Journal of Big Data

RESEARCH Open Access

Detecting web attacks using random undersampling and 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 imbalance is an important consideration for cybersecurity and machine learning. We explore classification performance in detecting web attacks in the recent CSE-CIC-IDS2018 dataset. This study considers a total of eight random undersampling (RUS) ratios: no sampling, 999:1, 99:1, 95:5, 9:1, 3:1, 65:35, and 1:1. Additionally, seven different classifiers are employed: Decision Tree (DT), Random Forest (RF), CatBoost (CB), LightGBM (LGB), XGBoost (XGB), Naive Bayes (NB), and Logistic Regression (LR). For classification performance metrics, Area Under the Receiver Operating Characteristic Curve (AUC) and Area Under the Precision-Recall Curve (AUPRC) are both utilized to answer the following three research questions. The first question asks: "Are various random undersampling ratios statistically different from each other in detecting web attacks?" The second question asks: "Are different classifiers statistically different from each other in detecting web attacks?" And, our third question asks: "Is the interaction between different classifiers and random undersampling ratios significant for detecting web attacks?" Based on our experiments, the answers to all three research questions is "Yes". To the best of our knowledge, we are the first to apply random undersampling techniques to web attacks from the CSE-CIC-IDS2018 dataset while exploring various sampling ratios.

Keywords: CSE-CIC-IDS2018, Intrusion Detection, Web Attacks, Class Imbalance, Random Undersampling, 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]. Detecting web attacks is important as hackers frequently attack web servers for information, money, or other interests. Security practitioners struggle with cyber risk [2], and improved capabilities in detecting web attacks can help mitigate such risks. To better accommodate modeling cyber risk and predicting web attacks, this study focuses on severe levels of class imbalance. When employing security analytics [3–5], defenders commonly confront 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. [6] 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.

To evaluate web attacks, we utilize the CSE-CIC-IDS2018 dataset which was created by Sharafaldin et al. [7] at the Canadian Institute for Cybersecurity. CSE-CIC-IDS2018 is a more recent intrusion detection dataset than the popular CIC-IDS2017 dataset [8], 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 [9], please refer to [10].

In this study, we only focus on web attacks with normal traffic and discard the other attack instances. Web attacks are comprised of the following labels from CSE-CIC-IDS2018: "SQL Injection", "Brute Force-Web", and "Brute Force-XSS". For illustrative purposes, Table 1 contains the breakdown for the entire CSE-CIC-IDS2018 dataset (although the focus of this current study is only on web attacks).

Through our data preparation process, we are able to evaluate web attacks from CSE-CIC-IDS2018 at a class ratio of 14,429:1 (normal instances:web attack). Our work is unique, in that existing works only evaluate class ratios as high as 2,896:1 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 ten days of normal traffic with the web attack instances. Other works only evaluate web attacks with one or two days of normal traffic. By combining all ten 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

Table 1 CSE-CIC-IDS2018 Dataset by Files (Days)

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

Bayes, and Logistic Regression. To quantify classification performance, we utilize two different metrics: Area Under the Receiver Operating Characteristic Curve (AUC) and Area Under the Precision-Recall Curve (AUPRC).

We pose the following research questions:

1. Are various random undersampling ratios statistically different from each other in detecting web attacks?
2. Are different classifiers statistically different from each other in detecting web attacks?
3. Is the interaction between different classifiers and random undersampling ratios significant for detecting web attacks?

The uniqueness of our contribution is that no current works explore the effects of various sampling ratios with the CSE-CIC-IDS2018 dataset. Additionally, no works utilize the AUPRC metric to evaluate performance with CSE-CIC-IDS2018. None of the existing works combine all the days of normal traffic from CSE-CIC-IDS2018 to analyze a single family of attack. Our work focuses exclusively on web attacks to answer the above research questions, while other related works we surveyed with web attacks from CSE-CIC-IDS2018 did not focus on these important aspects as they were more generalized studies considering all attack types (as detailed in the Related Work section below).

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 section answers our research questions and provides statistical analysis for our results. Finally, the Conclusion section concludes the work presented in this paper.

Related work

None of the prior four studies [11–14] 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 web attacks, we can experiment with higher levels of class imbalance as well as big data challenges.

Three of these four studies [11–13] 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 [14] are extremely high.

With the CSE-CIC-IDS2018 dataset, Basnet et al. [11] 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 one 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 ten 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 [12] 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 two 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 two days, whereas the other three studies consider each of these two 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. [13] 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 one day of the available two 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. [14] 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 motivate us to further explore and explain these differences. Additionally, we investigate class imbalance for 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. We dropped these 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.

Methodologies

Classifiers

For all experiments in this study, stratified 5-fold cross validation [15] is used. Stratified [16] 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 [17]. To account for randomness, each stratified 5-fold cross validation was repeated 10 times. Therefore, all of our AUC and AUPRC results are the mean values from 50 measurements (5 folds x 10 repeats). All classifiers from this experiment are implemented with Scikit-learn [18] and respective Python modules.

- Decision Tree (DT) is a learner which builds branches of a tree by splitting on features based on a cost [19]. 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 [20].
- Random Forest (RF) [21] 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) [22]. Diversity among the individual decision trees can improve overall classification performance, and so bagging is introduced to each of the individ-

ual decision trees to promote diversity. Bagging (bootstrap aggregation) [23] is a technique to sample the dataset with replacement to accommodate randomness for each of the decision trees.

- CatBoost (CB) [24] is based on gradient boosting, and is essentially another ensemble of tree-based learners. It utilizes an ordered boosting algorithm [25] 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 [26], is another learner based on Gradient Boosted Trees (GBTs) [27]. 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 Fea- ture Bundling (EFB) algorithms [28]. 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 [29] 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) [30].
- Naive Bayes (NB) [31] is a probabilistic classifier which uses the Bayes' theorem [32] to calculate the posterior probability that an instance can be classified as belonging to a certain class. The posterior probability is calculated by multiplying the prior times the likelihood over the evidence. It uses a naive assumption that features are independent of each other.
- Logistic Regression (LR) [33] is similar to linear regression [34], 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.

RF, CB, LGB, and XGB are ensemble classifiers, which are based on collections of independent decision tree classifiers. In other words, ensemble classifiers are more sophisticated in the sense that they are built upon combinations of independent classifiers. Ensembles have been shown to perform better than their non-ensemble counterparts in some cases [35]. And ensembles have been popular in kaggle competitions [36]. To compare ensemble performance, we have also included three classifiers which are not based on ensembles: DT, NB, and LR.

The hyper-parameters used to initialize the classifiers are indicated in Tables 2, 3, 4, and 5. 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 [37] measures the aggregate performance across all classification thresholds. The ROC curve [38,

Table 2 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 3 Random Forest classifier hyper-parameters

Parameter / value	Comment
n estimators = 5	Prevent overfitting
Max depth = 6	Prevent overfitting

Table 4 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 5 Decision Tree classifier hyper-parameters

Parameter / Value	Comment
Max depth = 5	Prevent overfitting

39] 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.

Area Under the Precision-Recall Curve (AUPRC) is a metric which mainly focuses on the performance of the positive class of interest. The precision-recall curve [40] plots the Positive Predictive Value (PPV) along the y-axis versus the TPR along the x-axis. The area under this precision-recall curve (AUPRC) [41] is also a value ranging between 0.0 to 1.0, but AUPRC is more sensitive to the positive class and class imbalance as compared to AUC. Accordingly, it is important to consider the level of class imbalance when interpreting an AUPRC score. Also, AUPRC does not consider true negatives as a factor for scoring performance since it is heavily weighted towards the positive class of interest.

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 [42]. Additional studies [43–46] have also employed RUS to deal with class imbalance. Table 6 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 14,429:1.

To illustrate how RUS works, we provide a simple example with the 1:1 RUS ratio. Our positive class has 928 web attack instances, and the negative class has 13,390,234 normal instances. The goal is to reduce the ratio between these two classes to a 1:1 ratio, and so the majority (negative) class will be undersampled to also contain only 928 instances. This is done by randomly eliminating normal instances from the negative class until only 928 normal instances are remaining, and then the 1:1 ratio will be achieved. Applying the 1:1 RUS ratio is a massive undersampling with such severe class imbalance, as it reduces the negative class from 13,390,234 instances to only 928 instances effectively reducing the negative class by a factor of 14,429. Please note this example is only for a simple illustration of how RUS works, and the RUS treatment is only applied to the training portion of the dataset. RUS is not applied to the test portion of the dataset).

Results and discussion

Results with no sampling

In this section, we first present results obtained without the application of sampling techniques. 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 7 shows the results with no sampling applied. In this table, the AUC and AUPRC values are the mean across 50 values from each 5-fold cross validation being repeated 10 times. The "SD" prefix for AUC and AUPRC 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. Essentially, with no RUS applied, LR was not

Table 6 RUS Sampling Ratio Levels Applied

RUS Ratio
None
999:1
99:1
95:5
9:1
3:1
65:35
1:1

Table 7 Results for classification with no sampling; classifiers: NB is Naive Bayes, RF is Random Forest, DT is Decision Tree, XGB is XGBoost, LGB is LightGBM, CB is CatBoost, LR is Logistic Regression; AUC stands for Area Under the Receiver Operating Characteristic Curve; SD stands for standard deviation reported for every metric; AUPRC stands for Area under the Precision-Recall Curve

Classifier	AUC	SD AUC	AUPRC	SD AUPRC
NB	0.72496	0.009	0.10793	0.04958
RF	0.64003	0.01826	0.37562	0.05405
DT	0.64708	0.01525	0.39216	0.03371
XGB	0.62339	0.01157	0.25387	0.02347
LGB	0.50233	0.00562	0.0023	0.00522
CB	0.62759	0.0195	0.322	0.05891
LR	0.5	0.0	0.00417	0.00069

able to correctly classify any of the positive instances. None of the other classifiers from this study exhibited this same problem.

Based on the results of Table 7 (with no sampling applied), Naive Bayes is the top-performing classifier in terms of AUC. However, in terms of AUPRC the Decision Tree classifier performs the best. Logistic Regression performs the worst in terms of AUC, and LightGBM performs the worst in terms of AUPRC.

Results with sampling

The following tables in the remainder of this section provide results for each classifier with various sampling ratios applied (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 and AUPRC 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 Tables 8, 9, 10, 11, 12, 13, and 14, the results suggest that applying RUS does indeed improve classification performance. Also, LightGBM seems to perform the best among the classifiers, while Logistic Regression appears to perform the worst. Interestingly, LightGBM performs the second worst among all the classifiers with no sampling applied (from Table 7), but it performs the best when RUS is applied. In the next section, we employ statistical analysis to more easily interpret our results as well as to more definitively answer our research questions.

Statistical analysis

In this section, we evaluate the answers to our research questions through statistical analysis. We conduct two different 2-factor ANalysis Of VAriance (ANOVA) [47] tests to determine the impact between the combination of learner and sampling ratio

Table 8 Results for Random Forest

Sampling	AUC	SD AUC	AUPRC	SD AUPRC
None	0.64003	0.01826	0.37562	0.05405
999:1	0.65047	0.0163	0.35197	0.0348
99:1	0.72736	0.02866	0.32367	0.03301
95:5	0.79801	0.02154	0.26788	0.0775
9:1	0.81982	0.02427	0.26089	0.06329
3:1	0.92439	0.02171	0.25407	0.07858
65:35	0.9413	0.0065	0.2426	0.10392
1:1	0.94345	0.00653	0.1624	0.08754

Sampling column reports negative to positive class ratio after applying random undersampling, or "None" for case when no undersampling is applied

AUC stands for Area Under the Receiver Operating Characteristic Curve, *SD* stands for standard deviation reported for every metric, *AUPRC* stands for Area under the Precision-Recall Curve

Table 9 Results for Logistic Regression

Sampling	AUC	SD AUC	AUPRC	SD AUPRC
None	0.5	0.0	0.00417	0.00069
999:1	0.5	0.0	0.00499	0.00101
99:1	0.56984	0.01155	0.00482	0.0009
95:5	0.56981	0.00937	0.00402	0.00073
9:1	0.59933	0.027	0.00341	0.00075
3:1	0.68762	0.01517	0.00271	0.0007
65:35	0.81354	0.02079	0.00178	0.0008
1:1	0.81707	0.0088	0.00133	0.00081

Sampling column reports negative to positive class ratio after applying random undersampling, or "None" for case when no undersampling is applied

AUC stands for Area Under the Receiver Operating Characteristic Curve, *SD* stands for standard deviation reported for every metric, *AUPRC* stands for Area under the Precision-Recall Curve

Table 10 Results for XGBoost

Sampling	AUC	SD AUC	AUPRC	SD AUPRC
None	0.62339	0.01157	0.25387	0.02347
999:1	0.66539	0.0153	0.33395	0.02927
99:1	0.74548	0.01752	0.31875	0.04356
95:5	0.79409	0.01724	0.27527	0.04971
9:1	0.80578	0.0167	0.25587	0.05697
3:1	0.9312	0.02359	0.23303	0.06146
65:35	0.93674	0.0074	0.21731	0.04368
1:1	0.9344	0.00813	0.2398	0.06638

Sampling column reports negative to positive class ratio after applying random undersampling, or "None" for case when no undersampling is applied

AUC stands for Area Under the Receiver Operating Characteristic Curve, *SD* stands for standard deviation reported for every metric, *AUPRC* stands for Area under the Precision-Recall Curve

on performance in terms of AUC and AUPRC. The data used for these tests is from the previous section with the results of seven different classifiers across eight different sampling ratios. A confidence level of 99% is used for all tests.

Table 11 Results for CatBoost

Sampling	AUC	SD AUC	AUPRC	SD AUPRC
None	0.62759	0.0195	0.322	0.05891
999:1	0.64623	0.02485	0.34685	0.05637
99:1	0.75901	0.03296	0.37749	0.06585
95:5	0.7931	0.01928	0.30189	0.096
9:1	0.80603	0.02243	0.17467	0.08558
3:1	0.89987	0.02741	0.14613	0.10772
65:35	0.92978	0.01256	0.13726	0.09471
1:1	0.9318	0.00712	0.08926	0.08841

Sampling column reports negative to positive class ratio after applying random undersampling, or "None" for case when no undersampling is applied

AUC stands for Area Under the Receiver Operating Characteristic Curve, *SD* stands for standard deviation reported for every metric, *AUPRC* stands for Area under the Precision-Recall Curve

Table 12 Results for Naive Bayes

Sampling	AUC	SD AUC	AUPRC	SD AUPRC
None	0.72496	0.009	0.10793	0.04958
999:1	0.72766	0.01801	0.23072	0.05382
99:1	0.72833	0.01782	0.22961	0.05635
95:5	0.72637	0.01106	0.2347	0.06242
9:1	0.72879	0.01711	0.22848	0.05356
3:1	0.73024	0.0109	0.21532	0.03268
65:35	0.72914	0.01221	0.24194	0.06997
1:1	0.73299	0.01246	0.21781	0.03926

Sampling column reports negative to positive class ratio after applying random undersampling, or "None" for case when no undersampling is applied

AUC stands for Area Under the Receiver Operating Characteristic Curve, *SD* stands for standard deviation reported for every metric, *AUPRC* stands for Area under the Precision-Recall Curve

Table 13 Results for decision tree

Sampling	AUC	SD AUC	AUPRC	SD AUPRC
None	0.64708	0.01525	0.39216	0.03371
999:1	0.66721	0.02272	0.34765	0.03572
99:1	0.74832	0.0205	0.26034	0.03611
95:5	0.76497	0.02007	0.19095	0.09203
9:1	0.79877	0.01877	0.2394	0.06643
3:1	0.93562	0.00458	0.28988	0.03425
65:35	0.92418	0.01147	0.2019	0.08249
1:1	0.9235	0.01043	0.11302	0.12839

Sampling column reports negative to positive class ratio after applying random undersampling, or "None" for case when no undersampling is applied

AUC stands for Area Under the Receiver Operating Characteristic Curve, *SD* stands for standard deviation reported for every metric, *AUPRC* stands for Area under the Precision-Recall Curve

Table 14 Results for LightGBM

Sampling	AUC	SD AUC	AUPRC	SD AUPRC
None	0.50233	0.00562	0.0023	0.00522
999:1	0.59007	0.06865	0.08966	0.07125
99:1	0.83497	0.01653	0.52058	0.03466
95:5	0.87266	0.01644	0.41269	0.09535
9:1	0.8944	0.01642	0.40133	0.08941
3:1	0.92909	0.01137	0.34962	0.09301
65:35	0.9398	0.01012	0.32749	0.08289
1:1	0.94641	0.00758	0.23169	0.09381

Sampling column reports negative to positive class ratio after applying random undersampling, or "None" for case when no undersampling is applied

AUC stands for Area Under the Receiver Operating Characteristic Curve, *SD* stands for standard deviation reported for every metric, *AUPRC* stands for Area under the Precision-Recall Curve

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

Factor	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Sampling ratio	7	28.58	4.08	10639.26	0.0000
Classifier	6	10.85	1.81	4713.38	0.0000
Interaction	42	7.19	0.17	446.08	0.0000
Residuals	2744	1.05	0.00		

The results of the ANOVA tests are shown in Table 15 for AUC results and in Table 18 for AUPRC results. Since the p-values for the ANOVA tests are 0 in all cases, we conduct Tukey's Honestly Significant Difference (HSD) [48] tests to find the optimal values for learner and sampling ratio. We conduct a total of four HSD tests: two tests to determine groupings of sampling ratios by performance in terms of AUC and AUPRC, and two tests to determine groupings of classifiers, also by performance in terms of AUC and AUPRC.

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.

Tukey's HSD groupings enable us to more easily interpret the results from Tables 8, 9, 10, 11, 12, 13, and 14 regarding the performance impact from various classifiers and sampling ratios. From the HSD groupings in Table 16, sampling ratios from RUS are ranked by AUC and we can easily ascertain that the RUS 1:1 and RUS 65:35 sampling ratios perform the best for detecting web attacks. As ranked by AUC, "None" (no sampling) performs the worse. There is a general trend across the various sampling levels to perform better as increased levels (more undersampling) is applied.

Similarly in Table 17, classifiers are ranked by AUC performance according to Tukey's HSD groupings for detecting web attacks, and LightGBM performs the best in terms of AUC. Logistic Regression performs the worst in terms of AUC, followed by

Table 16 HSD groupings of sampling ratios after 2-factor ANOVA where classifier and sampling ratio are factors; groups are by performance in terms of AUC

Group a consists of: RUS 1:1, 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

Table 17 HSD groupings of classifiers after 2-factor ANOVA where classifier and sampling ratio are factors; groups are by performance in terms of AUC

Group a consists of: LightGBM

Group b consists of: Random Forest, XGBoost

Group bc consists of: Decision Tree

Group c consists of: CatBoost

Group d consists of: Naive Bayes

Group e consists of: Logistic Regression

Table 18 ANOVA results for 2-factor test of classifier and sampling ratio, including their interaction; in terms of AUPRC

Factor	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Sampling ratio	7	4.03	0.58	138.67	0.0000
Classifier	6	23.87	3.98	958.07	0.0000
Interaction	42	16.01	0.38	91.80	0.0000
Residuals	2744	11.40	0.00		

Table 19 HSD groupings of sampling ratios after 2-factor ANOVA where classifier and sampling ratio are factors; groups are by performance in terms of AUPRC

Group a consists of: RUS 99:1

Group b consists of: RUS 999:1, RUS 95:5

Group c consists of: RUS 9:1, RUS 3:1

Group cd consists of: None

Group d consists of: RUS 65:35

Group e consists of: RUS 1:1

Naive Bayes as the second worst. Interestingly, the five tree-based learners all perform better than the two non tree-based learners: LR and NB. Also interesting is the observation that the simplistic Decision Tree performs marginally better than CatBoost, which is a more sophisticated classifier with an ensemble of trees.

Next, we continue our analysis by comparing performance in terms of AUPRC. Table 18 provides ANOVA results in terms of AUPRC between a 2-factor test of classifier and sampling ratio.

From Table 19, the HSD groupings for sampling ratios are ranked in terms of AUPRC for web attacks. The sampling ratio of RUS 99:1 performs the best with RUS 1:1 performing the worst. The rankings for sampling ratios in terms of AUPRC do not indicate a clear trend, and are different than the HSD groupings of sampling ratio as ranked by AUC from Table 16. As future work, we plan to further explore these ranking differences of sampling ratios between AUPRC and AUC by evaluating different performance metrics as was done by Calvert et al. [49].

In Table 20, the classifiers are ranked in terms of AUPRC for the web attacks in this experiment. LightGBM performs the best and Logistic Regression performs the worst. The classifier rankings between AUPRC (Table 20) and AUC (Table 17) are mainly in agreement with only minor differences. Overall, the tree-based classifiers perform better than NB and LR, which are not based on trees. Also, the ensemble-based classifiers (LGB, RF, XGB, CB) generally perform better than the DT, NB, and LR classifiers, which are not based on ensembles. Although, the simple Decision Tree did perform significantly better than the ensemble-based CatBoost. Future work with web attacks from other datasets can further evaluate whether this trend of better performance with tree-based and ensemble classifiers generally perform better.

Research questions

In this section, we provide answers to our research questions. We use the ANOVA tables from the previous statistical analysis section to provide these answers.

Research Question 1: "Are various random undersampling ratios statistically different from each other in detecting web attacks?"

Yes, the RUS ratios are statistically different from each other for detecting web attacks in terms of both AUC and AUPRC. From Tables 15 and 18 the *p*-values for sampling ratio are both equal to zero at a confidence level of 99%, and the Tukey HSD rankings from Tables 16 and 19 indicate statistical differences between the RUS ratios in detecting web attacks.

Research Question 2: "Are different classifiers statistically different from each other in detecting web attacks?"

Yes, the classifiers are statistically different from each other for detecting web attacks for both AUC and AUPRC. Table 15 for AUC and Table 18 for AUPRC both have *p*-values equal to zero at a 99% confidence level, and the Tukey HSD ranking from Tables 17 and 20 indicate statistical differences between the classifiers in detecting web attacks.

Table 20 HSD groupings of classifiers after 2-factor ANOVA where classifier and sampling ratio are factors; groups are by performance in terms of AUPRC

Group a consists of: LightGBM
Group ab consists of: Random Forest
Group bc consists of: XGBoost
Group c consists of: Decision Tree
Group d consists of: CatBoost
Group e consists of: Naive Bayes
Group f consists of: Logistic Regression

Research Question 3: "Is the interaction between different classifiers and random undersampling ratios significant for detecting web attacks?"

Yes, the interaction between different classifiers and random undersampling ratios is a significant factor for detecting web attacks. The *p*-values for interaction are equal to zero for both AUC in Table 15 and also for AUPRC in Table 18. This interaction factor indicates combining classifier and sampling ratio factors also has a statistically significant impact.

Conclusion

In this experiment, we model severe class imbalance as found in the real world for web attacks with the CSE-CIC-IDS2018 dataset and our unique data preparation framework. A severe class imbalance ratio of 14,429:1 was obtained by combining all ten days of normal traffic from CSE-CIC-IDS2018 with only the web attack labels for binary classification. Before applying any random undersampling, the classifiers from our experiment struggled with poor classification performance which showcases a problem statement for the class imbalance problem with web attacks. Classification performance generally improved as more random undersampling was applied to treat this class imbalance problem.

Based on the results and statistical analysis from Tables 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, and 20, we can conclude that the answers to all three of our research questions are yes. The random undersampling ratios are significantly different from each other for both AUC and AUPRC metrics in detecting web attacks in the CSE-CIC-IDS2018 dataset. Future work can explore using different performance metrics to gain further insight into sampling ratio performance from the different AUC and AUPRC metrics.

The classifiers are also statistically different from each other in detecting web attacks in terms of both AUC and AUPRC with the CSE-CIC-IDS2018 dataset. LightGBM performed the best among the learners, while Logistic Regression performed the worst. Our four ensemble classifiers generally performed better than their three non-ensemble counterparts, but only after random undersampling was applied. Before RUS was applied, all of our classifiers broke down with poor classification performance. The interaction between classifier and sampling ratio was also statistically significant.

Future work with the CSE-CIC-IDS2018 dataset can investigate other families of attacks, individual web attack labels (as compared to the combined web attack labels used in this study), and the effects of rarity [50]. Other datasets could also be included for future work, as well as additional performance metrics, classifiers, and sampling techniques.

Abbreviations

AE-IDS: Auto-Encoder Intrusion Detection System; ANOVA: ANalysis Of VAriance; AUC: Area Under the Receiver Operating Characteristic Curve; AUPRC: Area Under the Precision-Recall Curve; CB: CatBoost; DDoS: Distributed Denial of Service; DoS: Denial of Service; DT: Decision Tree; 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; PPV: Positive Predictive Value; RF: Random Forest; ROC: Receiver Operator Characteristic; RUS: random undersampling; TPR: True Positive Rate; XGB: XGBoost.

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: 28 January 2021 Accepted: 3 May 2021
Published online: 27 May 2021

References

1. Young J. US Ecommerce Sales Grow 14.9% in 2019. Accessed: 2020-11-28. https://www.digitalcommerce360.com/article/us-ecommerce-sales/
2. Radanliev P, De Roure D, Walton R, Van Kleek M, Montalvo RM, Santos O, Burnap P, Anthi E, et al. Artificial intelligence and machine learning in dynamic cyber risk analytics at the edge. SN Applied Sciences. 2020;2(11):1–8.
3. Leevy JL, Hancock J, Zuech R, Khoshgoftaar TM. Detecting cybersecurity attacks across different network features and learners. Journal of Big Data. 2021;8(1):1–29.
4. 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), 2014;pp. 850–854 . IEEE
5. Najafabadi MM, Khoshgoftaar TM, Seliya N. Evaluating feature selection methods for network intrusion detection with kyoto data. International Journal of Reliability, Quality and Safety Engineering. 2016;23(01):1650001.
6. 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.
7. Sharafaldin I, Lashkari AH, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP, 2018;pp. 108–116.
8. CICIDS2017 Dataset. Accessed: 2020-08-28. https://www.unb.ca/cic/datasets/ids-2017.html
9. CSE-CIC-IDS2018 Dataset. Accessed: 2020-08-28. https://www.unb.ca/cic/datasets/ids-2018.html
10. 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):1–19.
11. 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.
12. Atefinia R, Ahmadi M. Network intrusion detection using multi-architectural modular deep neural network. J Supercomput 2020;1–23.
13. 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.
14. D'hooge L, Wauters T, Volckaert B, De Turck F. Inter-dataset generalization strength of supervised machine learning methods for intrusion detection. J Inf Secur Appl. 2020;54:102564.
15. Arlot S, Celisse A, et al. A survey of cross-validation procedures for model selection. Stat Surv. 2010;4:40–79.
16. Forman G, Scholz M. Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. Acm Sigkdd Explor Newslett. 2010;12(1):49–57.
17. Kohavi R et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Ijcai. Montreal, Canada; 1995. vol. 14, p. 1137–1145
18. Scikit-learn website. https://scikit-learn.org/stable/. Accessed 30 Jan 2021.
19. Myles AJ, Feudale RN, Liu Y, Woody NA, Brown SD. An introduction to decision tree modeling. J Chemom J Chemom Soc. 2004;18(6):275–85.
20. Raileanu LE, Stoffel K. Theoretical comparison between the gini index and information gain criteria. Ann Math Artif Intell. 2004;41(1):77–93.

21. Khoshgoftaar TM, Golawala M, Van Hulse J. An empirical study of learning from imbalanced data using random forest. In: 19th IEEE international conference on tools with artificial intelligence (ICTAI 2007). IEEE; 2007. vol. 2, p. 310–317
22. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32.
23. Breiman L. Bagging predictors. Mach Learn. 1996;24(2):123–40.
24. Hancock JT, Khoshgoftaar TM. Catboost for big data: an interdisciplinary review. J Big Data. 2020;7(1):1–45.
25. 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. 6638–6648.
26. LightGBM GitHub website. https://github.com/microsoft/LightGBM. Accessed 28 Aug 2020.
27. Natekin A, Knoll A. Gradient boosting machines, a tutorial. Front Neurorobot. 2013;7:21.
28. 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. 3146–3154.
29. 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. 785–794.
30. Guo C, Berkhahn F. Entity embeddings of categorical variables. arXiv preprint arXiv:1604.06737 2016.
31. Naive Bayes scikit-learn documentation. https://scikit-learn.org/stable/modules/naive_bayes.html. Accessed 28 Aug 2020.
32. Hartigan JA. Bayes theory. Berlin/Heidelberg: Springer; 2012.
33. sklearn.linear_model.LogisticRegression scikit-learn documentation. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. Accessed 28 Aug 2020
34. Montgomery DC, Peck EA, Vining GG. Introduction to linear regression analysis, vol. 821. Hoboken, NJ: Wiley; 2012.
35. Lahmiri S, Bekiros S, Giakoumelou A, Bezzina F. Performance assessment of ensemble learning systems in financial data classification. Intell Syst Account Financ Manag. 2020;27(1):3–9.
36. Kaggle competitions website. https://www.kaggle.com/competitions. Accessed 30 Jan 2021.
37. Bradley AP. The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recognit. 1997;30(7):1145–59.
38. Bewick V, Cheek L, Ball J. Statistics review 13: receiver operating characteristic curves. Crit Care. 2004;8(6):1–5.
39. Cook NR. Use and misuse of the receiver operating characteristic curve in risk prediction. Circulation. 2007;115(7):928–35.
40. Davis J, Goadrich M. The relationship between precision-recall and roc curves. In: Proceedings of the 23rd international conference on machine learning; 2006. pp. 233–240.
41. Boyd K, Eng KH, Page CD. Area under the precision-recall curve: point estimates and confidence intervals. In: Joint European conference on machine learning and knowledge discovery in databases. Springer; 2013. p. 451–466
42. Hasanin T, Khoshgoftaar TM, Leevy JL, Bauder RA. Severely imbalanced big data challenges: investigating data sampling approaches. J Big Data. 2019;6(1):107.
43. Van Hulse J, Khoshgoftaar TM, Napolitano A. Experimental perspectives on learning from imbalanced data. In: Proceedings of the 24th international conference on machine learning; 2007. pp. 935–942.
44. Bauder RA, Khoshgoftaar TM. The effects of varying class distribution on learner behavior for medicare fraud detection with imbalanced big data. Health Inf Sci Syst. 2018;6(1):9.
45. 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.
46. Hasanin T, Khoshgoftaar TM, Bauder RA. Impact of data sampling with severely imbalanced big data. Reuse Intell Syst 2020;1.
47. Tabachnick BG, Fidell LS. Experimental designs using ANOVA. Belmont, CA: Thomson/Brooks/Cole; 2007.
48. Tukey JW. Comparing individual means in the analysis of variance. Biometrics. 1949;99–114.
49. Calvert CL, Khoshgoftaar TM. Threshold based optimization of performance metrics with severely imbalanced big security data. In: 2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI). IEEE; 2019. p. 1328–1334
50. Hasanin T, Khoshgoftaar TM, Leevy JL, Bauder RA. Investigating class rarity in big data. J Big Data. 2020;7(1):1–17.