	*0.26700*	*0.56838*	*0.67842*	*0.72590*
	ROS	–	0.00909	0.01027	0.03608	0.08623	0.29951
	ADASYN	–	0.03665	0.10203	0.16093	0.20660	0.24092
	SMOTE	–	0.04778	0.16448	0.23132	0.26808	0.31371
	SMOTEb1	–	0.04571	0.08312	0.12967	0.14453	0.18056
	SMOTEb2	–	0.03546	0.07203	0.10473	0.10693	0.14532

The highest value within each column (class distribution ratio) of each sub-table is in italic type, and the highest value within each row (sampling method) of each sub-table is underlined

In relation to the SlowlorisBig performance for the GM metric, shown in part (b) of Table 5, RUS outperformed all the other oversampling methods for all three learners. Note that when the model fails to correctly classify any positive instances during all 10

Table 5 Case 2: SlowlorisBig results

Learner	Method	(All:all)	(99:1)	(90:10)	(75:25)	(65:35)	(50:50)
(a) AUC							
GBT	None	0.68678	–	–	–	–	–
	RUS	–	*0.84644*	*0.97226*	*0.96541*	*0.96724*	*0.96685*
	ROS	–	0.65312	0.50947	0.69950	0.66151	0.65531
	ADASYN	–	0.47154	0.74951	0.68449	0.82351	0.46056
	SMOTE	–	0.57069	0.58314	0.69230	0.63663	0.70906
	SMOTEb1	–	0.70283	0.65169	0.62276	0.62191	0.67668
	SMOTEb2	–	0.69876	0.62302	0.63359	0.66083	0.71559
LR	None	0.59203	–	–	–	–	–
	RUS	–	*0.62018*	*0.84740*	*0.90919*	*0.97113*	*0.95052*
	ROS	–	0.59869	0.63752	0.60610	0.60996	0.62989
	ADASYN	–	0.77948	0.49306	0.43431	0.43311	0.46447
	SMOTE	–	0.60657	0.64287	0.61986	0.62587	0.61301
	SMOTEb1	–	0.59232	0.59301	0.59212	0.59242	0.59190
	SMOTEb2	–	0.59257	0.59164	0.59254	0.59189	0.59143
RF	None	0.86773	–	–	–	–	–
	RUS	–	0.88343	0.88444	0.91207	*0.95425*	*0.96045*
	ROS	–	0.88391	0.90186	0.91679	0.93715	0.95694
	ADASYN	–	0.75805	0.68151	0.48584	0.46859	0.40685
	SMOTE	–	*0.88436*	*0.89994*	*0.91701*	0.94070	0.95690
	SMOTEb1	–	0.87027	0.85896	0.88157	0.87659	0.86275
	SMOTEb2	–	0.87138	0.88829	0.86941	0.87098	0.86720
(b) GM							
GBT	None	0.25168	–	–	–	–	–
	RUS	–	*0.67700*	*0.83073*	*0.90015*	*0.94949*	*0.96174*
	ROS	–	0.48453	0.34405	0.53269	0.52886	0.49330
	ADASYN	–	0.24369	0.31263	0.16892	0.31140	0.10138
	SMOTE	–	0.47393	0.34644	0.59461	0.44976	0.57714
	SMOTEb1	–	0.29552	0.30041	0.30273	0.32697	0.26873
	SMOTEb2	–	0.28809	0.26814	0.27291	0.27253	0.28508
LR	None	0.64486	–	–	–	–	–
	RUS	–	0.62135	*0.82268*	*0.90304*	*0.96983*	*0.94733*
	ROS	–	0.66742	0.72338	0.69552	0.70111	0.72352
	ADASYN	–	*0.76272*	0.41830	0.37992	0.38049	0.41591
	SMOTE	–	0.64121	0.72341	0.71979	0.72363	0.68929
	SMOTEb1	–	0.64057	0.64489	0.64421	0.64489	0.64038
	SMOTEb2	–	0.63685	0.63808	0.64065	0.64097	0.63461
RF	None	0	–	–	–	–	–
	RUS	–	0	0.15255	*0.56097*	*0.65159*	*0.90195*
	ROS	–	0	*0.38138*	0.42997	0.64770	0.65151
	ADASYN	–	0	0	0	0	0
	SMOTE	–	0	0.34325	0.53447	0.64772	0.65155
	SMOTEb1	–	0	0	0	0	0
	SMOTEb2	–	0	0	0	0	0

The highest value within each column (class distribution ratio) of each sub-table is in italic type, and the highest value within each row (sampling method) of each sub-table is underlined

runs, the GM scores will be zero as shown with the RF case. We clearly can see that changing the performance metric may lead to different conclusions. Measuring the performance using AUC may give a general estimate of overall model performance when the threshold between TP_{rate} and *False Positive Rate* (FP_{rate}) is varied. On the other hand, measuring the performance using GM really means taking the square root of the product of TP_{rate} and TN_{rate} at a threshold where both rates are equal.

Figure 1 illustrates the results from Tables 4 and 5. It is noticeable that, on average, RUS as the only undersampling method used in our study outperformed the other six oversampling techniques plus the full, unsampled data. However, the average can be very misleading in statistics. For instance, Fig. 1 shows that ADASYN, with a ratio of 99:1, performed better on average than the other six sampling methods when building the LR models. It is also noticeable that the conclusion differs when comparing the AUC results with those obtained using the GM performance metric, especially when building the RF model.

The use of average values for variations of repetitive model building statistically enhances the score results assigned to the models. In addition, to demonstrate statistical significance of the observed experimental results, a hypothesis test is performed with *ANalysis Of VAriance* (ANOVA) [49], followed by post hoc analysis with Tukey's HSD test. ANOVA is a statistical test determining whether the means of one or several independent factors are significant. Tukey's HSD assigns group letters indicating the significance factors between each level.

We investigated the intersection of both factors (sampling techniques and class distribution ratios) to determine their effect on the three learners (GBT, RF, LR). If the *p*-value

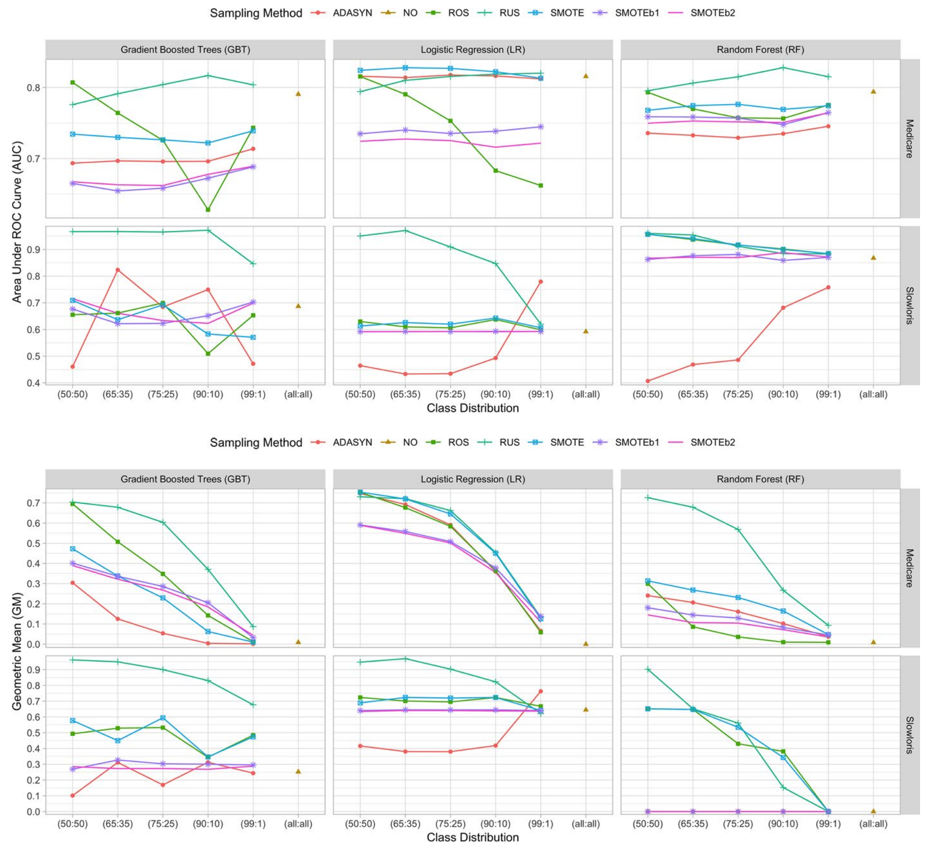

Fig. 1 Results for averages of sampling methods ratios

in the ANOVA table is less than or equal to a certain level (0.05), the associated factor is significant. A 95% ($\alpha = 0.05$) significance level for ANOVA and other statistical tests is the most commonly used value. In this work we obtained a total of 12 two-factor and 72 one-way ANOVA tables. We saw no need to show the ANOVA results as significance factors are implied in the Tukey's HSD tests, which are derived from the ANOVA tables.

Tables 6 and 7 relate to the Medicare and SlowlorisBig Datasets, respectively, and show the results of the Tukey's HSD test. The tables show the significance between the performance metric and sampling approach for each learner. The factors are ordered by the average of the performance metrics used and show the number of repetitions "r". We also show the standard deviation "std" for the repetitive models for each factor. The tables are also associated with the maximum, minimum, first quartile (Q25), second quartile (Q50), and third quartile (Q75). Q25 is the middle point between the minimum number and the median of the results. Q50 is the median of the repetitive results. Q75 is the middle point value between the median and the maximum performance of the model. From Tables 6 and 7, we see that the medians of the sampling method can be higher and, in some cases, lower than the averages.

SMOTE and its variants (SMOTEb1, SMOTEb2, and ADASYN) performed satisfactorily with the Medicare dataset and the LR learner, but there does not appear to be any value in using the more specific borderline cases to generate artificial examples. Using traditional SMOTE, which randomly selected instances from all generated examples, shows better average AUC scores (among the 10 runs for each sampling method and ratio) in more cases than its variants of SMOTEb1, SMOTEb2, and ADASYN. This could indicate a lack of distinct borders around the class labels from the k-nearest neighbors approach. The reason that the SMOTE variants perform similarly using RF, and not LR or GBT, is most likely due to the majority voting results across all trees. GBT, while using trees in an ensemble fashion, iteratively adjusts weights based on prior classification performance indicators, and thus is not able to take advantage of different sampled data subsets to generate the final classification as with RF.

Tables 8 and 9 present the results of Tukey's HSD test for the Medicare and SlowlorisBig Datasets, respectively. The results indicate the significance between class ratio and sampling approach for each learner.

Based on ANOVA, an empty column (missing group letters) indicates there is no significance between the factor levels. Thus, all of these levels were assigned to group 'a' by the Tukey's HSD test. Furthermore, an "NA" assignment instead of a group letter means that RF failed to classify any true positives, as is the case for SMOTEb1, SMOTEb2, and ADASYN.

The first row shows the group letters for all the ratios combined with the full, unsampled dataset. Group values (e.g. 'a' to 'e') indicate significant differences between the factor levels, or ratios, with the best group assigned the letter 'a'. Note that the full, unsampled dataset ("all:all" ratio) is included for comparative purposes. The ranked groups assigned by Tukey's HSD test corroborate the previously presented results shown in Tables 4 and 5 regarding the performance with and without data sampling.

To visualize the group 'a' results of Tables 8 and 9, Fig. 2 has been included. From the box plot distributions shown in this figure, we see that RUS outperforms all other sampling approaches for the SlowlorisBig case study when measuring the

Table 6 Case 1: Medicare-Tukey's HSD test

Learner	Sampling	AUC	std	r	g	Min	Max	Q25	Q50	Q75
(a) AUC										
GBT	RUS	0.79833	0.02599	250	a	0.72815	0.87018	0.78092	0.80045	0.81537
	None	0.79047	0.02386	50	a	0.72580	0.83013	0.78059	0.79586	0.80595
	ROS	0.73363	0.06754	250	b	0.51519	0.84819	0.70903	0.74192	0.77947
	SMOTE	0.73031	0.02584	250	b	0.64410	0.81724	0.71385	0.72880	0.74982
	ADASYN	0.69918	0.02609	250	c	0.61985	0.76370	0.68276	0.69946	0.71667
	SMOTEb2	0.67189	0.03213	250	d	0.57265	0.74786	0.65170	0.67248	0.69508
	SMOTEb1	0.66769	0.03720	250	d	0.48250	0.76948	0.64574	0.66988	0.69252
LR	SMOTE	0.82279	0.02125	250	a	0.75783	0.87290	0.81044	0.82237	0.83636
	None	0.81554	0.02227	50	ab	0.75532	0.84700	0.80752	0.81924	0.82659
	ADASYN	0.81509	0.02287	250	ab	0.74781	0.88334	0.80130	0.81666	0.83065
	RUS	0.81169	0.02040	250	b	0.73199	0.86455	0.80016	0.81220	0.82536
	ROS	0.74079	0.06836	250	c	0.55630	0.85671	0.69202	0.75347	0.79658
	SMOTEb1	0.73868	0.02748	250	c	0.66533	0.81191	0.71970	0.73888	0.75844
	SMOTEb2	0.72293	0.03287	250	d	0.61406	0.80044	0.70363	0.72765	0.74389
RF	RUS	0.81195	0.02373	250	a	0.74285	0.86547	0.79696	0.81221	0.82930
	None	0.79383	0.02306	50	b	0.74416	0.83161	0.77569	0.79317	0.81477
	SMOTE	0.77240	0.02304	250	c	0.70450	0.84333	0.75649	0.77252	0.78692
	ROS	0.77042	0.02790	250	c	0.70014	0.85378	0.75188	0.77028	0.78984
	SMOTEb1	0.75732	0.02536	250	d	0.66211	0.81080	0.74231	0.76021	0.77356
	SMOTEb2	0.75383	0.02794	250	d	0.68869	0.82191	0.73425	0.75200	0.77434
	ADASYN	0.73559	0.02654	250	e	0.66474	0.80357	0.71806	0.73933	0.75122
Learner	**Sampling**	**GM**	**std**	**r**	**g**	**Min**	**Max**	**Q25**	**Q50**	**Q75**
(b) GM										
GBT	RUS	0.48872	0.23777	250	a	0	0.78014	0.33953	0.60566	0.68945
	ROS	0.34109	0.25087	250	b	0	0.77439	0.10295	0.35164	0.52541
	SMOTEb1	0.25244	0.13924	250	c	0	0.50945	0.17726	0.27133	0.36454
	SMOTEb2	0.24145	0.13200	250	cd	0	0.49887	0.14570	0.25059	0.33908
	SMOTE	0.22259	0.17815	250	d	0	0.56455	0	0.22840	0.37532
	ADASYN	0.09793	0.12322	250	e	0	0.39311	0	0	0.17759
	None	0.00907	0.03150	50	f	0	0.14509	0	0	0
LR	RUS	0.54028	0.23037	250	a	0	0.77315	0.41864	0.66278	0.71993
	SMOTE	0.53900	0.23596	250	a	0	0.80154	0.42166	0.64136	0.72653
	ADASYN	0.49105	0.25417	250	b	0	0.80124	0.32327	0.59041	0.70910
	ROS	0.48725	0.25459	250	b	0	0.79442	0.33676	0.57923	0.69986
	SMOTEb1	0.43478	0.17227	250	c	0	0.67841	0.35173	0.49188	0.56532
	SMOTEb2	0.42099	0.18294	250	c	0	0.70354	0.32119	0.48474	0.56333
	None	0	0	50	d	0	0	0	0	0
RF	RUS	0.46657	0.24978	250	a	0	0.77101	0.25077	0.57335	0.69443
	SMOTE	0.20508	0.10625	250	b	0	0.43130	0.14498	0.22834	0.28849
	ADASYN	0.14943	0.09303	250	c	0	0.35404	0.10257	0.14575	0.22886
	SMOTEb1	0.11672	0.07686	250	d	0	0.30758	0.10241	0.14469	0.17743
	SMOTEb2	0.09289	0.07056	250	de	0	0.23024	0	0.10260	0.14509
	ROS	0.08824	0.12095	250	e	0	0.39744	0	0	0.14505
	None	0.00823	0.02819	50	f	0	0.10314	0	0	0

Table 7 Case 2: SlowlorisBig-Tukey's HSD test

Learner	Sampling	AUC	std	r	g	Min	Max	Q25	Q50	Q75
(a) AUC										
GBT	RUS	0.94364	0.08565	50	a	0.56736	0.97822	0.96356	0.96704	0.97073
	None	0.68678	0.11066	10	b	0.48887	0.85775	0.64388	0.67656	0.74637
	SMOTEb2	0.66636	0.15255	50	b	0.35152	0.97593	0.58214	0.67336	0.75517
	SMOTEb1	0.65517	0.15317	50	b	0.35417	0.96539	0.52471	0.67769	0.75147
	SMOTE	0.63836	0.17643	50	b	0.43299	0.98249	0.45329	0.65340	0.68739
	ADASYN	0.63792	0.27363	50	b	0.18138	0.98483	0.45072	0.47832	0.96347
	ROS	0.63578	0.16737	50	b	0.43522	0.98169	0.45196	0.65518	0.68476
LR	RUS	0.85968	0.15064	50	a	0.46331	0.98434	0.74674	0.92661	0.96904
	SMOTE	0.62164	0.04412	50	b	0.47496	0.67054	0.59907	0.63122	0.65456
	ROS	0.61643	0.04297	50	b	0.49468	0.67039	0.59864	0.60161	0.65380
	SMOTEb1	0.59235	0.00155	50	b	0.58955	0.59388	0.59043	0.59325	0.59336
	None	0.59203	0.00181	10	bc	0.58977	0.59365	0.59001	0.59324	0.59347
	SMOTEb2	0.59201	0.00166	50	bc	0.58950	0.59382	0.58991	0.59314	0.59329
	ADASYN	0.52089	0.13781	50	c	0.42229	0.89815	0.43665	0.45708	0.49882
RF	SMOTE	0.91978	0.02812	50	a	0.85957	0.96023	0.90119	0.91684	0.94535
	ROS	0.91933	0.02724	50	a	0.86641	0.96169	0.90193	0.91198	0.94159
	RUS	0.91893	0.03494	50	a	0.85684	0.96880	0.89140	0.91016	0.95740
	SMOTEb2	0.87345	0.01368	50	b	0.85486	0.90771	0.86248	0.87021	0.88223
	SMOTEb1	0.87003	0.02088	50	b	0.79088	0.91191	0.85906	0.86786	0.88263
	None	0.86773	0.00890	10	b	0.85090	0.88340	0.86338	0.86753	0.87333
	ADASYN	0.56017	0.15056	50	c	0.33384	0.87529	0.45101	0.51294	0.67599

Learner	Sampling	GM	std	r	g	Min	Max	Q25	Q50	Q75
(b) GM										
GBT	RUS	0.86382	0.18048	50	a	0.24356	0.97083	0.79393	0.94445	0.97021
	SMOTE	0.48838	0.16479	50	b	0.08179	0.64737	0.33031	0.60297	0.60528
	ROS	0.47668	0.13952	50	b	0.23670	0.64740	0.33031	0.50990	0.60513
	SMOTEb1	0.29887	0.11561	50	c	0.23672	0.56217	0.24196	0.24197	0.24369
	SMOTEb2	0.27735	0.08665	50	c	0.23672	0.56291	0.24196	0.24197	0.24369
	None	0.25168	0.10408	10	c	0.08180	0.51158	0.23672	0.24196	0.24326
	ADASYN	0.22760	0.11021	50	c	0.05009	0.33034	0.07652	0.24369	0.33026
LR	RUS	0.85284	0.15640	50	a	0.38133	0.97066	0.72452	0.91562	0.97001
	ROS	0.70219	0.07309	50	b	0.44269	0.72367	0.72336	0.72338	0.72338
	SMOTE	0.69947	0.08142	50	b	0.38133	0.72455	0.72336	0.72338	0.72365
	None	0.64486	0.00017	10	bc	0.64473	0.64506	0.64473	0.64473	0.64506
	SMOTEb1	0.64299	0.00804	50	bc	0.60116	0.64570	0.64473	0.64473	0.64506
	SMOTEb2	0.63823	0.01460	50	c	0.58532	0.64506	0.64473	0.64473	0.64473
	ADASYN	0.47147	0.15695	50	d	0.37992	0.91572	0.38026	0.38128	0.46906
RF	RUS	0.45341	0.34982	50	a	0	0.96438	0	0.63766	0.64780
	SMOTE	0.43540	0.25934	50	a	0	0.71944	0.37773	0.63642	0.64454
	ROS	0.42211	0.24530	50	a	0	0.72048	0.37757	0.38138	0.64454

performance with GM and AUC. It is also clear that RUS performs the best in some situations, or adequately in others, for all methods and/or ratios. Overall, we can safely state that RUS is the best choice for both case studies as it results in models with a significantly smaller number of samples, thus reducing computational burden

Table 8 Case 1: Medicare-Tukey's HSD for Ratios group test

Learner		Gradient-Boosted Trees						Logistic Regression						Random Forest					
		RUS	ROS	SMOTE	SMOTEb1	SMOTEb2	ADASYN	RUS	ROS	SMOTE	SMOTEb1	SMOTEb2	ADASYN	RUS	ROS	SMOTE	SMOTEb1	SMOTEb2	ADASYN
AUC	Ratio	a	b	b	d	d	c	a	b	a	b	c	a	a	b	b	c	c	d
	(50:50)	d	a	bc	c	c	c	b	a	abc	b	b	–	c	a	b	bc	b	bc
	(65:35)	bc	b	bc	c	c	c	a	b	a	b	b	–	bc	bc	b	bc	b	bc
	(75:25)	ab	c	bc	c	c	c	a	c	ab	b	b	–	b	c	b	bc	b	c
	(90:10)	a	d	c	bc	bc	c	a	d	abc	b	b	–	a	c	b	c	b	bc
	(99:1)	b	c	b	b	b	b	a	e	c	b	b	–	b	b	b	b	b	b
	(All:all)	c	a	a	a	a	a	a	a	bc	a	a	–	c	a	a	a	a	a
GM	Ratio	a	b	c	c	c	d	a	ab	a	b	b	ab	a	d	b	cd	d	c
	(50:50)	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a
	(65:35)	b	b	b	b	b	b	a	b	b	b	b	b	b	b	b	b	b	b
	(75:25)	c	c	c	c	c	c	b	c	c	c	c	c	c	c	c	b	b	c
	(90:10)	d	d	d	d	d	d	c	d	d	d	d	d	d	c	d	c	c	d
	(99:1)	e	e	e	e	e	d	d	e	e	e	e	e	e	c	e	d	d	e
	(All:all)	f	e	e	e	f	d	e	f	f	f	f	f	f	c	f	e	d	e

Table 9 Case2: SlowlorisBig—Tukey's HSD for ratios group test

Learner		Gradient-Boosted Trees						Logistic Regression						Random Forest					
RUS			ROS	SMOTE	SMOTEb1	SMOTEb2	ADASYN	RUS	ROS	SMOTE	SMOTEb1	SMOTEb2	ADASYN	RUS	ROS	SMOTE	SMOTEb1	SMOTEb2	ADASYN
AUC	Ratio	a	b	b	b	b	b	a	b	b	b	b	c	a	a	a	b	ab	c
	(50:50)	a	–	–	–	–	b	a	–	–	–	–	cd	a	a	a	–	b	c
	(65:35)	a	–	–	–	–	a	a	–	–	–	–	d	a	b	b	–	b	c
	(75:25)	a	–	–	–	–	ab	ab	–	–	–	–	d	b	c	c	–	b	c
	(90:10)	a	–	–	–	–	ab	b	–	–	–	–	c	c	d	d	–	a	b
	(99:1)	b	–	–	–	–	b	c	–	–	–	–	a	c	e	e	–	b	b
	(All:all)	c	–	–	–	–	ab	c	–	–	–	–	b	d	f	f	–	b	a
GM	Ratio	a	b	b	c	c	c	a	b	bc	bc	c	d	a	a	a	b	b	b
	(50:50)	a	a	a	–	–	a	a	–	–	–	–	a	a	a	a	NA	NA	NA
	(65:35)	a	a	a	–	–	a	a	–	–	–	–	b	b	a	a	NA	NA	NA
	(75:25)	a	ab	ab	–	–	ab	ab	–	–	–	–	c	b	b	b	NA	NA	NA
	(90:10)	ab	ab	ab	–	–	ab	b	–	–	–	–	c	c	b	c	NA	NA	NA
	(99:1)	b	bc	bc	–	–	bc	c	–	–	–	–	c	d	c	d	NA	NA	NA
	(All:all)	c	c	c	–	–	c	c	–	–	–	–	c	d	c	d	NA	NA	NA

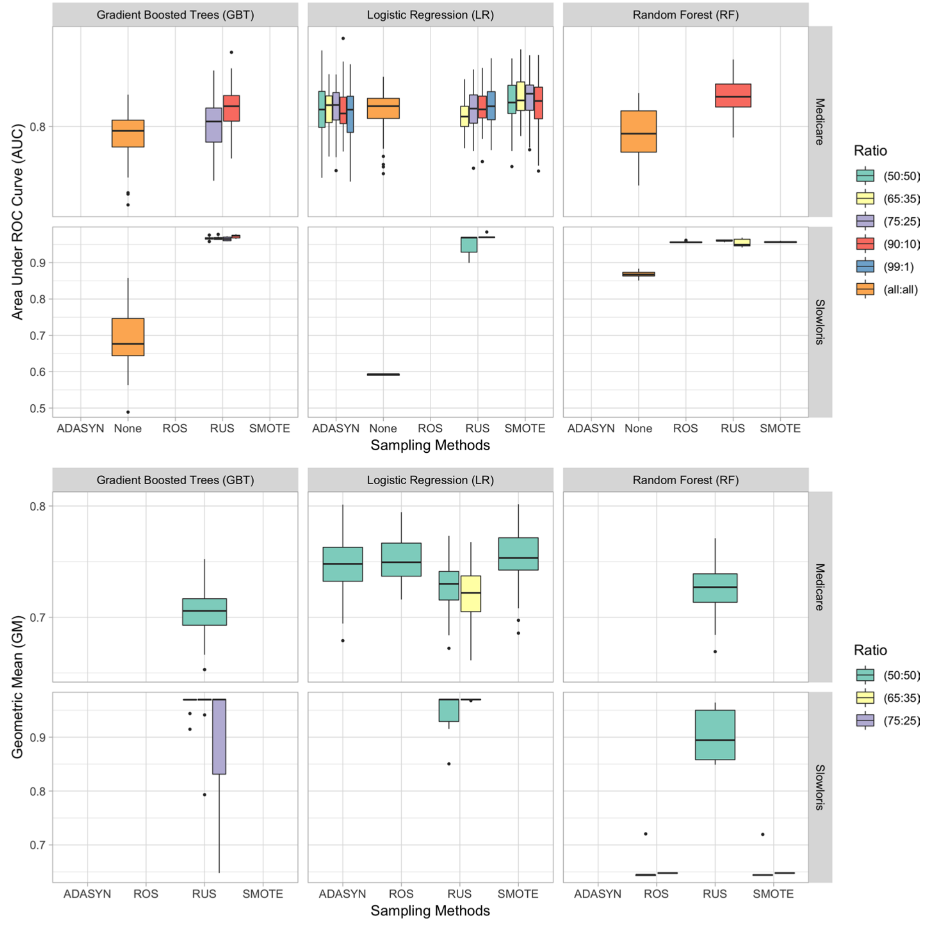

Fig. 2 Tukey's HSD test-Group a

and training time. For more visualization detail, readers may refer to Fig. 3 in Appendix A, which displays box plots for all the models built in this study.

Conclusion

Our work uniquely evaluates six data sampling approaches for addressing the effect that severe class imbalance has on Big Data analytics. To accomplish this, we compare results from two case studies involving imbalanced Big Data from different application domains. The outcome of this comparison enables us to better understand the extent to which our contribution is generalizable.

Results from the Medicare case study are not firmly conclusive for determining the best sampling approach where AUC and GM are concerned. For the AUC metric, the best sampled distribution ratios were obtained by RUS at 90:10, SMOTE at 65:35, and RUS at 90:10 for GBT, LR, and RF, respectively. With regards to the GM metric, the best sampled distribution ratios were obtained by RUS at 50:50, SMOTE at 50:50, and RUS at 50:50 for GBT, LR, and RF, respectively. It should be noted that RUS performed adequately in this first case study. For the AUC metric with LR, SMOTE (best value in sub-table) was labeled as group 'a' and RUS as group 'b'. For the GM metric with LR, both SMOTE (best value in sub-table) and RUS were labeled as group 'a'.

We show that RUS convincingly outperforms the other five sampling approaches for the SlowlorisBig case study when measuring the performance with AUC and GM. For the AUC metric, the best sampled distribution ratios achieved with RUS were 90:10, 65:35, and 50:50 for GBT, LR, and RF, respectively. With regards to the GM metric, the best sampled distribution ratios achieved with RUS were 50:50, 65:35, and 50:50 for GBT, LR, and RF, respectively.

Based on its classification performance in both case studies, RUS is the best choice as it generates models with a significantly smaller number of samples, which leads to a reduction in computational burden and training time. Future work using our evaluation methodology will involve additional performance metrics, such as *Area Under the Precision-Recall Curve* (AUPRC), and the investigation of Big Data from other application domains.

Abbreviations

AA: amino acid; ADASYN: ADAptive SYNthetic; AI: Artificial Intelligence; ANOVA: ANalysis Of VAriance; AUC: Area Under the Receiver Operating Characteristic Curve; AUPRC: Area Under the Precision Recall Curve; AUPRC: Area Under the Precision-Recall Curve; CASP9: 9th Community Wide Experiment on the Critical Assessment of Techniques for Protein Structure Prediction; CM: Contact Map; CMS: Centers for Medicare and Medicaid Services; CV: Cross Validation; DM: Data Mining; DOS: Denial of Service; DRF: Distributed Random Forest; DT: Decision Tree; ECBDL'14: Evolutionary Computation for Big Data and Big Learning; EUS: Evolutionary Undersampling; FAU: Florida Atlantic University; FI: Feature Importance; FN: False Negative; FP: False Positive; $\mathrm{FP}_{\mathrm{rate}}$: False Positive Rate; FS: Feature Selection; GBT: Gradient-Boosted Trees; GM: Geometric Mean; HDFS: Hadoop Distributed File System; HPC: High Performance Computing; HSD: Honestly Significant Difference; HTTP: Hypertext Transfer Protocol; IG: Information Gain; IRSC: Indian River State College; k-NN: k-nearest neighbor; LEIE: List of Excluded Individuals/Entities; LR: Logistic Regression; MB: megabytes; ML: Machine Learning; NSF: National Science Foundation; OIG: Office of Inspector General; OWASP: Open Web Application Security Project; PCA: Principal Component Analysis; PCARDE: Principal Components Analysis Random Discretization Ensemble; PDB: Protein Data Bank; PSP: Protein Structure Prediction; RF: Random Forest; ROS: Random Oversampling; ROSEFW-RF: Random OverSampling and Evolutionary Feature Weighting for Random Forest; RPII: Rare-PEARs II; RPII: Rare-PEARs; RUS: Random Undersampling; SMOTE: Synthetic Minority Over-sampling TEchnique; SMOTEb: borderline-SMOTE; SMOTEb1: SMOTE-borderline1; SMOTEb2: SMOTE-borderline2; SVMs: Support Vector Machines; TN: True Negative; $\mathrm{TN}_{\mathrm{rate}}$: True Negative Rate; TP: True Positive; $\mathrm{TP}_{\mathrm{rate}}$: True Positive Rate; YARN: Yet Another Resource Negotiator.

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 *National Science Foundation* (NSF) (CNS-1427536). Opinions, findings, conclusions, or recommendations in this paper are the authors' and do not reflect the views of the NSF.

Authors' contributions

TH carried out the conception and design of the research, performed the implementation and experimentation, performed the evaluation and validation, and drafted the manuscript. TH, and JJL, RAB performed the primary literature review for this work. RAB prepared the Medicare dataset. All authors provided feedback to TH and helped shape the research. TH and JJL manuscript the work. TMK introduced this topic to TH, and helped to complete and finalize this work. All authors read and approved the final manuscript.

Funding

Not applicable.

Availability of data and materials

Not applicable.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Appendix A: Box Plots for all results

See Fig. 3

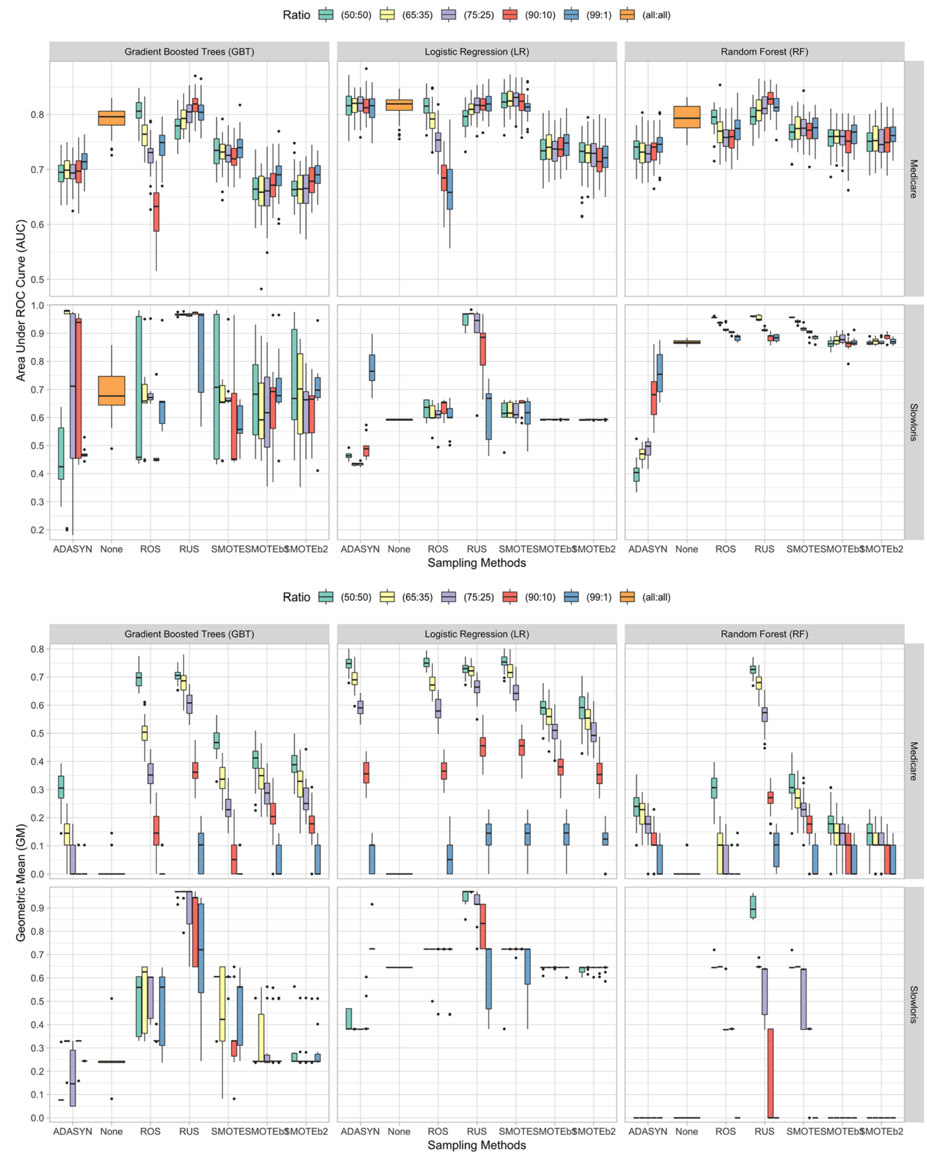

Fig. 3 Box Plot data distribution for all models

Received: 7 September 2019 Accepted: 18 November 2019
Published online: 30 November 2019

References

1. Kaisler S, Armour F, Espinosa JA, Money W. Big Data: issues and challenges moving forward. In: 2013 46th Hawaii international conference on system sciences. IEEE; 2013. p. 995–1004.
2. Datamation: Big Data Trends. https://www.datamation.com/big-data/big-data-trends.html
3. Senthilkumar S, Rai BK, Meshram AA, Gunasekaran A, Chandrakumarmangalam S. Big Data in healthcare management: a review of literature. Am J Theory Appl Bus. 2018;4:57–69.
4. Bauder RA, Khoshgoftaar TM, Hasanin T. An empirical study on class rarity in Big Data. In: 2018 17th IEEE international conference on machine learning and applications (ICMLA). IEEE; 2018. p. 785–90.
5. Leevy JL, Khoshgoftaar TM, Bauder RA, Seliya N. A survey on addressing high-class imbalance in Big Data. J Big Data. 2018;5(1):42.

6. Witten IH, Frank E, Hall MA, Pal CJ. Data mining: practical machine learning tools and techniques. Burlington: Morgan Kaufmann; 2016.
7. Olden JD, Lawler JJ, Poff NL. Machine learning methods without tears: a primer for ecologists. Q Rev Biol. 2008;83(2):171–93.
8. Galindo J, Tamayo P. Credit risk assessment using statistical and machine learning: basic methodology and risk modeling applications. Comput Econ. 2000;15(1):107–43.
9. 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, 2009. ICTAI'09. IEEE; 2009. p. 59–66.
10. Batista GE, Prati RC, Monard MC. A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explor Newslett. 2004;6(1):20–9.
11. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. Smote: synthetic minority over-sampling technique. J Artif Intell Res. 2002;16:321–57.
12. Dittman DJ, Khoshgoftaar TM, Wald R, Napolitano A. Comparison of data sampling approaches for imbalanced bioinformatics data. In: The Twenty-Seventh International FLAIRS Conference; 2014
13. Triguero I, Galar M, Merino D, Maillo J, Bustince H, Herrera F. Evolutionary undersampling for extremely imbalanced Big Data classification under apache spark. In: 2016 IEEE Congress on Evolutionary Computation (CEC). IEEE; 2016. p. 640–7.
14. The Apache Software Foundation: Apache Hadoop. http://hadoop.apache.org/
15. Venner J. Pro Hadoop. New York: Apress; 2009.
16. White T. Hadoop: the definitive guide. Newton: O'Reilly Media Inc; 2012.
17. Bauder RA, Khoshgoftaar TM, Hasanin T. Data sampling approaches with severely imbalanced Big Data for medicare fraud detection. In: 2018 IEEE 30th international conference on tools with artificial intelligence (ICTAI). IEEE; 2018. p. 137–42.
18. LEIE: Medicare provider utilization and payment data: Physician and other supplier. https://oig.hhs.gov/exclusions/index.asp
19. Tukey JW. Comparing individual means in the analysis of variance. Biometrics. 1949;5:99–114.
20. Calvert C, Khoshgoftaar TM, Kemp C, Najafabadi MM. Detection of slowloris attacks using netflow traffic. In: 24th ISSAT international conference on reliability and quality in design; 2018. p. 191–6
21. Calvert C, Khoshgoftaar TM, Kemp C, Najafabadi MM. Detecting slow http post dos attacks using netflow features. In: The thirty-second international FLAIRS conference; 2019.
22. Ali A, Shamsuddin SM, Ralescu AL. Classification with class imbalance problem: a review. Int J Adv Soft Comput Appl. 2015;7(3):176–204.
23. Fernández A, del Río S, Chawla NV, Herrera F. An insight into imbalanced Big Data classification: outcomes and challenges. Complex Intell Syst. 2017;3(2):105–20.
24. Evolutionary computation for Big Data and big learning workshop, data mining competition 2014: self-deployment track. http://cruncher.ico2s.org/bdcomp/ (2014)
25. Triguero I, del Río S, López V, Bacardit J, Benítez JM, 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.
26. Meng X, Bradley J, Yavuz B, Sparks E, Venkataraman S, Liu D, Freeman J, Tsai D, Amde M, Owen S, et al. Mllib: Machine learning in apache spark. J Mach Learn Res. 2016;17(1):1235–41.
27. Del Río S, López V, Benítez JM, Herrera F. On the use of mapreduce for imbalanced Big Data using random forest. Inf Sci. 2014;285:112–37.
28. Del Río S, Benítez JM, Herrera F. Analysis of data preprocessing increasing the oversampling ratio for extremely imbalanced Big Data classification. In: 2015 IEEE Trustcom/BigDataSE/ISPA, vol. 2, IEEE; 2015. pp. 180–5.
29. Tsai C-F, Lin W-C, Ke S-W. Big Data mining with parallel computing: a comparison of distributed and mapreduce methodologies. J Syst Softw. 2016;122:83–92.
30. Park SH, Kim SM, Ha YG. Highway traffic accident prediction using vds Big Data analysis. J Supercomput. 2016;72(7):2815–31.
31. Park SH, Ha YG. Large imbalance data classification based on mapreduce for traffic accident prediction. In: 2014 Eighth international conference on innovative mobile and internet services in Ubiquitous computing; 2014. p. 45–9.
32. Chai KE, 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.
33. CMS: Medicare provider utilization and payment data: Physician and other supplier. https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Physician-and-Other-Supplier.html
34. Liu Y-h, Zhang H-q, Yang Y-j. A dos attack situation assessment method based on qos. In: Proceedings of 2011 international conference on computer science and network technology. IEEE; 2011. p. 1041–5.
35. Yevsieieva O, Helalat SM. Analysis of the impact of the slow http dos and ddos attacks on the cloud environment. In: 2017 4th international scientific-practical conference problems of infocommunications. Science and Technology (PIC S&T). IEEE; 2017. p. 519–23.
36. Hirakaw T, Ogura K, Bista BB, Takata T. A defense method against distributed slow http dos attack. In: 2016 19th international conference on network-based information systems (NBiS)). IEEE; 2016. p. 519–23.
37. Slowloris.py. https://github.com/gkbrk/slowloris
38. Shvachko K, Kuang H, Radia S, Chansler R. The hadoop distributed file system. In: 2010 IEEE 26th symposium on mass storage systems and technologies (MSST). IEEE; 2010. p. 1–10.
39. Vavilapalli VK, Murthy AC, Douglas C, Agarwal S, Konar M, Evans R, Graves T, Lowe J, Shah H, Seth S, et al. Apache hadoop yarn: Yet another resource negotiator. In: Proceedings of the 4th annual symposium on cloud computing. ACM; 2013. p. 5.
40. Chawla NV. Data mining for imbalanced datasets: an overview. Data mining and knowledge discovery handbook, ISBN 978-0-387-09822-7. New York: Springer Science+ Business Media, LLC; 2010. p. 875.

41. Han H, Wang W-Y, Mao B-H. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In: International conference on intelligent computing. Springer; 2005. p. 878–87.
42. He H, Bai Y, Garcia EA, Li S. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence). IEEE; 2008. p. 1322–1.
43. Lemaître G, Nogueira F, Aridas CK. Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. J Mach Learn Res. 2017;18(17):1–5.
44. Le Cessie S, Van Houwelingen JC. Ridge estimators in logistic regression. J R Stat Soc. 1992;41(1):191–201.
45. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32.
46. Natekin A, Knoll A. Gradient boosting machines, a tutorial. Front Neurorobot. 2013;7:21.
47. Huang J, Ling CX. Using auc and accuracy in evaluating learning algorithms. IEEE Trans Knowl Data Eng. 2005;17(3):299–310.
48. Hanley JA, McNeil BJ. The meaning and use of the area under a receiver operating characteristic (roc) curve. Radiology. 1982;143(1):29–36.
49. Iversen GR, Wildt AR, Norpoth H, Norpoth HP. Analysis of variance. New York: Sage; 1987.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.