Journal of Big Data

RESEARCH **Open Access**

# Investigating rarity in web attacks with ensemble learners

Richard Zuech*, John Hancock and Taghi M. Khoshgoftaar

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

**Abstract**

Class rarity is a frequent challenge in cybersecurity. Rarity occurs when the positive (attack) class only has a small number of instances for machine learning classifiers to train upon, thus making it difficult for the classifiers to discriminate and learn from the positive class. To investigate rarity, we examine three individual web attacks in big data from the CSE-CIC-IDS2018 dataset: "Brute Force-Web", "Brute Force-XSS", and "SQL Injection". These three individual web attacks are also severely imbalanced, and so we evaluate whether random undersampling (RUS) treatments can improve the classification performance for these three individual web attacks. The following eight different levels of RUS ratios are evaluated: no sampling, 999:1, 99:1, 95:5, 9:1, 3:1, 65:35, and 1:1. For measuring classification performance, Area Under the Receiver Operating Characteristic Curve (AUC) metrics are obtained for the following seven different classifiers: Random Forest (RF), CatBoost (CB), LightGBM (LGB), XGBoost (XGB), Decision Tree (DT), Naive Bayes (NB), and Logistic Regression (LR) (with the first four learners being ensemble learners and for comparison, the last three being single learners). We find that applying random undersampling does improve overall classification performance with the AUC metric in a statistically significant manner. Ensemble learners achieve the top AUC scores after massive undersampling is applied, but the ensemble learners break down and have poor performance (worse than NB and DT) when no sampling is applied to our unique and harsh experimental conditions of severe class imbalance and rarity.

**Keywords:** Rarity, CSE-CIC-IDS2018, Intrusion detection, Web attacks, Class imbalance, Random undersampling, Big data, Ensemble learners

## Introduction

Cybersecurity is an important consideration for the modern Internet era, with consumers spending over $600 billion on e-commerce sales during 2019 in the United States [1]. Security practitioners struggle to properly defend this increasingly important cyberspace in a constant arms race against criminals and other adversaries. When employing security analytics [2–4], one important aspect that defenders confront is the issue of class imbalance.

Class imbalance occurs when one class label is disproportionately represented as compared to another class label. For example, in cybersecurity, it is not uncommon for a cyberattack to be lost in a sea of normal instances similar to the proverbial "needle in a haystack" analogy. Amit et al. [5] at Palo Alto Networks and Shodan, state that in cybersecurity "imbalance ratios of 1 to 10,000 are common." We agree with their assessment that very high imbalance ratios are common in cybersecurity, which is a motivation for this study to explore sampling ratios in cybersecurity web attacks.

Class rarity is an extreme case of class imbalance, and rarity is not uncommon in cybersecurity especially among more stealthy or sophisticated attacks [6]. Throughout this document, the term rarity will always refer to class rarity. Rarity occurs in machine learning when the Positive Class Count (PCC) has less than a few hundred instances [7], as compared to many more negative instances. For example, 10,000,000 total instances with an imbalance level of 1% from the positive class would yield a PCC of 100,000 which is typically enough positive class instances for machine learning classifiers to discriminate class patterns (and this example would only be highly imbalanced). On the other hand, 1,000 total instances with that same imbalance level of 1% would only provide a PCC of 10, and this would constitute rarity as machine learning classifiers will generally struggle with such few instances from the positive class [8]. For the purposes of our experiment, we will consider a PCC of less than 300 instances to constitute rarity.

To evaluate web attacks, we utilize the CSE-CIC-IDS2018 dataset which was created by Sharafaldin et al. [9] at the Canadian Institute for Cybersecurity. CSE-CIC-IDS2018 is a more recent intrusion detection dataset than the popular CIC-IDS2017 dataset [10], which was also created by Sharafaldin et al. The CSE-CIC-IDS2018 dataset includes over 16 million instances which includes normal instances, as well as the following family of attacks: web attack, Denial of Service (DoS), Distributed Denial of Service (DDoS), brute force, infiltration, and botnet. For additional details on the CSE-CIC-IDS2018 dataset [11], please refer to [12].

The CSE-CIC-IDS2018 dataset is big data, as it contains over 16 million instances. While big data has not been formally defined in terms of the number of instances, one study [13] considers only 100,000 instances to be big data. Other studies [14, 15] have considered 1,000,000 instances to be big data. Since CSE-CIC-IDS2018 is more than 1,000,000 instances, we consider it to be big data as well.

For illustrative purposes, Table 1 contains the breakdown for the entire CSE-CIC-IDS2018 dataset (although the entire dataset is not used in these experiments, and this table should only be used for reference purposes). In this study, we only focus on web attacks with normal traffic and discard the other attack instances (further details of creating the datasets are provided in the "Data preparation" section below).

Table 2 contains the three datasets we use for the experiments in this study, where each of these three datasets are comprised of web attacks from the following labels in CSE-CIC-IDS2018: "Brute Force-Web", "Brute Force-XSS", and "SQL Injection". The "Imbalance Classification" column in Table 2 indicates the varying levels of class imbalance and rarity we can explore within our experimental frameworks.

Authors of the CSE-CIC-IDS2018 dataset utilized the Damn Vulnerable Web App (DVWA) [16] and Selenium framework [17] for implementing their three web attacks. The "Brute Force-Web" label corresponds to brute force login attacks targeting web pages. Next, the "Brute Force-XSS" label refers to a cross-site scripting (XSS) attack [18] where attackers inject malicious client-side scripts into susceptible web pages targeting web users which view those pages. Finally, the "SQL Injection" label

**Table 1** Entire CSE-CIC-IDS2018 dataset by files/days (only web attacks and normal traffic are used in our experiments)

| Day | Normal instances | Attack instances |
|---|---|---|
| 02/14 Wed—Brute Force | 667,626 | 380,949 |
| 02/15 Thurs—DoS | 996,077 | 52,498 |
| 02/16 Fri—DoS | 446,772 | 601,802 |
| 02/20 Tues—DDoS | 7,372,557 | 576,191 |
| 02/21 Wed—DDoS | 360,833 | 687,742 |
| 02/22 Thu—Web | 1,048,213 | 362 |
| 02/23 Fri—Web | 1,048,009 | 566 |
| 02/28 Wed—Infiltration | 544,200 | 688,871 |
| 03/01 Thurs—Infiltration | 238,037 | 93,063 |
| 03/02 Fri—Bot | 762,384 | 286,191 |
| Total Records | 13,484,708 | 2,748,235 |

**Table 2** Individual attacks used in this experiment from CSE-CIC-IDS2018

| Attack type | PCC | Normal instances | Imbalance ratio | Imbalance classification |
|---|---|---|---|---|
| Brute Force Web | 611 | 13,390,234 | 21,915:1 | Severely imbalanced |
| XSS Web | 230 | 13,390,234 | 58,218:1 | Rarity |
| Sql Injection Web | 87 | 13,390,234 | 153,911:1 | Rarity |

*PCC* Positive Class Count

represents a code injection technique [19] where attackers craft special sequences of characters and submit them to web page forms in an attempt to directly query the back-end database of that website.

Through our data preparation process, we are able to evaluate web attacks from CSE-CIC-IDS2018 at a class ratio for normal to attack of: 21,915:1 for Brute Force, 58,218:1 for XSS, and 153,911:1 for SQL Injection web attacks. Our work is unique, in that existing works only evaluate class ratios as high as 2896:1 for web attacks and none of the existing works evaluate the effects of applying sampling techniques. The CSE-CIC-IDS2018 dataset is comprised of ten different days of files, and we combine all 10 days of normal traffic with the web attack instances. Other works only evaluate web attacks with 1 or 2 days of normal traffic. By combining all 10 days of normal traffic, we can obtain a higher imbalance ratio as well as have a richer backdrop of normal data as compared to other studies. We provide further details for this in the "Related work" and "Data preparation" sections.

To evaluate the effects of class imbalance, we explore eight different levels of sampling ratios with random undersampling (RUS): no sampling, 999:1, 99:1, 95:5, 9:1, 3:1, 65:35, and 1:1. We also compare the following seven different classifiers in our experiments with web attacks: Decision Tree, Random Forest, CatBoost, LightGBM, XGBoost, Naive Bayes, and Logistic Regression. To quantify classification performance, we utilize the Area Under the Receiver Operating Characteristic Curve (AUC) metric.

The uniqueness of our contribution is that no current works explore the effects of various sampling ratios with the CSE-CIC-IDS2018 dataset. None of the existing works combine all the days of normal traffic from CSE-CIC-IDS2018 to analyze individual web attacks, as we have uniquely done with our data preparation process to isolate these three individual web attacks with binary classification and imbalance ratios exceeding the highest 2,896:1 ratio from existing CSE-CIC-IDS2018 literature. Our work considers severe imbalance ratios as high as 153,911:1. Additionally, no works with CSE-CIC-IDS2018 explore the effects of class rarity as we present in this study with XSS and SQL Injection web attacks and their low Positive Class Count (PCC) as outlined in Table 2.

Our work focuses exclusively on web attacks to consider the above research issues, while other related works we surveyed with web attacks from CSE-CIC-IDS2018 were more generalized studies considering all attack types (as detailed in the "Related work" section below). For the few studies that did consider individual web attacks through multi-class classification, they had extremely poor classification results for those web attacks. Thus, we were surprised when our classification performance yielded such good results. We statistically validated classification performance improvements resulting from our sampling treatments, but our extensive data preparation process might have also helped as some other studies had data preparation mistakes and they were generally not specified very well.

The remaining sections of this paper are organized as follows. The "Related work" section studies existing literature for web attacks with CSE-CIC-IDS2018 data. In the "Data preparation" section, we describe how the datasets used in our experiments were cleaned and prepared. Then, the "Methodologies" section describes the classifiers, performance metrics, and sampling techniques applied in our experiments. The Results and Discussion provides our results and statistical analysis. Finally, the "Conclusion" section concludes the work presented in this paper.

## Related work

None of the prior four studies [20–23] for web attacks with CSE-CIC-IDS2018 provided any results for class imbalance analysis. No sampling techniques are applied to explore class imbalance issues for web attacks in CSE-CIC-IDS2018. None of these four studies combine the full normal traffic (all days) from CSE-CIC-IDS2018 with the individual web attacks for analysis, and instead they only use a single day of normal traffic when considering web attacks.

By combining all the normal traffic with the three individual web attacks, we can experiment with big data challenges as well as more severe levels of class imbalance which has not previously been done. Additionally, our data preparation framework allows us to isolate the three individual web attacks from all other attack traffic to research class imbalance with binary classification. Plus, this allows us to explore class rarity which has not previously been done with CSE-CIC-IDS2018.

Three of these four studies [20–22] utilized multi-class classification for the "Web" attacks, resulting in extremely poor classification performance for each of the three individual web attack labels ("Brute Force-Web", "Brute Force-XSS", and "SQL Injection"). In many cases, not even one instance could be correctly classified for an individual web attack. However, classification results for the aggregated web attacks in [23] are extremely high.

This performance discrepancy in literature between the three individual web attacks and those same web attacks combined (aggregated), motivated us to conduct this study. We were surprised to find our results to be so much better than the three other studies [20–22] analyzing these same three individual web attacks through multi-class classification. Our random undersampling approach definitely helped, although some of our classifiers still fared much better even when no sampling was applied which was likely due to our rigorous data preparation approach.

With the CSE-CIC-IDS2018 dataset, Basnet et al. [20] benchmark different deep learning frameworks: Keras-Tensorflow, Keras-Theano, and fast.ai using 10-fold cross validation. However, full results are only produced for fast.ai which is likely due to the computational constraints they frequently mention (where in some cases it took weeks to produce results). They achieve 99.9% accuracy for the aggregated web attacks with binary classification. However, the multi-class classification for those same three individual web attacks tell a completely different story with: 53 of 121 “Brute Force-Web” classified correctly, 17 of 45 “Brute Force-XSS” classified correctly, and 0 of 16 “SQL Injection” classified correctly.

Basnet et al. only provide classification results in terms of the Accuracy metric and confusion matrices (where only accuracy is provided for the aggregated web attacks). Their 99.9% accuracy scores for the aggregated web attacks can be deceptive when dealing with such high levels of class imbalance, as such a high accuracy can still be attained even with zero instances from the positive class correctly classified. When dealing with high levels of class imbalance, performance metrics which are more sensitive to class imbalance should be utilized. For web attacks, only two separate days of traffic from CSE-CIC-IDS2018 are evaluated with imbalance levels of 2,880:1 (binary) and 30,665:7.32:2.32:1 (multi-class) for 1 day and 1,842:1 (binary) and 19,666:6.83:2.85:1 (multi-class) for the other day. Such high imbalance levels require metrics more sensitive to class imbalance. Also, perhaps better classification performance might have been achieved by properly treating the class imbalance problem.

Basnet et al. use seven of the 10 days from CSE-CIC-IDS2018, and drop approximately 20,000 samples that contained “Infinity”, “NaN”, or missing values. Destination_Port and Protocol fields are treated as categorical, and the rest of the features as numeric. They state their cleaned datasets contain 79 features, which would include 8 fields containing all zero values. Instead, they should have filtered out these fields containing all zero values. Similarly, none of the other studies cited here state whether those 8 fields were filtered out or not (although it appears for most cases that of them did not filter out these 8 fields containing all zero values were not filtered out).

Atefinia and Ahmadi [21] propose a new “modular deep neural network model” and test it with CSE-CIC-IDS2018 data. Web attacks perform very poorly in their model with multi-class classification results of: 56 of 122 “Brute Force-Web” classified correctly, 0 of 46 “Brute Force-XSS” classified correctly, and 0 of 18 “SQL Injection” classified correctly. For two of the three web attacks, their model does not correctly classify even one instance of the test data. They only produce results with their one custom learner, and so benchmarking their approach is not easy.

Experimental specifications from Atefinia and Ahmadi are not clear. They state they use 2 days of web attack data from CSE-CIC-IDS2018, and "the train and test dataset are generated using 20:80 Stratified sampling of each subset". But even if we infer the test dataset to be 20% of the total, we still do not know how many instances they dropped during their preprocessing steps and for what reasons. Also, the class labels from the confusion matrix in their Fig. 10 do not match what they state for their legend: "for Web attacks, classes 1, 2, 3, and 4 represent Benign, Brute Force-Web, Brute Force-XSS and SQL Injection" (where "class 4" would result in the "SQL Injection" class to have 416,980 instances while the entire CSE-CIC-IDS2018 dataset only contains 87 instances with the "SQL Injection" label). Vague experimental specifications are a serious deficiency among the CSE-CIC-IDS2018 literature in general, and the ability to reproduce these experiments is a problem.

The work of Atefinia and Ahmadi is unique compared to the other three CSE-CIC-IDS2018 studies considering web attacks in that Atefinia and Ahmadi combine the two web attack days together with the attack and normal traffic for only those 2 days, whereas the other three studies consider each of these 2 days separately for the web attack data (days: Thursday 02/22/2018 and Friday 02/23/2018). The classification results with their new model are very poor for the web attacks, and they do not explore treating the class imbalance problem.

Unfortunately, Atefinia and Ahmadi do not provide any preprocessing details for how they cleaned and prepared the data other than stating they properly scaled the features and "the rows with missing values and the columns with too much missing values are also dropped". This statement is very ambiguous, especially since they could have easily listed the dropped columns, which is an important omission. And they state they remove IP addresses, but CSE-CIC-IDS2018 does not contain IP addresses in 9 of the 10 downloaded .csv files. Plus, the entire CSE-CIC-IDS2018 dataset contained very few missing values (only a total of 59 rows have missing values which is mainly due to repeated header lines). They do not state how they handle "Infinity" and "NaN" values.

Li et al. [22] create an unsupervised Auto-Encoder Intrusion Detection System (AE-IDS), which is based on an anomaly detection approach utilizing 85% of the normal instances as the training dataset with the testing dataset consisting of the remaining 15% of the normal instances plus all the attack instances. They only analyze 1 day of the available 2 days of "Web" attack traffic from CSE-CIC-IDS2018, and they evaluate the three different web attacks separately (versus aggregating the "Web" category together). The three individual web attacks perform very poorly with AE-IDS and multi-class classification results of: 147 of 362 "Brute Force-Web" classified correctly, 26 of 151 "Brute Force-XSS" classified correctly, and 6 of 53 "SQL Injection" classified correctly. Overall, less than half of the web attacks are classified correctly for each of the three different web attacks.

The confusion matrices provided by Li et al. are not correct and have major errors. When inspecting the confusion matrix from their Table 5 for "SQL Injection" (the class with the least number of instances) for their AE-IDS, we can see 6 True Positive instances but an incorrect number of 1,689 False Negative instances for SQL Injection. The entire CSE-CIC-IDS2018 dataset only contains 87 instances for the SQL Injection class, which is much less than their results of 1,689 False Negative instances for

SQL Injection. Instead, it seems their “Actual” and “Predicted” axes for their confusion matrices should be reversed which would instead yield a number of 47 False Negative instances for that SQL Injection example. All their confusion matrices have this problem where the “Actual” and “Predicted” axes seem incorrect, and should be the opposite versus what they reported in their results.

A major component of their experiment includes dividing the CSE-CIC-IDS2018 dataset into different sparse and dense matrices for separate evaluation. However, this sparse and dense matrix experimental factor introduces serious ambiguity in the results. First, their different results for each of these matrix approaches might actually be a result from purely partitioning the dataset into different datasets based upon different values of the data (they partition the dataset into a “sparse matrix dataset” when the “value of totlen FWD PKTS and totlen BWD PKTS is very small”. Instead, a better way may have been to randomly partition the dataset into sparse and dense matrices so that the underlying different data values themselves were not responsible for the different results from the two different sparse and dense matrix approaches.

The AE-IDS approach of Li et al. was only compared to one other learner called “KitNet” , where their AE-IDS results provided a better score for Recall. Recall is the metric they decided to use to compare all experiments. However, Precision should also be considered when comparing results with Recall. When dealing with such high levels of class imbalance such as with these web attacks, it is important to use metrics which are more sensitive to class imbalance.

Li et al. did provide AUC scores, but only for the more prominent portions of their experiments where the data was partitioned separately into sparse and dense matrices based upon certain field values. Unfortunately, as mentioned earlier, the different results for these different matrix approaches might be purely due to the fact that very different data values are being fed into these different matrix encoding approaches. Additionally, for their sparse matrix approaches, they never stated whether they were rounding down the “very small” values to zero which would be an additional concern to consider. They also assert their approach helps with class imbalance, but they do not provide any results or statistical validation to substantiate their brief commentary regarding class imbalance treatments.

Li et al. replace “Nan” and “Infinity” values with zero, but instead these imputed values should be very high, based upon manually inspecting the data. They mention no other data preparation steps other than normalizing the data, and further splitting the dataset into sparse matrices and dense matrices.

D’hooge et al. [23] evaluate each day of the CSE-CIC-IDS2018 dataset separately for binary classification with 12 different learners and stratified 5-fold cross validation. The F1 and AUC scores for the two different days with “Web” categories are generally very high, with some perfect F1 and AUC scores achieved with XGBoost. Other learners varied between 0.9 and 1.0 for both F1 and AUC scores, with the first day of “Web” usually having better performance than the second day of “Web”. The three other studies we evaluated all used multi-class classification for these same web attacks, but they all had extremely poor classification performance (many times with zero attack instances classified correctly).

D'hooge et al. state overfitting might have been a problem for CIC-IDS2017 in this same study, and "further analysis is required to be more conclusive about this finding". Given such extremely high classification scores, overfitting may have been a problem in their CSE-CIC-IDS2018 results as well (for example in their source code, we noticed the max_depth hyperparameter set to a value of 35 for Decision Tree and Random Forest learners).

In addition, their model validation approach is not clear. They state they utilize two-thirds of each day's data with stratified 5-fold cross validation for hyperparameter tuning. And then, they utilize "single execution testing". However, it is not clear how this single execution testing was performed and whether there is indeed a "gold standard" holdout test set.

D'hooge et al. replace "Infinity" values with "NaN" values in CSE-CIC-IDS2018, but "NaN" should not be used to replace other values. In the case of these "Infinity" values for CSE-CIC-IDS2018, imputed values should be very high, based upon manual inspection of the "Flow Bytes/s" and "Flow Packets/s" features. An even better alternative is to simply filter out those instances containing the "Infinity" values, as they comprise less than 1% of the data and very little attack instances are lost. The authors made no other mention for any other data preparations with CSE-CIC-IDS2018.

In summary, these enormous discrepancies in classification performance between aggregated web attacks and the three individual web attacks from CSE-CIC-IDS2018 motivated us to further explore and explain these differences. Additionally, we investigate severe class imbalance and rarity for the three individual web attacks in CSE-CIC-IDS2018 which has not previously been done.

## Data preparation

In this section, we describe how we prepared and cleaned the dataset files used in our experiments. Properly documenting these steps is important in being able to reproduce experiments.

We dropped the "Protocol" and "Timestamp" fields from CSE-CIC-IDS2018 during our preprocessing steps. The "Protocol" field is somewhat redundant as the "Dst Port" (Destination_Port) field mostly contains equivalent "Protocol" values for each Destination_Port value. Additionally, we dropped the "Timestamp" field as we wanted the learners not to discriminate attack predictions based on time especially with more stealthy attacks in mind. In other words, the learners should be able to discriminate attacks regardless of whether the attacks are high volume or slow and stealthy. Dropping the "Timestamp" field also allows us the convenience of combining or dividing the datasets into ways more compatible with our experimental frameworks. Additionally, a total of 59 records were dropped from CSE-CIC-IDS2018 due to header rows being repeated in certain days of the datasets. These duplicates were easily found and removed by filtering records based on a white list of valid label values.

The fourth downloaded file named "Thuesday-20-02-2018_TrafficForML_CICFlowMeter.csv" was different than the other nine files from CSE-CIC-IDS2018. This file contained four extra columns: "Flow ID", "Src IP", "Src Port", and "Dst IP". We dropped these four additional fields. Also of note is that this one particular file contained nearly half of all the records for CSE-CIC-IDS2018. This fourth file contained 7,948,748 records of the dataset's total 16,232,943 records.

Certain fields contained negative values which did not make sense and so we dropped those instances with negative values for the "Fwd_Header_Length", "Flow_Duration", and "Flow_IAT_Min" fields (with a total of 15 records dropped from CSE-CIC-IDS2018 for these fields containing negative values). Negative values in these fields were causing extreme values that can skew classifiers which are sensitive to outliers.

Eight fields contained constant values of zero for every instance. In other words, these fields did not contain any value other than zero. Before running machine learning, we filtered out the following list of fields (which all had values of zero):

1. Bwd_PSH_Flags
2. Bwd_URG_Flags
3. Fwd_Avg_Bytes_Bulk
4. Fwd_Avg_Packets_Bulk
5. Fwd_Avg_Bulk_Rate
6. Bwd_Avg_Bytes_Bulk
7. Bwd_Avg_Packets_Bulk
8. Bwd_Avg_Bulk_Rate

We also excluded the "Init_Win_bytes_forward" and "Init_Win_bytes_backward" fields because they contained negative values. These fields were excluded since about half of the total instances contained negative values for these two fields (so we would have removed a very large portion of the dataset by filtering all these instances out). Similarly, we did not use the "Flow_Duration" field as some of those values were unreasonably low with zero values.

The "Flow Bytes/s" and "Flow Packets/s" fields contained some "Infinity" and "NaN" values (with less than 0.6% of the records containing these values). We dropped these instances where either "Flow Bytes/s" or "Flow Packets/s" contained "Infinity" or "NaN" values. Upon carefully and manually inspecting the entire CSE-CIC-IDS2018 dataset for such values, there was too much uncertainty as to whether they were valid records or not. As sorted from minimum to maximum on these fields, neighboring records were very different where "Infinity" was found. Similar to Zhang et al. [24], we did attempt to impute values for these columns by taking the maximum value of the column and adding one. In the end, we abandoned this imputation approach and dropped 95,760 records from CSE-CIC-IDS2018 for records containing any "Infinity" or "NaN" values.

We also excluded the Destination_Port categorical feature which contains more than 64,000 distinct categorical values. Since Destination_Port has so many values, we determined that finding an optimal encoding technique was out of scope for this study. For each of the three web attacks in Table 2, we dropped all the other attack instances and kept all the normal instances from all 10 days in Table 1 (except for those instances which we removed as indicated earlier in this section). Each of the three final datasets for our individual web attacks ended up having roughly 13 million instances as specified in Table 2.

## Methodologies

### Classifiers

For all experiments in this study, stratified 5-fold cross validation [25] is used. Stratified [26] refers to evenly splitting each training and test fold so that each class is proportionately weighted across all folds equally. Splitting in a stratified manner is especially important when dealing with high levels of class imbalance, as randomness can inadvertently skew the results between folds [27]. To account for randomness, each stratified 5-fold cross validation was repeated 10 times. Therefore, all of our AUC results are the mean values from 50 measurements (5 folds x 10 repeats). All classifiers from this experiment are implemented with Scikit-learn [28] and respective Python modules.

- Decision Tree (DT) is a learner which builds branches of a tree by splitting on features based on a cost [29]. The algorithm will attempt to select the most important features to split branches upon, and iterate through the feature space by building leaf nodes as the tree is built. The cost function utilized to evaluate splits in the branches is called the Gini impurity [30].
- Random Forest (RF) is an ensemble of independent decision trees. Each instance is initially classified by every individual decision tree, and the instance is then finally classified by consensus among the individual trees (e.g., majority voting) [31]. Diversity among the individual decision trees can improve overall classification performance, and so bagging is introduced to each of the individual decision trees to promote diversity. Bagging (bootstrap aggregation) [32] is a technique to sample the dataset with replacement to accommodate randomness for each of the decision trees.
- CatBoost (CB) [33] is based on gradient boosting, and is essentially another ensemble of tree-based learners. It utilizes an ordered boosting algorithm [34] to overcome prediction shifting difficulties which are common in gradient boosting. CatBoost has native built-in support for categorical features.
- LightGBM (LGB), or Light Gradient Boosted Machine [35], is another learner based on Gradient Boosted Tree (GBTs) [36]. To optimize and avoid the need to scan every instance of a dataset when considering split points, LightGBM implements Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) algorithms [37]. LightGBM also offers native built-in support for categorical features.
- XGBoost (XGB) is another ensemble based on GBTs. To help determine splitting points, XGBoost utilizes a Weighted Quantile Sketch algorithm [38] to improve upon where split points should occur. Additionally, XGBoost employs a sparsity-aware algorithm to help with sparse data to determine default tree directions for missing values. Categorical features are not natively supported by XGBoost, and must be encoded outside of the learner with a technique such as One Hot Encoding (OHE) [39].
- Naive Bayes (NB) [40] is a probabilistic classifier which uses the Bayes' theorem [41] to calculate the posterior probability that an instance can be classified as belonging to a certain class. The posterior probability is calculated by multiplying

**Table 3** XGBoost classifier hyper-parameters

| Parameter / Value | Comment |
|---|---|
| objective = 'binary logistic' | Specify objective function for binary classification |
| n jobs = 8 | Take advantage of parallel processing functionality |
| n estimators = 4 | Prevent overfitting |
| n max depth = 5 | Prevent overfitting |

**Table 4** Random Forest classifier hyper-parameters

| Parameter/value | Comment |
|---|---|
| n estimators = 5 | Prevent overfitting |
| max depth = 6 | Prevent overfitting |

**Table 5** CatBoost classifier hyper-parameters

| Parameter / Value | Comment |
|---|---|
| thread count = 8 | Take advantage of parallel processing functionality |
| iterations = 4 | Prevent overfitting |
| max depth = 5 | Prevent overfitting |

**Table 6** Decision Tree classifier hyper-parameters

| Parameter / Value | Comment |
|---|---|
| max depth = 5 | Prevent overfitting |

the prior times the likelihood over the evidence. It uses a naive assumption that features are independent of each other.

- Logistic Regression (LR) [42] is similar to linear regression [43], and converts the output of a linear regression to a classification (categorical) value. This binary classification value is determined by applying a logarithmic function to the output value of the linear regression value.

Four of these learners are ensemble learners: Random Forest, CatBoost, LightGBM, and XGBoost. These particular learners are built upon ensembles of independent Decision Tree classifiers. Ensembles have been shown to perform very well versus their non-ensemble counterparts [44], and have been popular in Kaggle competitions [45]. In this study, we will highlight any trends for the ensemble-based learners (as well as any for those which are not based on the ensembles).

The hyper-parameters used to initialize the classifiers are indicated in Tables 3, 4, 5, and 6. The settings of these parameters were selected based on preliminary experimentation. Only the default hyper-parameters were used for LightGBM, Naive Bayes, and Logistic Regression, and so tables are not provided for these three classifiers.

## Performance metrics

Area Under the Receiver Operating Characteristic Curve (AUC) is a metric which measures the area under the Receiver Operator Characteristic (ROC) curve. AUC [46] measures the aggregate performance across all classification thresholds. The ROC curve [47, 48] is a plot of the True Positive Rate (TPR) along the y-axis versus the False Positive Rate (FPR) along the x-axis. The area under this ROC curve corresponds to a numeric value ranging between 0.0 to 1.0, where an AUC value equal to 1.0 would correspond to a perfect classification system. An AUC value equal to 0.5 would represent a classifier system which performs as well as a random guess similar to flipping a coin. The AUC metric is used to score how effective a classification system is in terms of comparing TPR to FPR over the total range of learner threshold values.

## Sampling techniques

Random undersampling (RUS) is a sampling technique to improve imbalance levels of the classes to the desired target by removing instances from the majority class(es). The removal of instances from the majority class is done without replacement, which means once an instance is removed from the majority class it is deleted and not replaced back into the majority class. RUS has been shown to be an effective sampling technique as compared to other techniques in [49]. Additional studies [50–52] have also employed RUS to deal with class imbalance.

Table 7 indicates the eight different sampling ratios applied in this study, where "None" represents no sampling applied. When no sampling is applied, the default class ratio between normal to web attacks is: 21,915:1 for Brute Force, 58,218:1 for XSS, and 153,911:1 for SQL Injection web attacks. In addition to these severe class imbalances, the XSS and SQL Injection web attacks exhibit rarity with a low Positive Class Count (PCC) as indicated in Table 2. These extreme imbalance and rarity conditions provide a problem statement as to whether RUS treatments can improve classification performance. In addition to these severe class imbalances, the XSS and SQL Injection web attacks exhibit rarity [53] with a low PCC as indicated in Table 2.

**Table 7** Random undersampling (RUS) sampling ratio levels applied

| RUS ratio |
|---|
| None |
| 999:1 |
| 99:1 |
| 95:5 |
| 9:1 |
| 3:1 |
| 65:35 |
| 1:1 |

## Results and discussion

This section is divided into three subsections for each of the 3 datasets we evaluated for our three different individual web attacks from CSE-CIC-IDS2018: Brute Force, XSS, and SQL Injection from Table 2. These three sections are presented by increasing levels of class imbalance with Brute Force web attacks presented first being only severely imbalanced. Next, the XSS results are presented with more imbalance and a slight degree of rarity. Finally, the SQL Injection results are presented last with the most severe form of class rarity where the Positive Class Count (PCC) is very low along with extreme class imbalance.

Each of these three subsections are broken down further into three additional subsections. The first subsection for each web attack are the results before any sampling is applied, and this section identifies the problem showing poor classification performance without any RUS class imbalance treatments. The next subsection for each web attack are the results with RUS applied. Then, each web attack is concluded with a subsection which includes statistical analysis.

### Results for Brute Force web attacks

#### *Results with no sampling—Brute Force web attacks*

In this section, we first present results obtained without the application of sampling techniques to Brute Force web attacks. No feature selection was applied for any of the results in this entire study, as we found all 66 features performed better with web attacks compared to our preliminary attempts with feature selection. Table 8 shows the results with no sampling applied. In this table, the AUC values are the mean across 50 values from each 5-fold cross validation being repeated 10 times. The "SD" prefix for AUC refers to the standard deviation across the 50 measurements previously described.

One minor issue we encountered for Logistic Regression, was an AUC value equal to 0.5 with a standard deviation of 0.0. Upon close inspection of the results with no RUS applied, LR was not able to correctly classify any of the positive instances. None of the other classifiers exhibited this same problem for Brute Force web attacks.

Based on the results of Table 8 (with no sampling applied), Naive Bayes is the top-performing classifier in terms of AUC for Brute Force web attacks. Logistic Regression performs the worst in terms of AUC. Overall, these classification performance scores

**Table 8** Results for classification of Brute Force web attacks and no sampling applied; classifiers: RF is Random Forest, CB is CatBoost, NB is Naive Bayes, LR is Logistic Regression, DT is Decision Tree, XGB is XGBoost, LGB is LightGBM; AUC stands for Area Under the Receiver Operating Characteristic Curve; SD stands for standard deviation

| Classifier | AUC | SD AUC |
|---|---|---|
| **NB** | **0.71786** | 0.02205 |
| RF | 0.60966 | 0.02465 |
| DT | 0.6081 | 0.01541 |
| CB | 0.60474 | 0.0237 |
| XGB | 0.59958 | 0.01578 |
| LGB | 0.52347 | 0.04591 |
| LR | 0.5 | 0.0 |

Bold value indicates highest score

are not very good considering an AUC score of 0.5 is equivalent to a random guess. This establishes a baseline of poor classification performance with such high class imbalance for Brute Force web attacks, and the next section explores whether applying RUS can improve upon this problem.

### *Results with sampling—Brute Force web attacks*

Table 9 in this section provides results for each classifier with various sampling ratios applied to Brute Force web attacks (and no feature selection is applied). The following seven sampling ratios are applied to each of the classifiers with random undersampling: 999:1, 99:1, 95:5, 9:1, 3:1, 65:35, and 1:1. In addition, "no sampling" is also indicated in the tables with the value "None" from the results of the previous section. Therefore, a total of eight different sampling ratios are evaluated. These seven classifiers are evaluated in the following tables for our various RUS ratios: RF, LR, XGB, CB, NB, DT, and LGB. Similar to the previous section, AUC results are the mean across 50 different measurements (5-fold cross validation repeated 10 times). The "SD" prefix refers to the standard deviation across these 50 different measurements.

In general, when we visually inspect the results of the different classifiers from Table 9, the results suggest that applying RUS does indeed improve classification performance. In some cases, the improvements from applying sampling are very substantial. For example, LightGBM improves from an AUC score of 0.52347 with no sampling applied to an AUC score of 0.94182 with a 1:1 RUS ratio applied. Overall, LightGBM achieves the highest AUC score of the seven classifiers. Although, Random Forest is a close second place with an AUC score of 0.9416 and 1:1 RUS ratio applied. All four of the ensemble learners (LGB, RF, XGB, and CB) have dramatic improvements in AUC scores as increased levels of RUS ratios are applied.

While Naive Bayes performs the best among all the classifiers with no sampling applied (from Table 8), the classification performance of Naive Bayes does not substantially improve as more RUS is applied (unlike substantial improvements seen from applying RUS to the other learners). Future work can explore both of these phenomena with Naive Bayes. When considering high levels of sampling like the RUS 1:1 ratio, Naive Bayes performs much worse than all the ensemble classifiers and Decision Tree. Overall, Logistic Regression appears to perform the worst among all the classifiers. Upon visual inspection of all the results from Table 9, it does appear that applying RUS does substantially improve upon the problem of such severe class imbalance. In the next section, we will apply statistical analysis to validate our visual interpretations of this table.

### *Statistical analysis—Brute Force web attacks*

We conduct two two-factor ANalysis Of VAriance (ANOVA) [54] tests to test the impact of the combination of learner and sampling ratio on performance in terms of AUC for Brute Force web attacks. The results of the ANOVA tests are in Table 10. The data for these tests is the same data we summarize from the prior section with the results of seven different classifiers across eight different sampling ratios. A confidence level of 99% is used for all tests.

**Table 9** Results for all seven classifiers and Brute Force web attacks across eight sampling ratios; Sampling column reports negative to positive class ratio after applying random undersampling, or "None" for case when no undersampling is applied; abbreviations are the same as in Table 8

| Classifier | Sampling | AUC | SD AUC | — | Classifier | Sampling | AUC | SD AUC |
|---|---|---|---|---|---|---|---|---|
| LGB | None | 0.52347 | 0.04591 | — | DT | None | 0.6081 | 0.01541 |
| LGB | 999:1 | 0.56732 | 0.07109 | — | DT | 999:1 | 0.71138 | 0.02993 |
| LGB | 99:1 | 0.82661 | 0.02345 | — | DT | 99:1 | 0.77387 | 0.0369 |
| LGB | 95:5 | 0.86462 | 0.02088 | — | DT | 95:5 | 0.81454 | 0.01794 |
| LGB | 9:1 | 0.89189 | 0.02004 | — | DT | 9:1 | 0.81491 | 0.02548 |
| LGB | 3:1 | 0.92633 | 0.01321 | — | DT | 3:1 | 0.92268 | 0.02977 |
| LGB | 65:35 | 0.9345 | 0.01298 | — | DT | **65:35** | **0.92906** | 0.01979 |
| **LGB** | **1:1** | **0.94182** | 0.01042 | — | DT | 1:1 | 0.92573 | 0.0118 |
| RF | None | 0.60966 | 0.02465 | — | NB | None | 0.71786 | 0.02205 |
| RF | 999:1 | 0.7004 | 0.04502 | — | NB | 999:1 | 0.71812 | 0.021 |
| RF | 99:1 | 0.79814 | 0.03736 | — | NB | 99:1 | 0.72041 | 0.02533 |
| RF | 95:5 | 0.81556 | 0.01969 | — | NB | 95:5 | 0.71591 | 0.01397 |
| RF | 9:1 | 0.82477 | 0.02217 | — | NB | 9:1 | 0.71825 | 0.01794 |
| RF | 3:1 | 0.91671 | 0.02861 | — | NB | 3:1 | 0.72329 | 0.02742 |
| RF | 65:35 | 0.94079 | 0.00935 | — | NB | 65:35 | 0.72421 | 0.02528 |
| RF | **1:1** | **0.9416** | 0.00757 | — | NB | **1:1** | **0.72745** | 0.02637 |
| XGB | None | 0.59958 | 0.01578 | — | LR | None | 0.5 | 0.0 |
| XGB | 999:1 | 0.66607 | 0.02099 | — | LR | 999:1 | 0.5 | 0.0 |
| XGB | 99:1 | 0.81074 | 0.02 | — | LR | 99:1 | 0.60291 | 0.01992 |
| XGB | 95:5 | 0.81301 | 0.02315 | — | LR | 95:5 | 0.60286 | 0.01719 |
| XGB | 9:1 | 0.81341 | 0.01607 | — | LR | 9:1 | 0.60202 | 0.01278 |
| XGB | 3:1 | 0.91119 | 0.03282 | — | LR | 3:1 | 0.69736 | 0.02262 |
| XGB | **65:35** | **0.93861** | 0.0095 | — | LR | 65:35 | 0.71535 | 0.04137 |
| XGB | 1:1 | 0.93797 | 0.00782 | — | LR | **1:1** | **0.81256** | 0.01301 |
| CB | None | 0.60474 | 0.0237 | — | | | | |
| CB | 999:1 | 0.61818 | 0.02931 | — | | | | |
| CB | 99:1 | 0.80572 | 0.02413 | — | | | | |
| CB | 95:5 | 0.81255 | 0.0217 | — | | | | |
| CB | 9:1 | 0.81618 | 0.02156 | — | | | | |
| CB | 3:1 | 0.8831 | 0.03923 | — | | | | |
| CB | 65:35 | 0.92989 | 0.01351 | — | | | | |
| CB | **1:1** | **0.9311** | 0.00945 | — | | | | |

Bolded values indicate highest scores

Since the $p$-values for the ANOVA tests are 0 in all cases, we conduct Tukey's Honestly Significant Difference (HSD) [55] tests to find the optimal values for learner and sampling ratio. We conduct a total of two HSD tests: one test to determine groupings of classifiers by performance in terms of AUC, and one test to determine groupings of sampling ratios, also by performance in terms of AUC.

These ANOVA results are based on ten iterations of 5-fold cross validation for 8 sampling levels of 7 different classifiers, hence a total of $10 \times 5 \times 8 \times 7 = 2,800$ combinations to analyze.

**Table 10** ANOVA results for 2-factor test of classifier and sampling ratio with Brute Force web attacks, including their interaction; in terms of AUC

| Factor | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
|---|---|---|---|---|---|
| Sampling ratio | 7 | 27.85 | 3.98 | 6231.33 | 0.0000 |
| Classifier | 6 | 12.24 | 2.04 | 3194.76 | 0.0000 |
| Interaction | 42 | 6.61 | 0.16 | 246.61 | 0.0000 |
| Residuals | 2744 | 1.75 | 0.00 | | |

**Table 11** HSD groupings of classifiers after 2-factor ANOVA where classifier and sampling ratio are factors with Brute Force web attacks; groups are by performance in terms of AUC

| |
|---|
| Group a consists of: Random Forest |
| Group ab consists of: Decision Tree |
| Group b consists of: XGBoost, LightGBM |
| Group c consists of: CatBoost |
| Group d consists of: Naive Bayes |
| Group e consists of: Logistic Regression |

For the choice of classifier, the *p*-value is equal to zero from Table 10 and this indicates the choice of classifier is statistically significant for classification performance of detecting Brute Force web attacks in this experiment. Table 11 provides Tukey's HSD groupings of the seven different classifiers as ranked by AUC. However, we must emphasize caution about interpreting these rankings of the classifiers as they are ranked across all of the various sampling ratio levels in general. For example, Random Forest ranks best across all sampling ratio levels for Brute Force web attacks according to the HSD rankings. But, LightGBM actually achieved the top score at the specific RUS ratio of 1:1. Still, these rankings can be useful in order to gain a general sense of their robustness across a various spectrum of sampling ratios (especially when the top ranked classifier also happens to achieve the highest AUC score too).

The sampling ratio factor is also statistically significant based upon the *p*-value being equal to zero from Table 10 for Brute Force web attacks and AUC. Table 12 provides Tukey's HSD rankings for RUS ratios across all seven of the classifiers for Brute Force web attacks and AUC, and indicates a clear trend that classification performance improves as more random undersampling (RUS) is applied. This is very important to our problem statement that statistically shows that applying sampling improves AUC scores in detecting Brute Force web attacks across seven different learners in this experiment.

## Results for XSS web attacks

### *Results with no sampling—XSS web attacks*

In this section, we first present results obtained without the application of sampling techniques to XSS web attacks. No feature selection was applied for any of the results in this study, as we found all 66 features performed better with web attacks compared to our preliminary attempts with feature selection. Table 13 shows the results with no

**Table 12** HSD groupings of sampling ratios after 2-factor ANOVA where classifier and sampling ratio are factors with Brute Force web attacks; groups are by performance in terms of AUC

| |
|---|
| Group a consists of: RUS 1:1 |
| Group b consists of: RUS 65:35 |
| Group c consists of: RUS 3:1 |
| Group d consists of: RUS 9:1, RUS 95:5 |
| Group e consists of: RUS 99:1 |
| Group f consists of: RUS 999:1 |
| Group g consists of: None |

**Table 13** Results for classification of XSS web attacks and no sampling applied; abbreviations are the same as in Table 8

| Classifier | AUC | SD AUC |
|---|---|---|
| **NB** | **0.74257** | 0.04591 |
| DT | 0.73696 | 0.03051 |
| RF | 0.73478 | 0.03907 |
| CB | 0.73087 | 0.02937 |
| XGB | 0.71674 | 0.03711 |
| LGB | 0.51823 | 0.03288 |
| LR | 0.5 | 0.0 |

Bolded values indicate highest scores

sampling applied. In this table, the AUC values are the mean across 50 values from each 5-fold cross validation being repeated 10 times. The "SD" prefix for AUC refers to the standard deviation across the 50 measurements previously described.

With no undersampling applied, Logistic Regression classification resulted with an AUC value equal to 0.5 with a standard deviation of 0.0. Essentially, with no RUS applied, LR was not able to correctly classify any of the positive instances. None of the other classifiers exhibited this same problem for XSS web attacks (and the Brute Force web attacks had this very same issue with LR and no sampling applied).

Based on the results of Table 13 (with no sampling applied), Naive Bayes is the top-performing classifier in terms of AUC for XSS web attacks. Logistic Regression performs the worst in terms of AUC. These AUC scores are not very good with such severe class imbalance and a small degree of rarity. This establishes our problem statement as to whether applying sampling can improve upon the poor classification performance. In the next section, we present results of eight different RUS ratios to see whether applying sampling can improve in detecting XSS web attacks with AUC.

### *Results with sampling—XSS web attacks*

Table 14 in this section provides results for each classifier with various sampling ratios applied to XSS web attacks (and no feature selection is applied). The following seven sampling ratios are applied to each of the classifiers with random undersampling: 999:1, 99:1, 95:5, 9:1, 3:1, 65:35, and 1:1. In addition, "no sampling" is also indicated in the tables with the value "None" from the results of the previous section. Therefore, a total of eight different sampling ratios are evaluated. These seven classifiers are evaluated in the following tables for our various RUS ratios: RF, LR, XGB, CB, NB, DT, and LGB. Similar to the previous section, AUC results are the mean across 50 different measurements (5-fold cross validation repeated 10 times). The "SD" prefix refers to the standard deviation across these 50 different measurements.

In general, when we visually inspect the results of the different classifiers from Table 14, the results suggest that applying RUS does indeed improve classification performance. Random Forest achieves the top AUC score of 0.9524 at a RUS ratio of 65:35. All three of the other ensemble classifiers (LGB, XGB, and CB) and Decision Tree compete closely with the top score, and they all achieve their best AUC scores at a RUS ratio of 65:35 as well.

Logistic Regression appears to perform the worst overall across all the sampling ratios for detecting XSS web attacks. Again, Naive Bayes performs the best among all the classifiers with no sampling applied (from Table 13), but the classification performance of Naive Bayes does not improve as more RUS is applied (unlike improvements seen from applying RUS to the other learners). All the classifiers besides Native Bayes have a substantial improvement in AUC scores as more sampling is applied (based upon visual inspection of the table).

Overall, applying more RUS does substantially improve classification performance until the 65:35 RUS ratio is reached at which point the 1:1 RUS ratio seems to perform similar or a little worse than the 65:35 ratio. This is important as applying RUS does improve AUC scores for XSS web attacks with such severe class imbalance. In the next section, we employ statistical analysis to validate the visual interpretation of our results.

### *Statistical analysis—XSS web attacks*

We conduct two two-factor ANOVA tests to test the impact of the combination of learner and sampling ratio on performance in terms of AUC for XSS web attacks. The results of the ANOVA tests are in Table 15. The data for these tests is the same data we summarize from the prior section with the results of seven different classifiers across eight different sampling ratios. A confidence level of 99% is used for all tests.

Since the $p$-values for the ANOVA tests are 0 in all cases, we conduct Tukey's HSD tests to find the optimal values for learner and sampling ratio. We conduct a total of two HSD tests: one test to determine groupings of classifiers by performance in terms of AUC, and one test to determine groupings of sampling ratios, also by performance in terms of AUC.

These ANOVA results are based on ten iterations of 5-fold cross validation for 8 sampling levels of 7 different classifiers, hence a total of $10 \times 5 \times 8 \times 7 = 2,800$ combinations to analyze.

For the choice of classifier, the $p$-value is equal to zero from Table 15 and this indicates the choice of classifier is statistically significant for classification performance of detecting XSS web attacks in this experiment. Table 16 provides Tukey's HSD groupings of the seven different classifiers as ranked by AUC. Again, we must emphasize caution about interpreting these rankings of the classifiers as they are ranked across all of the various sampling ratio levels in general. Both Random Forest and Decision Tree rank the best across all sampling ratio levels for XSS web attacks according to

**Table 14** Results for all seven classifiers and XSS web attacks across eight sampling ratios; Sampling column reports negative to positive class ratio after applying random undersampling, or "None" for case when no undersampling is applied; abbreviations are the same as in Table 8

| Classifier | Sampling | AUC | SD AUC | — | Classifier | Sampling | AUC | SD AUC |
|---|---|---|---|---|---|---|---|---|
| LGB | None | 0.51823 | 0.03288 | — | DT | None | 0.73696 | 0.03051 |
| LGB | 999:1 | 0.67307 | 0.11519 | — | DT | 999:1 | 0.81954 | 0.0312 |
| LGB | 99:1 | 0.87713 | 0.02719 | — | DT | 99:1 | 0.82293 | 0.03471 |
| LGB | 95:5 | 0.90041 | 0.02433 | — | DT | 95:5 | 0.85849 | 0.04602 |
| LGB | 9:1 | 0.92192 | 0.02306 | — | DT | 9:1 | 0.90477 | 0.04835 |
| LGB | 3:1 | 0.94149 | 0.01856 | — | DT | 3:1 | 0.93976 | 0.03266 |
| LGB | **65:35** | **0.9473** | 0.01831 | — | DT | **65:35** | **0.94484** | 0.0168 |
| LGB | 1:1 | 0.94449 | 0.01905 | — | DT | 1:1 | 0.93476 | 0.01734 |
| RF | None | 0.73478 | 0.03907 | — | NB | None | 0.74257 | 0.04591 |
| RF | 999:1 | 0.74152 | 0.03453 | — | NB | 999:1 | 0.74326 | 0.04973 |
| RF | 99:1 | 0.82985 | 0.0411 | — | NB | 99:1 | 0.74306 | 0.04848 |
| RF | 95:5 | 0.88201 | 0.0343 | — | NB | 95:5 | 0.74327 | 0.04545 |
| RF | 9:1 | 0.91671 | 0.02994 | — | NB | 9:1 | 0.74286 | 0.04941 |
| RF | 3:1 | 0.94445 | 0.02233 | — | NB | 3:1 | 0.74451 | 0.04826 |
| **RF** | **65:35** | **0.9524** | 0.01519 | — | NB | 65:35 | 0.74629 | 0.0448 |
| RF | 1:1 | 0.94678 | 0.01268 | — | NB | **1:1** | **0.75074** | 0.04973 |
| XGB | None | 0.71674 | 0.03711 | — | LR | None | 0.5 | 0.0 |
| XGB | 999:1 | 0.73696 | 0.03592 | — | LR | 999:1 | 0.5 | 0.0 |
| XGB | 99:1 | 0.82035 | 0.02958 | — | LR | 99:1 | 0.5 | 0.0 |
| XGB | 95:5 | 0.86605 | 0.03605 | — | LR | 95:5 | 0.49865 | 0.00056 |
| XGB | 9:1 | 0.90751 | 0.03115 | — | LR | 9:1 | 0.62693 | 0.09265 |
| XGB | 3:1 | 0.94018 | 0.02408 | — | LR | 3:1 | 0.73673 | 0.06536 |
| XGB | **65:35** | **0.94541** | 0.01506 | — | LR | **65:35** | **0.90051** | 0.01441 |
| XGB | 1:1 | 0.93096 | 0.02018 | — | LR | 1:1 | 0.87242 | 0.02487 |
| CB | None | 0.73087 | 0.02937 | — | | | | |
| CB | 999:1 | 0.73739 | 0.03659 | — | | | | |
| CB | 99:1 | 0.75932 | 0.05014 | — | | | | |
| CB | 95:5 | 0.84788 | 0.04208 | — | | | | |
| CB | 9:1 | 0.88504 | 0.03583 | — | | | | |
| CB | 3:1 | 0.9375 | 0.02523 | — | | | | |
| CB | **65:35** | **0.94357** | 0.02074 | — | | | | |
| CB | 1:1 | 0.93429 | 0.0142 | — | | | | |

Bolded values indicate highest scores

the HSD rankings. But, LGB and XGB actually have top scores a little higher than Decision Tree for the 65:35 RUS ratio. Nonetheless, these HSD rankings can still be useful when carefully employed. For example, one could select RF for detecting XSS web attacks as it had the top score across all sampling ratios and learners and was also in the top performing HSD group of classifiers across all sampling ratios (meaning its AUC performance could generalize relatively well across different RUS sampling ratios).

The sampling ratio factor is also statistically significant based upon the *p*-value being equal to zero from Table 15 for XSS web attacks and AUC. Table 17 provides Tukey's HSD rankings for RUS ratios across all seven of the classifiers for XSS web attacks and AUC, and indicates a clear trend that classification performance improves as more RUS

**Table 15** ANOVA results for 2-factor test of classifier and sampling ratio with XSS web attacks, including their interaction; in terms of AUC

| Factor | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
|---|---|---|---|---|---|
| Sampling ratio | 7 | 20.30 | 2.90 | 1944.81 | 0.0000 |
| Classifier | 6 | 17.68 | 2.95 | 1976.14 | 0.0000 |
| Interaction | 42 | 9.68 | 0.23 | 154.49 | 0.0000 |
| Residuals | 2744 | 4.09 | 0.00 | | |

**Table 16** HSD groupings of classifiers after 2-factor ANOVA where classifier and sampling ratio are factors with XSS web attacks; groups are by performance in terms of AUC

| |
|---|
| Group a consists of: Decision Tree, Random Forest |
| Group b consists of: XGBoost |
| Group c consists of: CatBoost, LightGBM |
| Group d consists of: Naive Bayes |
| Group e consists of: Logistic Regression |

is applied until the 65:35 RUS ratio. Both the 65:35 and 1:1 RUS ratios are the top performing sampling ratios, and are not statistically different from each other in terms of AUC performance across all seven classifiers. These HSD rankings indicate statistically that applying sampling does improve AUC scores for detecting XSS web attacks across seven different learners in this experiment.

## Results for SQL injection web attacks

### *Results with no sampling—SQL injection web attacks*

In this section, we first present results obtained without the application of sampling techniques to SQL Injection web attacks. No feature selection was applied for any of the results in this study, as we found all 66 features performed better with web attacks compared to our preliminary attempts with feature selection. Table 18 shows the results with no sampling applied. In this table, the AUC values are the mean across 50 values from each 5-fold cross validation being repeated 10 times. The "SD" prefix for AUC refers to the standard deviation across the 50 measurements previously described.

With no undersampling applied, three different classifiers have AUC scores less than or equal to 0.5 showing their difficulty in dealing with class rarity. LightGBM, XGBoost, and Logistic Regression all performed roughly as well as randomly guessing (as an AUC score of 0.5 is comparable to random guesses). With the more pronounced class rarity with SQL Injection attacks, some of these learners highlight their difficulties in dealing with class rarity and start to break down under a PCC of only 85 instances.

Based on the results of Table 18 (with no sampling applied), by far Naive Bayes is the top-performing classifier with an AUC score of 0.889 for SQL Injection web attacks. Interestingly, all of the ensemble learners perform very poorly with no sampling applied in detecting SQL Injection web attacks. Random Forest achieves the highest score for all of the ensembles with a paltry AUC score of 0.65645 (which is outperformed by the simplistic Decision Tree).

### *Results with sampling—SQL injection web attacks*

Table 19 in section provides results for each classifier with various sampling ratios applied to SQL Injection web attacks (and no feature selection is applied). The following seven sampling ratios are applied to each of the classifiers with random undersampling: 999:1, 99:1, 95:5, 9:1, 3:1, 65:35, and 1:1. In addition, "no sampling" is also indicated in the tables with the value "None" from the results of the previous section. Therefore, a total of eight different sampling ratios are evaluated. These seven classifiers are evaluated in the following tables for our various RUS ratios: RF, LR, XGB, CB, NB, DT, and LGB. Similar to the previous section, AUC results are the mean across 50 different measurements (5-fold cross validation repeated 10 times). The "SD" prefix refers to the standard deviation across these 50 different measurements.

In general, when we visually inspect the results of the different classifiers from Table 19, the results suggest that applying RUS does indeed improve classification performance. LightGBM achieves the top AUC score of 0.946 at the RUS ratio of 65:35. All three of the other ensemble classifiers (LGB, XGB, and CB) and Decision Tree perform a little less than the LGB top score, and 65:35 is the RUS ratio for which they all achieve their best AUC scores as well.

Logistic Regression appears to perform the worst among all the classifiers for detecting SQL Injection web attacks. Again, Naive Bayes performs the best among all the classifiers with no sampling applied (from Table 18), but its classification performance still does not improve very much as more RUS is applied (unlike improvements seen from applying RUS to the other learners). Overall, all the classifiers except Naive Bayes have dramatic improvements in classification performance as more RUS is applied (up until the 3:1 RUS ratio). Based upon visual inspection of our results, it is clear that applying sampling does substantially improve performance for such extreme class imbalance and rarity. In the next section, we employ statistical analysis to further validate our observations.

**Table 17** HSD groupings of sampling ratios after 2-factor ANOVA where classifier and sampling ratio are factors with XSS web attacks; groups are by performance in terms of AUC

| |
|---|
| Group a consists of: RUS 65:35, RUS 1:1 |
| Group b consists of: RUS 3:1 |
| Group c consists of: RUS 9:1 |
| Group d consists of: RUS 95:5 |
| Group e consists of: RUS 99:1 |
| Group f consists of: RUS 999:1 |
| Group g consists of: None |

**Table 18** Results for classification of SQL Injection web attacks and no sampling applied; abbreviations are the same as in Table 8

| Classifier | AUC | SD AUC |
|---|---|---|
| **NB** | **0.889** | 0.00622 |
| DT | 0.67703 | 0.04423 |
| RF | 0.65634 | 0.04689 |
| CB | 0.5651 | 0.04702 |
| LR | 0.5 | 0.0 |
| XGB | 0.5 | 0.0 |
| LGB | 0.49986 | 0.00049 |

Bolded values indicate highest scores

### *Statistical analysis—SQL injection web attacks*

We conduct two two-factor ANOVA tests to test the impact of the combination of learner and sampling ratio on performance in terms of AUC for SQL Injection web attacks. The results of the ANOVA tests are in Table 20. The data for these tests is the same data we summarize from the prior section with the results of seven different classifiers across eight different sampling ratios. A confidence level of 99% is used for all tests.

Since the $p$-values for the ANOVA tests are 0 in all cases, we conduct Tukey's HSD tests to find the optimal values for learner and sampling ratio. We conduct a total of two HSD tests: one test to determine groupings of classifiers by performance in terms of AUC, and one test to determine groupings of sampling ratios, also by performance in terms of AUC.

These ANOVA results are based on ten iterations of 5-fold cross validation for 8 sampling levels of 7 different classifiers, hence a total of $10 \times 5 \times 8 \times 7 = 2,800$ combinations to analyze.

For the choice of classifier, the $p$-value is equal to zero from Table 20 and this indicates the choice of classifier is statistically significant for classification performance of detecting SQL Injection web attacks in this experiment. Table 21 provides Tukey's HSD groupings of the seven different classifiers as ranked by AUC. Again, we must emphasize caution about interpreting these rankings of the classifiers as they are ranked across all of the various sampling ratio levels in general. Naive Bayes ranks the best across all sampling ratio levels for SQL Injection web attacks according to the HSD rankings. All of the classifiers except LR actually have higher scores than NB at the 65:35 or 1:1 RUS ratios. However, Naive Bayes is still surprisingly competitive against all the classifiers even at the highest 65:35 and 1:1 RUS ratios. For example, at the 1:1 RUS ratio, NB's AUC score of 0.90454 performs better than XGB's score of 0.899 (even though XGB performs better than NB at the 65:35 RUS ratio). LightGBM achieves the top AUC score of 0.946 at a RUS ratio of 65:35.

The sampling ratio factor is also statistically significant based upon the $p$-value being equal to zero from Table 20 for SQL Injection web attacks and AUC. Table 22 provides Tukey's HSD rankings for RUS ratios across all seven of the classifiers for SQL Injection web attacks and AUC, and indicates a clear trend that classification performance improves as more RUS is applied until the 3:1 RUS ratio. The 1:1 RUS ratio is the top performing sampling ratio, followed by the 65:35 and then 3:1 RUS ratios in terms of

**Table 19** Results for all seven classifiers and SQL Injection web attacks across eight sampling ratios; Sampling column reports negative to positive class ratio after applying random undersampling, or "None" for case when no undersampling is applied; abbreviations are the same as in Table 8

| Classifier | Sampling | AUC | SD AUC | — | Classifier | Sampling | AUC | SD AUC |
|---|---|---|---|---|---|---|---|---|
| LGB | None | 0.49986 | 0.00049 | — | DT | None | 0.67703 | 0.04423 |
| LGB | 999:1 | 0.56805 | 0.08605 | — | DT | 999:1 | 0.69047 | 0.04916 |
| LGB | 99:1 | 0.83408 | 0.05856 | — | DT | 99:1 | 0.79755 | 0.07772 |
| LGB | 95:5 | 0.87746 | 0.05233 | — | DT | 95:5 | 0.82941 | 0.06519 |
| LGB | 9:1 | 0.9024 | 0.0411 | — | DT | 9:1 | 0.84456 | 0.08486 |
| LGB | 3:1 | 0.9333 | 0.0379 | — | DT | 3:1 | 0.90096 | 0.05703 |
| **LGB** | **65:35** | **0.946** | 0.02762 | — | DT | **65:35** | **0.92392** | 0.03285 |
| LGB | 1:1 | 0.94017 | 0.02363 | — | DT | 1:1 | 0.91443 | 0.03074 |
| RF | None | 0.65634 | 0.04689 | — | NB | None | 0.889 | 0.00622 |
| RF | 999:1 | 0.70019 | 0.05357 | — | NB | 999:1 | 0.89069 | 0.00288 |
| RF | 99:1 | 0.75521 | 0.06054 | — | NB | 99:1 | 0.89114 | 0.00325 |
| RF | 95:5 | 0.80161 | 0.06555 | — | NB | 95:5 | 0.89224 | 0.00329 |
| RF | 9:1 | 0.86003 | 0.05757 | — | NB | 9:1 | 0.89475 | 0.0043 |
| RF | 3:1 | 0.93073 | 0.03798 | — | NB | 3:1 | 0.89921 | 0.00573 |
| RF | **65:35** | **0.93146** | 0.03533 | — | NB | 65:35 | 0.90168 | 0.00404 |
| RF | 1:1 | 0.92295 | 0.02953 | — | NB | **1:1** | **0.90454** | 0.00526 |
| XGB | None | 0.5 | 0.0 | — | LR | None | 0.5 | 0.0 |
| XGB | 999:1 | 0.68787 | 0.0593 | — | LR | 999:1 | 0.5 | 0.0 |
| XGB | 99:1 | 0.6929 | 0.04802 | — | LR | 99:1 | 0.5 | 0.0 |
| XGB | 95:5 | 0.80654 | 0.05411 | — | LR | 95:5 | 0.5 | 0.0 |
| XGB | 9:1 | 0.85162 | 0.07574 | — | LR | 9:1 | 0.5 | 0.0 |
| XGB | **3:1** | **0.92985** | 0.02822 | — | LR | 3:1 | 0.71601 | 0.06113 |
| XGB | 65:35 | 0.92172 | 0.04479 | — | LR | 65:35 | 0.73246 | 0.05251 |
| XGB | 1:1 | 0.899 | 0.0279 | — | LR | **1:1** | **0.81729** | 0.02758 |
| CB | None | 0.5651 | 0.04702 | — | | | | |
| CB | 999:1 | 0.68882 | 0.0493 | — | | | | |
| CB | 99:1 | 0.69655 | 0.05597 | — | | | | |
| CB | 95:5 | 0.72241 | 0.05719 | — | | | | |
| CB | 9:1 | 0.7693 | 0.05974 | — | | | | |
| CB | 3:1 | 0.89057 | 0.0555 | — | | | | |
| CB | **65:35** | **0.92163** | 0.03531 | — | | | | |
| CB | 1:1 | 0.90485 | 0.03245 | — | | | | |

Bolded values indicate highest scores

AUC performance across all seven classifiers. These HSD rankings indicate statistically that applying sampling does improve AUC scores for detecting SQL Injection web attacks across seven different learners in this experiment.

## Conclusion

Applying random undersampling improves classification performance for detecting web attacks in big data from the CSE-CIC-IDS2018 dataset. Based on statistical analysis, the RUS ratio is a significant factor for the AUC metric in detecting all three individual web attacks in the CSE-CIC-IDS2018 dataset for: Brute Force, XSS, and

**Table 20** ANOVA results for 2-factor test of classifier and sampling ratio with SQL Injection web attacks, including their interaction; in terms of AUC

| Factor | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
|---|---|---|---|---|---|
| Sampling ratio | 7 | 28.16 | 4.02 | 1997.34 | 0.0000 |
| Classifier | 6 | 20.64 | 3.44 | 1708.01 | 0.0000 |
| Interaction | 42 | 9.99 | 0.24 | 118.14 | 0.0000 |
| Residuals | 2744 | 5.53 | 0.00 | | |

**Table 21** HSD groupings of classifiers after 2-factor ANOVA where classifier and sampling ratio are factors with SQL Injection web attacks; groups are by performance in terms of AUC

Group a consists of: Naive Bayes
Group b consists of: Decision Tree, Random Forest, LightGBM
Group c consists of: XGBoost
Group d consists of: CatBoost
Group e consists of: Logistic Regression

SQL Injection web attacks. Either the 1:1, 65:35, or 3:1 RUS ratios achieved the top AUC scores for the seven different classifiers and three different web attacks.

Classification performance problems with such severe class imbalance and rarity for these three web attacks were all significantly improved with the application of RUS. In general, classification performance was mostly very poor for all three web attacks until massive levels of undersampling were applied. Classification performance improvements from applying sampling were easily observable and statistically validated across all seven classifiers and eight levels of RUS ratios. When a RUS ratio of 1:1 is applied to an imbalance ratio of 153,911:1 like we had for SQL Injection web attacks in our experiment, training machine learning models will be much more computationally efficient with such massive amounts of undersampling and can be helpful with big data challenges.

The choice of classifier is also a significant factor in detecting the three web attacks in terms of AUC with the CSE-CIC-IDS2018 dataset. All four of the ensemble learners (Random Forest, LightGBM, XGBoost, and CatBoost) performed well at the highest RUS ratios for all three web attacks, but these ensemble learners broke down with very poor performance when challenged by the class rarity of the SQL Injection web

**Table 22** HSD groupings of sampling ratios after 2-factor ANOVA where classifier and sampling ratio are factors with SQL Injection web attacks; groups are by performance in terms of AUC

Group a consists of: RUS 1:1
Group ab consists of: RUS 65:35
Group b consists of: RUS 3:1
Group c consists of: RUS 9:1
Group d consists of: RUS 95:5
Group e consists of: RUS 99:1
Group f consists of: RUS 999:1
Group g consists of: None

attacks (and ensembles generally did not perform well across all three web attacks without the help RUS). The ensemble learners obtained the top AUC score in detecting each of the three web attacks: LGB (0.94182) for Brute Force, RF (0.9524) for XSS, and LGB (0.946) for SQL Injection web attacks.

The simplistic Decision Tree classifier was very competitive with the ensemble learners in most cases, and beat the ensemble learners when no sampling was applied (except for one case with Brute Force web attacks where RF did slightly better when no sampling was applied). Logistic Regression did not perform well overall. Our unique data preparation framework was rigorous as compared to other CSE-CIC-IDS2018 related works and was likely helpful towards classification performance, as well as providing a harsh experimental test-bed for severe class imbalance and rarity (to model cybersecurity conditions confronted in the real world).

Future work can explore Naive Bayes and its noteworthy classification performance when no sampling is applied under conditions of severe class imbalance and rarity (as well as its insensitivity to improvements when applying RUS). Other datasets could also be included for future work, as well as additional performance metrics, families of attacks, classifiers, sampling techniques, and rarity levels [56].

**Abbreviations**
AE-IDS: Auto-encoder intrusion detection system; ANOVA: ANalysis of variance; AUC: Area under the receiver operating characteristic curve; CB: CatBoost; DDoS: Distributed denial of service; DoS: Denial of service; DT: Decision tree; DVWA: Damn vulnerable web app; EFB: Exclusive feature bundling; FPR: False positive rate; GBT: Gradient boosted tree; GOSS: Gradient-based one-side sampling; HSD: Honestly significant difference; LGB: LightGBM; LR: Logistic regression; NB: Naive Bayes; OHE: One hot encoding; PCC: Positive class count; RF: Random forest; ROC: Receiver operator characteristic; RUS: Random undersampling; TPR: True positive rate; XGB: XGBoost; XSS: Cross-site scripting.

**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**
RZ prepared the manuscript and the primary literary review for this work. JH performed the statistical analyses. All authors provided feedback to TMK and helped shape the research. TMK introduced this topic to RZ, 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.

## Declarations

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

**Consent for publication**
Not applicable.

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

Received: 13 March 2021 Accepted: 3 May 2021
Published online: 20 May 2021

## References

1. Young J. US ecommerce sales grow 14.9% in 2019. https://www.digitalcommerce360.com/article/us-ecommerce-sales/. Accessed 28 Nov 2020.
2. Leevy JL, Hancock J, Zuech R, Khoshgoftaar TM. Detecting cybersecurity attacks using different network features with lightgbm and xgboost learners. In: 2020 IEEE second international conference on cognitive machine intelligence (CogMI). IEEE; 2020, pp. 190–7.
3. Wald R, Villanustre F, Khoshgoftaar TM, Zuech R, Robinson J, Muharemagic E. Using feature selection and classification to build effective and efficient firewalls. In: Proceedings of the 2014 IEEE 15th international conference on information reuse and integration (IEEE IRI 2014). IEEE; 2014, pp. 850–4.
4. Najafabadi MM, Khoshgoftaar TM, Seliya N. Evaluating feature selection methods for network intrusion detection with kyoto data. Int J Reliabil Qual Saf Eng. 2016;23(01):1650001.
5. Amit I, Matherly J, Hewlett W, Xu Z, Meshi Y, Weinberger Y. Machine learning in cyber-security-problems, challenges and data sets. arXiv preprint arXiv:1812.07858; 2018.
6. Langner R. Stuxnet: dissecting a cyberwarfare weapon. IEEE Secur Privacy. 2011;9(3):49–51.
7. 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, pp. 785–90.
8. Bauder RA, Khoshgoftaar TM. A study on rare fraud predictions with big medicare claims fraud data. Intell Data Anal. 2020;24(1):141–61.
9. Sharafaldin I, Lashkari AH, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP; 2018, pp. 108–16 .
10. CICIDS2017 Dataset. https://www.unb.ca/cic/datasets/ids-2017.html. Accessed 28 Aug 2020.
11. CSE-CIC-IDS2018 Dataset. https://www.unb.ca/cic/datasets/ids-2018.html. Accessed 28 Aug 2020.
12. Leevy JL, Khoshgoftaar TM. A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data. J Big Data. 2020;7:1–9.
13. Leevy JL, Khoshgoftaar TM, Bauder RA, Seliya N. A survey on addressing high-class imbalance in big data. J Big Data. 2018;5(1):1–30.
14. Soltysik RC, Yarnold PR. Megaoda large sample and big data time trials: separating the chaff. Optimal Data Anal. 2013;2:194–7.
15. Cao M, Chychyla R, Stewart T. Big data analytics in financial statement audits. Account Horizons. 2015;29(2):423–9.
16. Damn Vulnerable Web App GitHub website. https://github.com/digininja/DVWA. Accessed 30 Jan 2021.
17. Selenium framework website. https://www.selenium.dev/. Accessed 30 Jan 2021.
18. Hydara I, Sultan ABM, Zulzalil H, Admodisastro N. Current state of research on cross-site scripting (xss)—a systematic literature review. Inform Softw Technol. 2015;58:170–86.
19. Halfond WG, Viegas J, Orso A. et al. A classification of sql-injection attacks and countermeasures. In: Proceedings of the IEEE international symposium on secure software engineering. IEEE; 2006, vol. 1, pp. 13–5.
20. Basnet RB, Shash R, Johnson C, Walgren L, Doleck T. Towards detecting and classifying network intrusion traffic using deep learning frameworks. J Internet Serv Inf Secur. 2019;9(4):1–17.
21. Atefinia R, Ahmadi M. Network intrusion detection using multi-architectural modular deep neural network. J Supercomput. 2020;77:3571–93.
22. Li X, Chen W, Zhang Q, Wu L. Building auto-encoder intrusion detection system based on random forest feature selection. Comput Secur. 2020;95:101851.
23. D'hooge L, Wauters T, Volckaert B, De Turck F. Inter-dataset generalization strength of supervised machine learning methods for intrusion detection. J Inform Secur Appl. 2020;54:102564.
24. Zhang H, Huang L, Wu CQ, Li Z. An effective convolutional neural network based on smote and Gaussian mixture model for intrusion detection in imbalanced dataset. Comput Netw. 2020;177:107315.
25. Arlot S, Celisse A, et al. A survey of cross-validation procedures for model selection. Stat Surv. 2010;4:40–79.
26. Forman G, Scholz M. Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. Acm Sigkdd Explorations Newsletter. 2010;12(1):49–57.
27. Kohavi R. et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Ijcai, 1995; 14, 1137–45 . Montreal, Canada.
28. Scikit-learn website. https://scikit-learn.org/stable/. Accessed 30 Jan 2021.
29. Myles AJ, Feudale RN, Liu Y, Woody NA, Brown SD. An introduction to decision tree modeling. J Chemometr J Chemometr Soc. 2004;18(6):275–85.
30. Raileanu LE, Stoffel K. Theoretical comparison between the gini index and information gain criteria. Ann Math Artif Intell. 2004;41(1):77–93.
31. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32.
32. Breiman L. Bagging predictors. Mach Learn. 1996;24(2):123–40.
33. CatBoost home page. https://catboost.ai/. Accessed 28 Aug 2020.
34. Prokhorenkova L, Gusev G, Vorobev A, Dorogush AV, Gulin A. Catboost: unbiased boosting with categorical features. In: Advances in neural information processing systems; 2018, pp. 6638–48.
35. LightGBM GitHub website. https://github.com/microsoft/LightGBM. Accessed 28 Aug 2020.
36. Natekin A, Knoll A. Gradient boosting machines, a tutorial. Front Neurorob. 2013;7:21.
37. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu T-Y. Lightgbm: a highly efficient gradient boosting decision tree. In: Advances in neural information processing systems; 2017, pp. 3146–54.
38. 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.
39. Guo C, Berkhahn F. Entity embeddings of categorical variables. arXiv preprint arXiv:1604.06737; 2016.
40. Naive Bayes scikit-learn documentation. https://scikit-learn.org/stable/modules/naive_bayes.html. Accessed 28 Aug 2020.
41. Hartigan JA. Bayes theory. Berlin/Heidelberg: Springer; 2012.

42. sklearn.linear_model.LogisticRegression scikit-learn documentation. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. Accessed 28 Aug 2020.
43. Montgomery DC, Peck EA, Vining GG. Introduction to linear regression analysis, vol. 821. Hoboken: Wiley; 2012.
44. Lahmiri S, Bekiros S, Giakoumelou A, Bezzina F. Performance assessment of ensemble learning systems in financial data classification. Intell Syst Account Fin Manage. 2020;27(1):3–9.
45. Kaggle competitions website. https://www.kaggle.com/competitions. Accessed 30 Jan 2021.
46. Bradley AP. The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recogn. 1997;30(7):1145–59.
47. Bewick V, Cheek L, Ball J. Statistics review 13: receiver operating characteristic curves. Crit Care. 2004;8(6):1–5.
48. Cook NR. Use and misuse of the receiver operating characteristic curve in risk prediction. Circulation. 2007;115(7):928–35.
49. Hasanin T, Khoshgoftaar TM, Leevy JL, Bauder RA. Severely imbalanced big data challenges: investigating data sampling approaches. J Big Data. 2019;6(1):107.
50. Bauder RA, Khoshgoftaar TM. The effects of varying class distribution on learner behavior for medicare fraud detection with imbalanced big data. Health Inform Sci Syst. 2018;6(1):9.
51. Calvert CL, Khoshgoftaar TM. Impact of class distribution on the detection of slow http dos attacks using big data. J Big Data. 2019;6(1):67.
52. Hasanin T, Khoshgoftaar TM, Bauder RA. Impact of data sampling with severely imbalanced big data. In: Reuse in intelligent systems. 2020, p. 1.
53. Herland M, Bauder RA, Khoshgoftaar TM. The effects of class rarity on the evaluation of supervised healthcare fraud detection models. J Big Data. 2019;6(1):21.
54. Tabachnick BG, Fidell LS. Experimental designs using ANOVA. Belmont: Thomson/Brooks/Cole; 2007.
55. Tukey JW. Comparing individual means in the analysis of variance. Biometrics. 1949;5:99–114.
56. Hasanin T, Khoshgoftaar TM, Leevy JL, Bauder RA. Investigating class rarity in big data. J Big Data. 2020;7(1):1–17.

## Publisher's Note