A controversial study on Random Forest Accuracy for Attack Detection

Quentin Vacher, Philippe Owezarski

A controversial study on Random Forest Accuracy for Attack Detection

Quentin Vacher and Philippe Owezarski

LAAS-CNRS, Université de Toulouse
CNRS
Toulouse, France

Abstract. Machine learning based anomaly and attack detection is a topic under the spotlights for more than one decade. It is raising a significant research effort worldwide. The reasons of this interest lie in the ability of machine learning to introduce a large part of autonomy in the attacks detection process compared to the classical signature based approach, and then reduces the management cost of networks for NetOps, provides rapid and efficient results, and is prone to detect 0d attacks. This paper confirms this efficiency showing that using the simple Random Forest (RF) algorithms together with features selection allows detection ratio greater than 99%. However, despite a large set of attempts on the training stage for improving this detection ratio, RF never detects 100% of attacks. It especially always misses some attacks in the traffic, especially under-represented ones in the training dataset. False Negatives are certainly the most dramatic events for network security. In addition, it makes adversarial learning very easy to perform. This paper then presents a controversial study on machine learning based attack detection (using the significantly illustrative RF example), and its trustworthiness limits. This paper is then trying to break the current dogma that machine learning is THE solution for future cyber-security systems.

Keywords: Attack detection, Machine learning, Random Forest, adversarial learning, training, controversial thoughts

1 Introduction

Given the large variety of usages in the Internet, its traffic is characterized by a significant variety of its features: variety of applications, variable traffic, etc. This variability often exceeds some thresholds and disturbs the way the traffic is flowing in the network. These are called traffic anomalies. Such anomalies can have multiple forms. They can be benign due for instance to large flows transmissions, flash crowds, or even misconfigurations of some equipments. At the other side of the spectrum of anomalies are malicious anomalies as attacks and notably Denial of Service (DoS) attacks. In all cases, these anomalies require the network operators (NetOps) to trigger countermeasures or actions to mitigate them, and recovering an acceptable service for the flowing traffic. Anomaly detection at run-time is then an essential requirement nowadays, and is raising a large effort since several years.

The first approach was inspired from the security enforcement solutions as signature based detection. It consists in extracting from each encountered anomaly a signature based on traffic features and values due to the anomaly, to add them to an anomaly signatures database, that serves to detect any further anomaly of the same kind. However, this approach is very time-consuming as analysis of anomalies for producing signatures is mostly hand-made, costly, and most importantly, it does not allow the detection of unknown anomalies (called 0day or 0d anomalies).

To cope with 0d anomalies detection, current research focuses on statistical approaches. It consists in designing a statistical model on traffic features able to characterize normal traffic. Any variation of the traffic characteristics from this model then raises an alarm. Among these approaches, the ones based on machine learning (ML) are currently under the spotlights and raising the huge majority of research efforts. Their interest lies in their ability in building the traffic and anomalies models during a training phase (supervised learning)[1]. Indeed, the main problematic deals with both detecting and classifying traffic anomalies to allow NetOps to trigger appropriate countermeasures. Detection and classification are specifically two functionalities that machine learning algorithms have been designed for. ML algorithms exhibit high accuracy for detecting anomalies. The literature on the topic generally shows detection ratios higher than 95%, often reaching 98 or 99%. However, none of them reaches the perfect detection ratio of 100% [1]. And more surprisingly, none of previous paper investigated why ML algorithms do not allow a perfect detection. Papers on anomaly detection continuously propose new ML based detection algorithms built by combining existing ML techniques. These algorithms are becoming more and more complex, but do not leverage on clear and exhaustive analysis of the lacks of previously published algorithms, and are then unmotivated. They just target higher detection ratios, without learning from the mistakes of previous algorithms. In addition, it is obvious that there are biases in the assessment of all these detection algorithms, the used dataset being certainly the most important one: Datasets are mandatorily limited in terms of traffic and anomalies kinds, and they mandatorily influence the design of the detection algorithms. Therefore, proudly publishing the results of one algorithm because its detection ratio is few tens of percent, or event 1 or 2% higher than related algorithms is nonsense. The results could be inverted if the algorithms were assessed on another dataset. The only conclusion that comes from all these ML based detection papers is that ML based detection algorithms are very good, but not perfect.

That is why the work presented in this paper tries to understand how the ML algorithms work, and more specifically how training impacts the detection performance. This paper then proposes a black box based methodology on ML algorithms that adopts NetOps point of view and their need of trustworthy ML, for analyzing the way training influences the capabilities of ML algorithms to detect specific anomalies, especially targeting an explanation of the decision made by the RF algorithm. This methodology mainly focuses on adapting at each

[1] For unsupervised learning, both training and detection phases are joint.

stage the training dataset according to the anomalies that remained undetected on the previous stages. For illustration purpose, the Random Forest (RF) ML algorithm has been selected, as it is generally proved to be one of the best ranked algorithm for anomaly detection in the existing literature of this domain, and it is certainly the simplest.

The rest of this paper is as follows: section 2 presents an illustrative state of the art on existing works on the use of machine learning for anomaly detection. Section 3 describes the dataset used for this work, especially focusing on the anomalies it contains. Section 4 addresses the problem of RF hyper parameters optimization with an empirical and pragmatic approach, in order to work in the best possible conditions for anomaly detection. This is an important part of the methodology as several alternative approaches could have been chosen. The best one has then been pragmatically exhibited. Section 5 is devoted to the analysis of the impact of training on ML detection ratio. Based on these results, section 6 continues this work by trying to explain the RF decision making. Finally section 7 provides a final discussion on the impact of the results presented in the paper.

2 Related Works

Machine learning for cyber-security is raising a lot of effort since at least two decades. Historically, the first machine learning algorithm - K-mean [2] - was designed more than 50 years ago. Since then, many other machine learning algorithms were designed: DB-SCAN, Random Forest, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), to quote a few. Since two decades, they have been used as a support for detecting and classifying network anomalies and attacks, exhibiting good but still imperfect performances [3] [4] Mathematicians are still designing new machine learning algorithms, especially based on Neural Networks (NN), and that belong to the family of Deep Learning algorithms (DL). These algorithms are raising the interest of researchers and engineers in cyber-security; as soon as they appear, cyber-security scientists test them. This is the case of Long Short Term Memory (LSTM) [5] for instance.

The way scientists in cyber-security proceed with all these learning algorithm lacks motivation and methodology. Most (all?) papers in machine learning based attack detection select one machine learning algorithm without indicating why this choice is made, and evaluate its performance on a labeled dataset. This is always performed with a black box point of view of the machine learning algorithm. More recently, scientists combine several machine learning techniques. The complexity of the attack detection systems then increases, for a gain that is generally very limited: few percents of accuracy in the best cases, more often only few tens of percent, and this on lab experiments with very reduced traffic datasets that are far from representing the variety of existing traffic and attacks types [6]. But none of these attack detection algorithms is able to detect all attacks without any false alarms [1].

Then, the question that is legitimate to ask is: does it worth continuing working this way, i.e. continuing designing more and more complex attack de-

tection systems just to gain few tens of percent of accuracy compared to other systems? NetOps requirements give the answer. It is essential for them to rely on trustworthy attack detection system. For that, NetOps need to understand why the machine learning algorithm detected and classified one flow as benign or malicious. They then require explainable learning algorithms.

This paper then deals with a study of the RF algorithm trying to understand how it works and how it is making its detection decision. RF has been selected as it appears as one of the best algorithms for attack detection, with a detection accuracy often evaluated over 99% [7] [8].

3 The CIC-IDS-2017 Dataset

As introduced in section 1, this paper aims at analyzing the limits of machine learning algorithms for anomaly detection in Internet traffic, taking Random Forest as an illustrative example. This work considers the Random forest based anomaly detection tool as a blackbox, and then requires a labeled dataset to compare the RF based detection results with the exact labels of the input dataset. The CIC-IDS-2017 dataset has been selected for this purpose [9]. It nowadays appears as the new reference dataset for security tool assessment. It more or less replaces the ancient KDD'99 dataset that has been the reference for more than 20 years. This dataset has been created by the Canadian Institute for Cybersecurity with the purpose of giving researchers an open source Dataset for designing and evaluating network intrusion detection systems. The CIC-IDS-2017 dataset consists of eight .csv files. It represents a total of five days of traffic. Each gathered day contains one attack or more that were specifically generated. Table 1 indicates the list of attacks contained in the different files of the CIC-IDS-2017 dataset.

Each of the .csv files is the result of the network traffic analysis performed with CICFlowMeter, an open source tool that re-constructs bidirectional flows from the gathered packet pcap files, and extracts the specific features of these flows. In bidirectional flows, the first packet determines the forward (source to destination) and backward (destination to source) directions. Hence the statistical time-related features can be calculated separately in the forward and backward directions. CIC-IDS-2017 comes with 78 features. Some of these features are indicated in Table 2 as examples.

For using this dataset we had to merge all the files into one single .csv file. We then cleaned the dataset by taking off the rows with NaN, infinite or missing values. We also noticed that there were 8 features that had a variance of 0. We then took these features off the dataset. Finally we added a binary label column. The dataset is given with types of attack as labels, but we chose to add the option of using binary class with "1" to indicate an attack, and "0" for benign traffic.

After this preprocessing stage, the final dataset consists of 2 827 876 samples with 70 features and two label columns: a MultiClass label (which is the attack

Table 1. CIC-IDS-2017 files description

File name	Day	Attacks found
Monday-WorkingHours.csv	Monday	Benign (Normal Activity)
Tuesday-WorkingHours.csv	Tuesday	Benign, FTP-Patator, SSH-Patator
Wednesday-workingHours.csv	Wednesday	Benign, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, Heartbleed
Thursday-WorkingHours-Morning-WebAttacks.csv	Thursday	Benign, Web Attack - Brute Force, Web Attack - SQL injection, Web Attack - XSS
Thursday-WorkingHours-Afternoon-Infilteration.csv	Thursday	Benign, Infiltration
Friday-WorkingHours-Morning.csv	Friday	Benign, Bot
Friday-WorkingHours-Afternoon-PortScan.csv	Friday	Benign, PortScan
Friday-WorkingHours-Afternoon-DoS.csv	Friday	Benign, DDoS

type) and a binary label. Table 3 details the repartition of the different attack types in the dataset.

4 Random Forest Hyper-parameters optimization

4.1 RF Hyper-parameters

The goal of this paper is to analyze why ML algorithms for anomaly detection do not reach the 100% detection ratio, even when applied to well-known reference datasets with perfectly identified anomalies and attacks. The algorithms are however evaluated in their best possible configuration. Indeed, in such controlled condition it is possible to optimize the configuration parameters of the ML algorithm – here RF – for optimizing its detection performance. For that, it is possible to run the RF algorithm several times on the recorded dataset

Table 2. Examples of features from CIC-IDS-2017 dataset

Feature name	**Description of the Feature**
Init_Win_bytes_backward	The total number of bytes sent in initial window in the backward direction
Destination Port	Destination port of the flow
Bwd Packet Length Min	Minimum size of packet in backward direction
Average Packet Size	Average size of packet in the flow
Fwd Packet Length Min	Minimum size of packet in the forward direction

with different values for the configuration hyper-parameters, so as to empirically discover the ones providing the best results. This is what this section aims to exhibit. This is obviously not representative of the way ML based detection algorithms are used in operational conditions as NetOps cannot afford recording the flowing traffic and running the detection algorithms several times on it. This would forbid any real-time detection of the anomalies and attacks.

For finding out the optimal parameters when testing Random Forest on the CIC-IDS-2017 dataset, we selected the RandomForestClassifier [10] function from Python Scikit-Learn. Indeed, it appears as a widely used RF implementation according to current papers dealing with RF based anomaly detection. The hyper-parameters to be optimized are:

- n_estimators: the number of estimators, i.e. the number of decision trees
- max_depth: the maximum depth of the trees inside the RF
- max_features: the maximum number of features Random Forest is allowed to consider in individual trees when looking for the best split.

In this paper, two different evaluations will be presented: the first one deals with the classical labels of the CIC-IDS-2017 dataset that indicate the kind of anomaly/attack for each entry of the dataset files. We call it evaluation with multiclass labels (considering several classes of attacks). The second one uses the extra labels that we added to the CIC-IDS-2017 files, and that just indicates whether the entry is an attack or benign traffic. We called it evaluation with binary labels.

Each experiment will then be run in both multi classes vs. binary labels. Determining the optimal hyper-parameters will then be done for the two multiclass and binary labels approaches.

The methodology for finding out the 3 optimal RF hyper-parameters follows a 5-fold Cross-Validation approach. For each parameter, several runs with a wide range of values are performed, the other hyper-parameters being fixed to the best

Table 3. Distribution of Samples in CIC-IDS-2017

Attack Type	**Number of samples in the dataset**
BENIGN	2271320
DoS Hulk	230124
PortScan	158804
DDoS	128025
DoS GoldenEye	10293
FTP-Patator	7935
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1956
Web Attack - Brute Force	1507
Web Attack - XSS	652
Infiltration	36
Web Attack - Sql Injection	21
Heartbleed	11

values previously determined. For the scoring of each run, we use the function score() of Scikit-learn which gives the accuracy of the detection. The accuracy is defined as the number of correctly identified samples over the total number of samples.

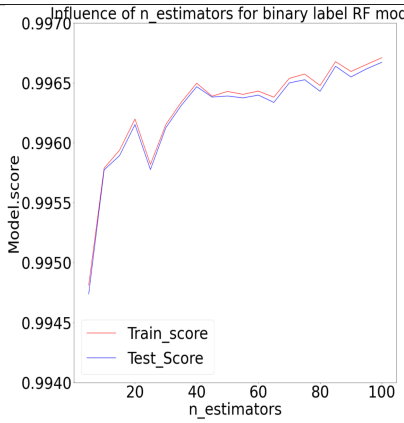
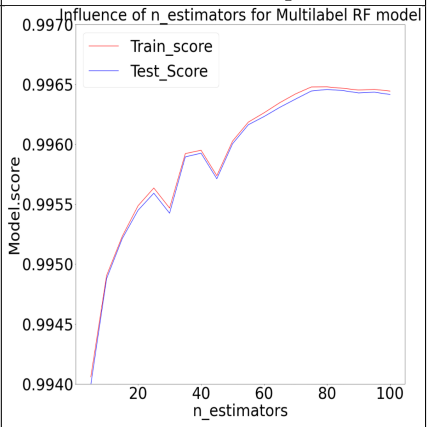
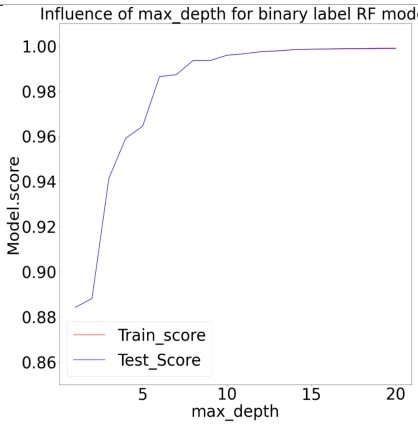
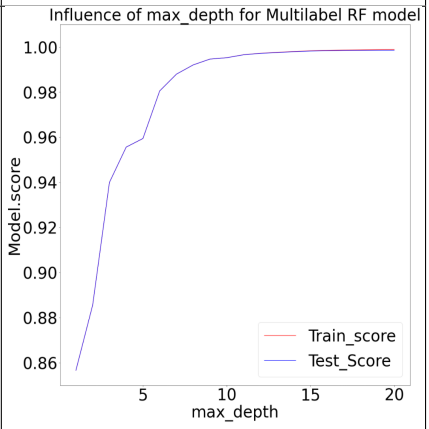
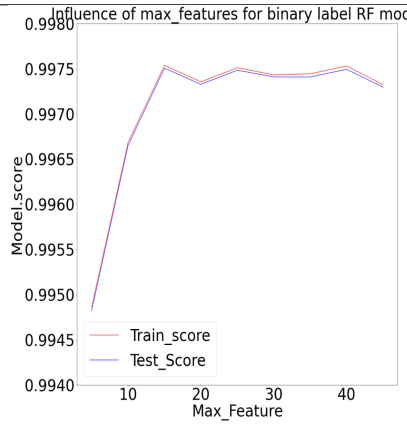
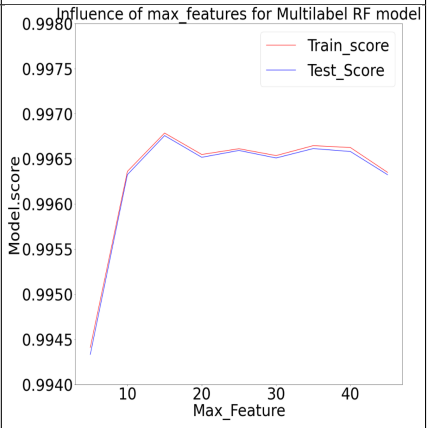
4.2 Results

The results of the determination of the 3 RF hyper-parameters optimal values are depicted in Table 4, where each cell of the table is the evaluation curve of the accuracy of the RF detection for one hyper-parameter. It especially shows the results obtained over both the train-set and the test-set, as a function of the value of the hyper-parameter being tested.

Based on these results we selected for each hyper-parameter the following values:

- n_estimators: we decided to fix the value to 80, because it is a maximum for the multiclass label, and that we need the same value for the binary label model in order to compare both binary and multiclass labels performances.
- max_depth : we chose to fix the value to 15, because for higher values, for both binary and multiclass models, the train and test scores stop increasing significantly.
- max_features : we selected 15 as the best value because it gives a maximum score for both binary and multiclass models.

Table 4. Hyper-parameters optimization results

	Binary Label Experiment	**MultiClass Label Experiment**
n_estimators		
max_depth		
max_features		

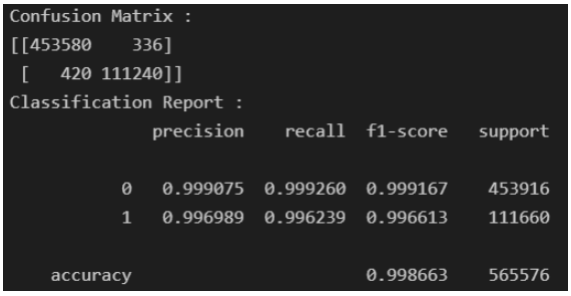


Fig. 1. Confusion matrix and classification report for binary label RF model

4.3 Analysis

Since the same resources are allocated to both binary and multiclass Classifiers, we can compare their results as achieved in the exact same situation. It clearly appears on the figures in table 4 that the use of binary labels gives better results than multiclass labels. Focusing only on binary classification results, Figure 1 shows several traditional evaluation parameters obtained with RF on the full CIC-IDS-2017 dataset. Figure 1 exhibits first the confusion matrix. The first line consists of two values: the number of True Negatives (TN) and False Positive (FP). The second line displays the number of False Negative (FN) and of True Positive (TP). The classification report shows a set of detection evaluation metrics (Precision, Recall, and F1-Score) for benign traffic class (line 0) and for attack class (line 1). The last column indicates for both benign traffic and attacks the number of sampled identified as such. Let us recall that the Precision, recall and F1-Score are computed as follows, for a class i :

$$\text{Precision}_i = \frac{\text{number of samples correctly assigned to class } i}{\text{number of samples assigned to class } i} \tag{1}$$

$$\text{Recall}_i = \frac{\text{number of samples correctly assigned to class } i}{\text{number of samples belonging to class } i} \tag{2}$$

$$\text{F1-Score} = 2 \cdot \frac{(\text{Precision} \cdot \text{Recall})}{(\text{Precision} + \text{Recall})} \tag{3}$$

The accuracy line displays first the Scikit-Learn model.score and the number of samples. The model score is defined by:

$$\text{Model.Score} = \frac{\text{number of samples correctly assigned to their class}}{\text{total number of samples}} \tag{4}$$

Finally, Figure 1 exhibits a very high accuracy score. This explains why Random Forest is so often praised for its intrusion detection capabilities.

To go deeper in the analysis of the capabilities of RF to detect attacks, Table 5 shows the detection performance for each of the 14 different attacks contained in the CIC-IDS-2017 dataset. Table 5 displays the percentage of False Negative

Table 5. FN rate per attack type for binary and multi class label RF detection

Attack Type	**Number of samples in the dataset**	**Percentage of undetected samples for the binary label model**	**Percentage of undetected samples for the multi-label model**
DoS Hulk	230124	0.09%	0.11%
PortScan	158804	0.02%	0.02%
DDoS	128025	0.08%	0.11%
DoS GoldenEye	10293	1.58%	6.25%
FTP-Patator	7935	0.06%	0.18%
SSH-Patator	5897	0.16%	0.08%
DoS slowloris	5796	1.23%	4.43%
DoS Slowhttptest	5499	1.66%	5.19%
Bot	1956	63.01%	63.26%
Web Attack - Brute Force	1507	5.29%	5.96%
Web Attack - XSS	652	4.20%	45.45%
Infiltration	36	80%	80%
Web Attack - Sql Injection	21	57.14%	100%
Heartbleed	11	0%	0%

(FN) per attack kind, together with the number of samples of each attack present in the dataset.

It clearly appears with Table 5 that despite a very low global FN rate, some attacks remain mostly undetected by RF based anomaly detection (some attacks as SQL web attacks are always missed by the RF detector). It also appears that the attacks that are not well detected are the ones with the fewest samples in the dataset. A possible explanation to this problem is precisely the fact that these attacks are underrepresented in the dataset.

5 Analyzing the impact of training on RF detection performance

Results presented in Table 5 might exhibit a link between the detection ratio of some attacks with the number of their instances in the training dataset. Indeed, a low number of samples of an attack in the training dataset might cause a poor detection ratio for this kind of attack. In this section, it is then proposed to act on the training dataset to balance highly and poorly represented attacks. Three different approaches will be used:

- Undersampling: It consists in taking samples of the most represented class off the dataset.
- OverSampling: It consists in creating new samples of a kind to increase its number of instances. For that, RandomOverSampler [13] is used. RandomOverSampler can create new samples with the features distribution of any attack type.
- Class Weighting: Contrary to the two previous approaches, this method does not change the dataset. It consists in only changing the weight the RF algorithm assigns to each class in the Gini index calculation.

5.1 Undersampling

Undersampling consists in taking some samples of traffic classes highly represented off the training-set. The dataset is split as usual: 80% go to the training and 20% are used as a test-set. Since the RF detection has to be assessed in realistic conditions, as much as possible, the balance of the test-set is not changed. Indeed, in real conditions, NIDSs face a large majority of benign traffic. In the following, the balance is defined as the number of benign samples over the number of attack samples.

To assess the RF detection performance, different balances will be considered from 4 to 0.1 by taking off progressively some samples of benign traffic, so as to increase the related ratio of attacks with few samples. For each modified train set, RF detection is applied on the unchanged test set. Figure 2 presents the obtained detection scores.

The best score is obtained with the training set balance set to 0.7. The related RF detection accuracy is 0.9989. This is a small increase of the global detection ratio with RF. The details of the detection ratio per attack kinds are presented in section 5.3. However, changing the balance that is originally of 5 in CIC-IDS-2017 to 0.7 is a huge transformation. In addition, this balance of 0.7 is not necessarily a constant value that can be applied to any dataset captured on any network and at any time. Last, with undersampling, it still remains a significant difference of the number of samples in the training dataset corresponding to the different attacks.

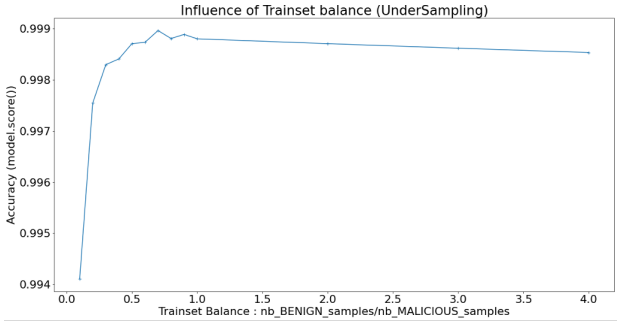


Fig. 2. Undersampling experiment results

```
Confusion Matrix :
[[453488    428]
 [   271 111389]]
Classification Report :
              precision    recall  f1-score   support

           0   0.999403  0.999057  0.999230    453916
           1   0.996172  0.997573  0.996872    111660

    accuracy                       0.998764    565576
```

Fig. 3. Results obtained after training RF on the OverSampled dataset

5.2 Oversampling

The oversampling method consists in creating new samples of under-represented classes of traffic and injecting them in the already collected dataset. Here, we will focus on how to create new instances of under-represented attacks. Since it is barely impossible to reproduce the actual conditions of the network system in which the dataset was created, we are duplicating existing samples. The RandomOverSampler function from the Python Imbalanced Learn library [11] is used for that purpose. Of course, the idea of randomly picking some samples of under-represented classes of attacks, and exactly repeating them in the dataset is clearly prone to heavy statistical biases, as we will have many strictly identical samples in both the training and testing datasets. However, the shrinkage floating parameter of the RandomOverSampler function can add a distortion to the samples to be duplicated. Since the distortion is proportional to the samples' class's features standard deviations, we have to OverSample by attack types.

Random Forest with binary labels has been trained on the oversampled dataset, and its detection accuracy has then been tested on the original one. Results are reported in Figure 3.

It exhibits an accuracy performance when the training is done on an oversampled dataset very similar to the one when the training is done on an undersampled training set: 0.9987 vs. 0.9989 respectively. It is also very close from the detection accuracy obtained without under- or oversampling: 0.9986 (Figure 1).

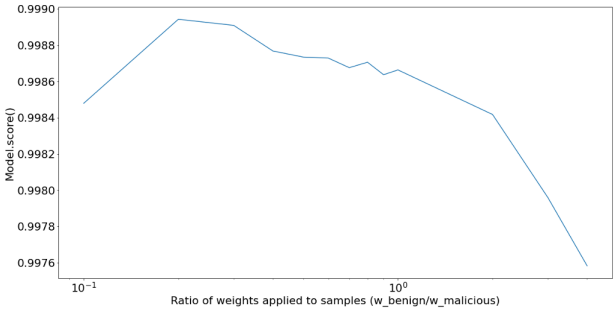


Fig. 4. Evaluation of the RF model for different Ratios of weight applied to classes

5.3 Class_weight

Given the poor improvement levels when training RF on under-or oversampled datasets, we finally tested another solution based on the "class_weight" entry parameter of the RandomForestClassifier function. This parameter allows the user to assign a weight to the different classes considered. This weight is aimed at changing the calculation of the Gini index. At each node split, when a tree is being built, the algorithm tries to find a feature and a threshold that minimize the Gini impurity. The Gini index (impurity index) for a node can be defined as:

$$I_g = \sum_i f_i \times (1 - f_i) = 1 - \sum_i f_i^2 \tag{5}$$

where f_i is defined according to the weight applied to the class i by the formula :

$$f_i = \frac{w_i.n_i}{\sum_i w_i.n_i} \tag{6}$$

where w_i is the weight applied to class i and n_i the number of samples belonging to class i

Then, if a weight of 2 is put on a class of samples (and 1 on the others), we expect the Gini index to give the same results as if these samples were appearing twice in the considered node. We expect this method to artificially multiply by 2 the weight of the related classes, and this without any risk of bias, at the opposite of what arises in the oversampling case. Indeed, these samples are in the dataset only once, and the algorithm simply counts them as twice more important.

The principle of class weighting has been tried with several different balances between the weights of benign vs. attack classes. In the following, the weight on the benign class is always set to 1, moving only the weights of the attack classes. Figure 4 shows the accuracy score as a function of the weights ratio between the benign and attack classes (in this first experiment, the same weight is set for all attack classes).

```
Model Score : 0.9989426708346889
Confusion Matrix :
[[453370    546]
 [    52 111608]]
   Attack Type                 Percentage of NON Detected
DoS Hulk                        0.008690928843020099 %
PortScan                        0.003131066441229883 %
DDoS                            0.019464341326689505 %
DoS GoldenEye                   0.19230769230769232 %
FTP-Patator                     0.0             %
SSH-Patator                     0.0             %
DoS slowloris                   0.41017227235438886 %
DoS Slowhttptest                0.09233610341643582 %
Bot                             5.612244897959184 %
Web Attack ? Brute Force         1.3245033112582782 %
Web Attack ? XSS                 0.8403361344537815 %
Infiltration                    40.0             %
Web Attack ? Sql Injection       42.857142857142854 %
Heartbleed                      0.0             %
```

Fig. 5. Accuracy Score, Confusion Matrix and FN percentages for the best RF model obtained with Class_weighting

Figure 4 exhibits that the best result is obtained for a weight ratio of 0.2, which means a weight of 1 for benign samples and 5 for attack samples. Given that there are originally five times more benign samples than attack samples in the dataset, the use of these weights somewhat brings us back to a balanced situation. The best accuracy obtained with this method is 0.9989 (similar to the first under-sampled approach). This "weight" based method nevertheless appears better than the under-sampled one, as with under-sampling, a lot of information are lost, as many benign samples are thrown away. Another advantage of the weight-based method appears when looking at the detection performance of individual classes of attacks, as shown on Figure 5.

With the weighed-based training approach, the detection rate of all attack types has been reduced. This is particularly the case for under-represented attack classes. For example, the Bot attack FN rate has been decreased from 63 to 5%. The Infiltration attack FN has been decreased from 80 to 40%. And more importantly, some attack types have seen all their samples detected as the Patator attacks. Globally, the total number of FN has been reduced to 52. Going further with the weighted-based training method, let us apply different weights on the different attack classes. In the preceding experiment a weight of 1 has been set to the benign class and a weight of 5 has been set to all attack class. Let us now change the weight of the Web SQL Injection attack from 5 to 10, the weights of the other attacks remaining the same. Figure 6 then shows that the FN rate for SQL Injection attack is reduced from 42 to 28%. On the other side, the detection ratio of bot attack is raising from 5 to 7%, and the one of Infiltration attack from 40 to 60%.

This type of results appeared for each attack type. When its weight has been increased with regard to the other attacks, the FN ratio has been decreased for this specific attack, but it is increased for many others. Differently changing the weight of attacks then creates some troubles in the way the training is done with RF.

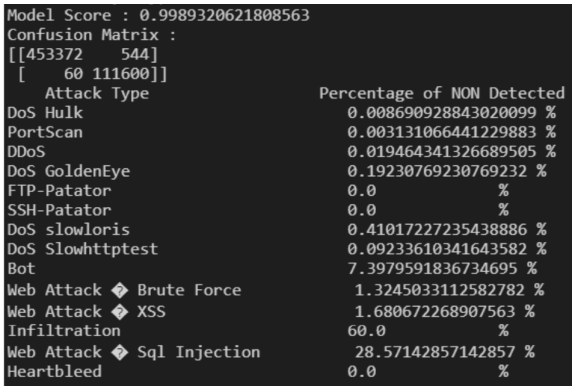


Fig. 6. Accuracy Score, Confusion Matrix and FN percentages when favoring the SQLI attack class with the weighted-based training method

6 Understanding the RF decision making

It is clear that using a black box approach is limiting the possibility of analyzing the decision making process of RF, and then its trustworthy nature. Thanks to the Scikit-Learn library [12], more information is available, especially on the trees inside the RF (here 80 trees), and for each tree, much information on the nodes and leaves are provided, as the features and thresholds considered. Given the material at disposal, this part would require another 20 pages paper. Because of space limit, this section will just briefly present two of the objectives of the study and the related conclusions.

The first study deals with evaluating the RF performances in detecting 0d attacks. For that, RF is trained with the same 80% of the dataset, as usual, minus the samples from the chosen attack class. Then the performance of RF is tested on the same test-set as usual, therefore containing an attack type it has not been trained on. It then appears that RF is not good at detecting attacks. It detects a 0d attack only when it has in its dataset attacks that have very similar shapes, and even in that favorable case, results are not very good. Only some DoS and Web attacks are well detected. For unique types of attack, like PortScan or Bot, the results are close to 100% of missed samples. The results also show that using binary labels provide better results, as it leads to a generalized forest model compared to a multiclass of attacks approach.

A second study deals with analyzing what features are taken as essential for each detection tree in the forest. The impurity score of Scikit-Learn library allows us to rank features from the most to the least important as considered by the RF algorithm. The real impact of all these features on the detection of attacks was then tested. The process consists in taking off features from the dataset and retraining RF to evaluate its new accuracy in detecting attacks. By taking off features from the dataset, starting from the most important ones, it appears that these important features are not so essential in the decision making. Indeed,

taking any of them off lowers the accuracy scores by only 0.01. Similarly, taking 20 features off the dataset sometimes does not imply any significant changes on the accuracy score. It is then very difficult to understand how RF makes its decision for classifying some slices of traffic as benign or malicious. A finer classification of attacks according to the multiclass approach is then very fuzzy, and the alarms not easy to understand for NetOps, that cannot trust them.

7 Concluding discussion

This paper provides a significant analysis of the RF algorithm for attack detection. RF has been configured in order to provide the best possible results. Then, the training dataset has been transformed in order to analyze its ability to detect specifically any kind of attacks contained in the dataset. This exhibits that improving the detection of one type of attacks can decrease the ability of detecting other kinds. A perfect detection has never been obtained. It is especially shown in the paper that it is impossible to explain how RF makes its detection decision. We argue that this conclusion can be extended to any machine learning algorithm for attack detection. They all can provide very good results higher than 99% of accuracy, but none of them is able to ensure a perfect detection of attacks without false alarms, and to provide explainable and trustworthy results.

This paper also exhibits that the proposed method in the related works are not necessarily usable with the expected level of accuracy in the actual network. Indeed, continuous training, or continuous algorithms parameters configuration are quite impossible to perform. However, these are essential stages for ensuring a high level of detection accuracy. Identifying the packets or flows constituting an attack is also difficult, very few algorithms being able of this identification necessary to trigger the appropriate countermeasures afterwards (in general only the ones with a solid pre-processing stage on the traffic and applying machine learning algorithms on some slices of the traffic can do it). In addition, the evaluation metrics as accuracy, F1-score, recall, etc. do not indicate what influences their results. As machine learning algorithms they are not easily explainable. This paper then aims at giving notice of the machine learning dogma in cyber-security. It also points out all biases in the current approaches and methodology. If machine learning is obviously necessary in cyber-security, it has to be combined with other techniques that can explain raised alarms for trustworthy detection results.

References

1. N. Elmrabit, F. Zhou, F. Li, H. Zhou, “Evaluation of Machine Learning Algorithms for Anomaly Detection”, IEEE International Conference on Cyber Security and Protection of Digital Services (Cyber Security), 2020
2. A. Kumar Jain, “Data Clustering: 50 Years Beyond K-Means”, Pattern Recognition Letters 31(8):651-666, June 2010

3. Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, F. Ahmad, "Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches", Transactions on Emerging Telecommunications Technologies, 32(1), 2021
4. A.P. Kelton, J. da Costa, O. Celso, R. Munoz, C. de Albuquerque, "Internet of Things: A survey on Machine Learning-Based Intrusion Detection Approaches", Computer Networks, Volume 151, Pages 147-157, 2019
5. F. Laghrissi, S. Douzi, K. Douzi, B. Hssina, "Intrusion Detection Systems Using Long Short-Term Memory (LSTM)", Journal on Big Data, vol.8, No 65, 2021
6. L. Alsulaiman, S. Al-Ahmadi, "Performance Evaluation of Machine Learning Techniques for DoS Detection", Journal of Wireless Sensor Network, 2021
7. P.A. Alves Resende, A. Costa Drummond, "A Survey of Random Forest Based Methods for Intrusion Detection Systems", ACM Computing Survey, vol. 51, No. 3, May 2018
8. J. Jiang, Q. Wang, Z. Shi, B. Lv, B. Qi, "RST-RF: A Hybrid Model Based on Rough Set Theory and Random Forest for Network Intrusion Detection", Proceedings of the 2nd International Conference on Cryptography, Security and Privacy, pp. 77?81, 2018
9. I. Sharafaldin, A. Habibi Lashkari, A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP, pages 108-116, 2018
10. RandomForestClassifier Scikit Learn: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
11. RandomOverSampler Imbalanced Learn: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html
12. Understanding the decision tree structure Scikit Learn : https://imbalanced-learn.org/stable/references/generated/imblearn.over_ sampling.RandomOverSampler.html