









Article

Synthetic Minority Oversampling Technique for Optimizing Classification Tasks in Botnet and Intrusion-Detection-System Datasets

David Gonzalez-Cuautle ¹, Aldo Hernandez-Suarez ¹, Gabriel Sanchez-Perez ¹, Linda Karina Toscano-Medina ¹, Jose Portillo-Portillo ¹, Jesus Olivares-Mercado ¹, Hector Manuel Perez-Meana ^{1,*} and Ana Lucila Sandoval-Orozco ² 

¹ Instituto Politecnico Nacional, ESIME Culhuacan, Mexico City 04440, Mexico; dgonzalezc1701@alumno.ipn.mx (D.G.-C.); alhernandezsu@ipn.mx (A.H.-S.); gasanchezp@ipn.mx (G.S.-P.); likatome@gmail.com (L.K.T.-M.); jportillo@ipn.mx (J.P.-P.); jolivares@ipn.mx (J.O.-M.)

² Departament of Electrical Engineering Faculty of Technology University of Brasilia (UnB), Campus Universitario Darcy Ribeiro, Brasilia CEP 70910-900, Brazil

* Correspondence: hmperez@ipn.mx; Tel.: +52-55-5624-2000

Received: 8 December 2019; Accepted: 17 January 2020; Published: 22 January 2020



Abstract: Presently, security is a hot research topic due to the impact in daily information infrastructure. Machine-learning solutions have been improving classical detection practices, but detection tasks employ irregular amounts of data since the number of instances that represent one or several malicious samples can significantly vary. In highly unbalanced data, classification models regularly have high precision with respect to the majority class, while minority classes are considered noise due to the lack of information that they provide. Well-known datasets used for malware-based analyses like botnet attacks and Intrusion Detection Systems (IDS) mainly comprise logs, records, or network-traffic captures that do not provide an ideal source of evidence as a result of obtaining raw data. As an example, the numbers of abnormal and constant connections generated by either botnets or intruders within a network are considerably smaller than those from benign applications. In most cases, inadequate dataset design may lead to the downgrade of a learning algorithm, resulting in overfitting and poor classification rates. To address these problems, we propose a resampling method, the Synthetic Minority Oversampling Technique (SMOTE) with a grid-search algorithm optimization procedure. This work demonstrates classification-result improvements for botnet and IDS datasets by merging synthetically generated balanced data and tuning different supervised-learning algorithms.

Keywords: imbalanced data; datasets; botnet detection; synthetic minority oversampling technique; machine learning; predictive models.

1. Introduction

Because of the rapid expansion of devices and applications offering services over the Internet, malicious actors have taken advantage of these connectivity resources to perform illegal activities like bot-based malware crafting. A botnet is a network of compromised computers that run a malicious program called a bot or agent [1]. The infected network is remotely controlled by a botmaster in charge of controlling the entire infrastructure through a Command and Control server (C and C), where all information from various network protocols, like HyperText Transfer Protocol (HTTP) and Internet Relay Chat (IRC), is stored. The involvement in botnet spreading is attributable to diverse motivations, varying from information leakage, spamming, phishing, and targeted Distributed Denial of Service (DDoS) attacks. A strain of botnet malware can eventually have harsh impact due that leads to great privacy, technological, and monetary losses.

Botnets and instruction attacks are packaged with more sophisticated techniques to camouflage themselves within a network, thereby emulating network traffic generated by benign applications, using strong encryption schemes through the HTTPS protocol [2] and unusual responses between C and C servers that are further accentuated with the constant injection of random and noisy network packets. Beside this, improper configurations in various networks devices can also lead to a potential malware infection [3]. As a result, infected hosts can be undetectable by network-security appliances such as firewalls, honeynets, Network Intruder Detection Systems (NIDS), Intruder Prevention Systems (IPS), and Intrusion Detection Systems (IDS).

Most physical and logical devices afford solutions on the basis of prebuilt signatures, anomaly based detection mechanics, or heuristic-based behavioral tests, proven to not be as effective due to many pitfalls leading to high false-positives ratios. An example of prominent detection failures is the constant updating processes that delay the identification of new threats, a lack of anomaly based rules for unknown samples, and behavioral tests that partially disclose malicious behaviors often mistaken as benign traffic.

On this basis, Machine-Learning (ML)-based solutions offer a sufficiently broad perspective on the early detection of a botnet attack. In this way, robust classification models can discover hidden malicious patterns in network flows, reveal notorious taxonomies, uncover particular attack dynamics, and distinguish unique features; important tasks that prebuilt security appliances may fail to assess [4]. However, for real-time botnet and IDS ML-based detection environments, data acquisition plays a major role, since the trade-off between dataset quality, quantity, and complexity reinforce the discriminative power of the chosen algorithms to effectively solve the initial formulation, thus reducing misclassification rates and increasing the trustworthiness of the implementation—a crucial step for protecting critical assets [5].

Although there are many publicly available datasets for botnets and IDSs resembling real scenarios [6–8], some drawbacks have been identified regarding the attributes of network captures (traffic redundancy), attack diversity, labeling-procedure reliability, and data dimensionality, more specifically, the compensation between the number of benign and malicious samples [9]. In supervised ML-based problems, when a certain number of classes are not equally distributed, the data are said to be unbalanced, impacting the algorithm capabilities to aptly learn from the samples of the predominant class, following to a degradation of classification performance. Considering [9,10], the credibility of an ML-based cyber–physical system depends on the predictive abilities of intelligent agents trained with a wide range of adversarial behavioral patterns, able to protect workloads against malicious activities. Indeed, data distribution has a great effect on the efficiency of different ML models [11,12]; thus, it is important to establish data-level strategies in the preprocessing and training stages. To deal with unbalanced datasets, two methods can be applied: resampling, which balances data by sample aggregation or deletion, and unbalanced learning [13] that seeks to improve the detection rate of minority classes. Data resampling can be performed in the following ways:

- Oversampling: samples from minority classes are duplicated until the amount is compensated with those from majority classes.
- Undersampling: samples belonging to majority classes are downsized at random until compensation with respect to the minority class is reached.
- Hybrid sampling: a superset is created with samples replicated from minority classes up to the same volume of the majority-class samples.

Even though previous research has implemented various resampling techniques, this work focuses on a novel approach named Synthetic Minority Oversampling Technique (SMOTE), a combination of oversampling, undersampling, and K-Nearest Neighbor (KNN) procedures to create a balanced set by virtue of synthetic generation [14,15]. KNN randomly selects k -nearest neighbors according to the magnitude of a majority class towards a minority class, emerging with new synthetic samples between the minority class and its nearest neighbors.

Even though data adjustments are important to increase the sensitivity of the classification models, it is important to consider a level of complexity in relation to algorithm performance. In [16], the authors suggested that a key factor for an ideal level is by considering inherent hyperparameters in which algorithms are constrained. In ML, hyperparameters are internal configuration variables used to improve skills at the learning phase, helping to make better generalization of the input information. The conventional scope for hyperparameter tuning is known as the grid search, a brute-force-like procedure that divides hyperparameters into regular grids exploiting the training models until the best grid point for minimizing the cost function is found.

In contrast with other state-of-the-art experiments, in this work we developed a deep inspection of botnet- and IDS-related datasets, tackling the unbalancing issues that cause irregular learning rates, and optimizing the training criteria of a set of supervised-learning algorithms, producing strengthened models that can transform data points into actionable knowledge. To achieve this goal, data acquisition must meet the following considerations [17]:

- Information ought to be from crude network flows as a main source of malicious-packet delivery;
- incorporation of a considerable collection of bot-based malware attacks obtained through environments closest to near-real contexts;
- data should cover requirements from operating cycles found in production deployments (working hours); and
- the proportion of more benign traffic packets must outnumber that of malicious ones since, in near-real-time conditions, only a slight portion of packets arise from infected sources.

This article is organized as follows. Section 2 provides an overview of several works related to botnet and intruder detection, the used algorithms, and the proposed balancing techniques for dealing with unbalanced datasets. Section 3 describes the proposed methodology for the following steps: data acquisition and examination, balancing minority-class samples via SMOTE, feature extraction and selection and grid search (hyperparameter tuning) techniques for a proposed algorithm portfolio. Section 4 details the obtained experiment results. Section 6 exemplifies the results discussion. Finally, Section 5 concludes this work.

2. Related Work

A network comprising bot-based malware and persistent intrusion attacks can cause extensive harm in a short period of time if cybersecurity consultants are not ceaselessly aware of what is going through their endpoints. Inspecting botnets and IDS network captures is not a trivial task; classical defensive tactics include honey-based detection, a helpful tool when computational resources are limited, and evasive rules can be quickly written from specific honeynets qualified to analyze malicious entries. Honey-based sensing is prone to be bypassed by advanced cyber-attacks, directly affecting scalability and practical responses from the net. IDS-based detection is mostly achieved by security policies triggered by the real-time monitoring of network activity; nevertheless, this kind of recognition cannot properly dissect the anomalous characteristics of malicious attacks, failing to perform persuasive mitigation [18]. As a consequence, botnet and IDS data analysis has attracted alternative areas of research, for instance, ML-based solutions. Prominent ML approaches facilitated the disclosure of underlying patterns of traffic flows, pointing out the importance of feature engineering (feature extraction and selection) and evaluating malicious traces with different assessments to reach higher accuracy in real implementations [19]. The authors in [20] addressed botnet and IDS detection relying on two notable ML ramifications, Supervised (SL) and Unsupervised Learning (UL). SL aims to acquire knowledge by mapping malicious and benign samples into a predefined set of labels, learning from intrinsic features and producing a classification model ready to predict incoming samples [21]. In contrast, UL employs a more rigorous exploration of similar patterns on underlying structures without labeling or categorizing samples, allowing the in-depth scrutiny of similarities between

different kinds of samples. The methodology of the present work is arranged by SL techniques, and related works are further described [22].

Because of the increasing number of types and characteristics of botnet and IDS records, classification remains a challenging research topic. According to the input metrics detailed in [23], malicious network flows can be outlined in four principal categories (Login, Inputs, Downloads and Geo-location), depending on specific missions for attackers. In general, well-known SL algorithms, including Logistic Regression (LG) [24], Support Vector Machine (SVM) [25], Artificial Neural Networks (ANN) [26], Decision Trees (DT) [27], Random Forest (RF) [27], Bayesian Networks (BN) [28], and Deep Learning (DL) Networks [29] have been fitted to overcome different menaces that directly depend on the conditions, circumstances, and settings in which botnet and IDS attacks are monitored and framed. Remarkable evidence is taken from IRC connections, P2P bots, DNS queries, anomalous traffic footprints, blacklisted IP addresses, irregular or malformed packet lengths, and abnormal intervals of multiple requests and responses over various network protocols [22]. Even so, some authors [30] agreed that given the nature of raw traffic flows, data are susceptible to balancing deficiencies between benign and malicious samples [31]. In an extensive set of trials, the authors in [32] considered that oversampling techniques are preferable over undersampling controls, since the resizing of large network flows can produce significant improvements in accuracy–sensitivity inter-relations, also highlighting SMOTE as the most suitable algorithm to conduct data-balancing measures.

In [15], a portion of malicious footprints were resampled via SMOTE, and tests proved that classification performance can be reinforced with the introduction of some level of noise to compensate for the merge of newly crafted observations. Similarly, in [33], random sampling fixed an offset between the number of minority classes by random generation, enabling a better detection ratio. The authors in [34] discussed that the repeated production and mixing of oversampled instances can create some disturbance in the original data distribution; instead, a series of SMOTE-based adaptations were adopted:

- Borderline-SMOTE1: only concentrates on edges of the smaller class; then, it crafts, calculates, and compares new synthetic samples around the distribution of the majority class to reconstruct the overall distribution of class samples.
- K-means SMOTE: identifies the area of the minority class and creates new instances on the basis of a seed pattern over the input space.
- Safe-level SMOTE: builds overall distribution upon a safe level of synthetic samples, using the nearest neighbor of minority instances.
- C-SMOTE: establishes a mean value center from the minority class samples as a basis, combining an interpolation algorithm to create and cluster new synthetic samples.
- CURE-SMOTE: provides synthetic samples by enhancing representative clusters from the original SMOTE distribution.

In [10], the authors said that malware-based datasets are for the most part constituted by benign samples, requiring a feasible structure that considers the degree of the belongingness of each minority class, a decisive factor to properly balance a dataset. As part of the methodology, a set of algorithms (DT, RF, Naive Bayes-NB, SVM, and AdaBoost-AB) were trough-fed preprocessed inputs under a fuzzy-theory-based SMOTE technique, reducing misclassification costs to a large extent [35].

The scalability constraints regarding the volume of data adjacent to Big Data (BD) security appliances, the inherent complexity of data centers work flows [36] and the properties of nonstructured information [37] are attainable by implementing appropriate preprocessing stages, especially if SL algorithms only consider overall accuracy without taking into account relative class distribution. Random Oversampling for Big Data (ROS-Big Data), Random Undersampling for Big Data (RUS-BigData), and Map Reduce (MR) are some methods responsible for resampling extensive concentrations of evenly distributed data. As the authors described in [37], such techniques were applied by unifying a SMOTE variation for BD, obtaining, at its best, a favorable number of synthetic

samples, avoiding some overgeneralization shortcomings to which SL are susceptible when handling a vast number of observations. Analogously, in [38], SMOTE was optimized by three major adjustments:

- SMOTE-TomekLinks: substantial feature examination is performed on most class samples, eliminating those that present an unbalancing factor against minority-class samples.
- SMOTE-ENN : aimed to weigh minority-class samples on the edge by employing Nearest Edited Neighbors (NEN).
- Borderline -SMOTE2: majority-class instances are oversampled, taking as reference their edge and inner-weighting factors.

Filtering and selecting inherent parameters in which SL algorithms may be constrained are important steps to strengthen classification models. In order to construct a robust methodology for classifying botnets and IDS data covering a full domain, it is important to establish a medium to choose settings that maximize the full potential of the selected algorithm. Positions and quantization levels can be determined by a well-known optimization technique known as grid search (GS), in which all regions of a defined space are searched, exhausting all possible combinations of values enclosed in a parameter. Little is known about improvements done to ML-based detection systems by optimizing a pool of algorithms. In early works [39], a support vector machine was tested by various kernel functions to detect massive intruder behaviors; furthermore, in [40], a first attempt using GS demonstrated that several algorithms can enhance predictive competencies for a series of malware families.

3. Proposed Methodology

The workflow of the proposed methodology is depicted in Figure 1. First, in the Data Acquisition block, a comprehensive search for datasets related to botnet and IDS traffic flows is conducted by collecting those that resemble real backgrounds, but with balancing issues. Furthermore, in the Feature Examination and Labeling block, datasets are subjected to an examination process aiming to exploit features that are considered useful by most authors in several state-of-the-art approaches [41–44]. Once data are normalized into a set of unique features, each sample is labeled as benign or malicious depending on the structure stated from the original data. During the Synthetic Minority Oversampling block, the percentage of minority-class samples from the training set are inspected to serve as a basis to oversample the data via synthetic production, resulting in a fully balanced training set. Subsequently, datasets are merged and split into training and testing sets. Therefore, in the Feature Extraction and Selection block, the balanced training set is preprocessed using Principal Component Analysis (PCA) [45] as the algorithm to extract and select the most informative and relevant features in a new dimensional subspace. In the Supervised Machine-Learning Classification Algorithms block, a portfolio of widely used algorithms is proposed to train the fully balanced set, thus enhancing the classification ratio through grid search, exhaustively tuning preconceived hyperparameters. Finally, the resulting classification models are evaluated by scoring the classification outcomes (predictions) from a testing set in terms of the following performance metrics: Accuracy, Recall, and F1-Score.

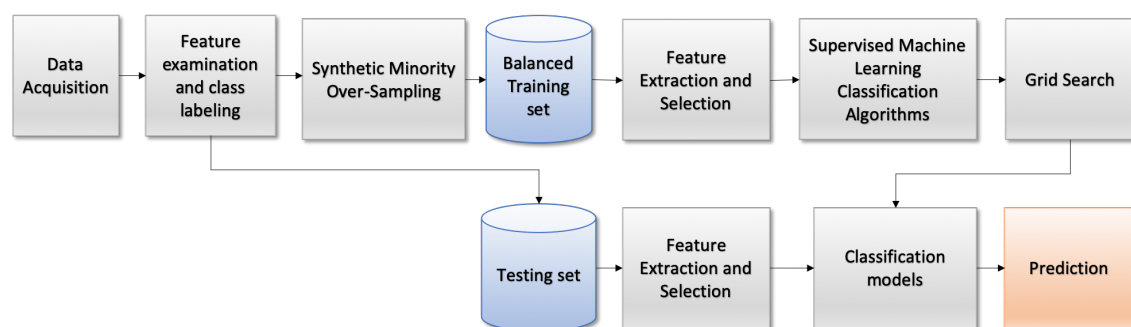


Figure 1. Workflow of proposed methodology.

3.1. Data Acquisition

The significant number of threats against network-based information services and the resilience of critical infrastructures constantly overlap. Profiling suspicious activities and taking prompt actions to evade or mitigate the possibility of an impact on security are a still big challenge. In an effort to provide robust and acute-sensing ML-based implementation, a meaningful portion of features must be directly reflected on the size and diversity of the collected observations. SL algorithms aim to make sense of the information provided by an external source; in this regard, a proper collection of samples is required to generalize the most important aspects to which data are exposed. In particular, a dataset extracted from real networks must reflect the innate behavior of adversaries attempting an attack. Therefore, it is not affordable to simulate the attack surface using simulated or mathematical models that can directly influence the usual state of the environment [17,44]. In this work, data acquisition was performed by examining specific datasets that reported new threats, authentic botnet and intruder footprints, regular traffic generated by benign applications alongside genuine network packets mirroring working-hour scenarios. The datasets used to accomplish the proposed methodology are described below:

- **ISCX-Bot-2014:** Provided by The Canadian Institute for Cybersecurity in which 16 different types of botnets are reported. These are, Neris, Rbot, Menti, Sogou, Murlo, Virut, NSIS, Zeus, SMTP Spam, UDP Storm, Tbot, Zero Access, Weasel, Smoke Bot, Zeus Control (C and C), and ISCX IRC bot. A total size of 5.3 GB of packet captures is formed by benign and malicious network traffic. As advised, 43.92% of overall network captures were identified as potentially malware-based. ISCX-Bot-2014 was built upon the consolidation of three main collections:
 1. **ISOT dataset [46]:** integrates various projects, particularly the Honeynet Project [47], Ericsson Research [48], and Lawrence Berkeley National Laboratory Research [49], each containing malicious traces from well-identified botnets like Flow Storm and Zeus. As for non-malicious connections, sources primarily came from game packages, HTTP traffic, and P2P applications.
 2. **ISCX 2012 IDS Dataset [50]:** crafted to be realistic, i.e., to include benign (HTTP, SMTP, SSH, IMAP, POP3, and FTP) and malicious (Botnet ISCX IRC) traffic produced by real devices.
 3. **Malware Capture Facility Project [51]:** the design of this dataset aims to generate and capture various traces of botnets, and it incorporates eight different kind of botnets (Neris, Rbot, Virut, NSIS, Menti, Sogou, and Murlo).
- **CIDDS-001 - Coburg Intrusion Detection:** provided by the University of Coburg in Germany, it was designed to gather real evidence of network intruders on the basis of anomaly-detection records and logs. Data mostly comprise network flows from small-business environments, including several email clients between rogue web-based modules and abnormal responses from multiple attacks, such as ping scanning, port scanning, brute force, and DoS.

3.2. Data Examination and Class Labeling

In Section 3.1 datasets were briefly introduced. As part of the initial examination of useful features, each dataset was explored for statistics regarding its composition. Because network traffic is mostly assembled in Packet Capture (PCAP) formats, the tool employed to dissect and replay the packets was *Tshark* [52], broadly used traffic-analyzer software. Then, data were submitted into a controlled sandboxed environment where malicious traces were monitored in depth. With regard to the ISCX-Bot-2014 dataset, Table 1 provides the percentage of different botnet flows captured on-the-fly.

Table 1. Distribution of different botnets used in dataset.

Botnet	Type	Flow Portion
Neris	IRC	25,967 (5.67%)
Rbot	IRC	83 (0.018%)
Menti	IRC	2878 (0.62%)
Sogou	HTTP	89 (0.019%)
Murlo	IRC	4881 (1.06%)
Virut	HTTP	58,576 (12.80%)
NSIS	P2P	757 (0.165%)
Zeus	P2P	502 (0.109%)
SMTP Spam	P2P	21,633 (4.72%)
UDP Storm	P2P	44,062 (9.63%)
Tbot	IRC	1296 (0.283%)
Zero Access	P2P	1011 (0.221%)
Weasel	P2P	42,313 (9.25%)
Smoke Bot	P2P	78 (0.017%)
Zeus Control (C and C)	P2P	31 (0.006%)
ISCX IRC bot	P2P	1816 (0.387%)

By large-scale observing the replay of the ISCX-Bot-2014 dataset, suspicious connections were identified in accordance with incoming and outgoing connections from IP addresses mapped to anomalous behaviors. The report is depicted in Table 2.

Table 2. List of malicious IPs on ISCX-Bot-2014 dataset.

Botnet	IP Addresses Ranges
IRC	192.168.2.112 -> 131.202.243.84 192.168.5.122 -> 198.164.30.2 192.168.2.110 -> 192.168.5.122 192.168.4.118 -> 192.168.5.122 192.168.2.113 -> 192.168.5.122 192.168.1.103 -> 192.168.5.122 192.168.4.120 -> 192.168.5.122 192.168.2.112 -> 192.168.2.110 192.168.2.112 -> 192.168.4.120 192.168.2.112 -> 192.168.1.103 192.168.2.112 -> 192.168.2.113 192.168.2.112 -> 192.168.4.118 192.168.2.112 -> 192.168.2.109 192.168.2.112 -> 192.168.2.105 192.168.1.105 -> 192.168.5.122
Neris	147.32.84.180
RBOT	147.32.84.170
Menti	147.32.84.150
Sogou	147.32.84.140
Murlo	147.32.84.130
Virut	147.32.84.160
IRCBot and black hole1	10.0.2.15
Black hole 2	192.168.106.141
Black hole 3	192.168.106.131
Tbot	172.16.253.130, 172.16.253.131 172.16.253.129, 172.16.253.240
Weasel	Botmaster IP: 74.78.117.238 Bot IP: 158.65.110.24
Zeus (zeus samples 1, 2 and 3, bin_zeus)	192.168.3.35, 192.168.3.25 192.168.3.65, 172.29.0.116
Osx_trojan	172.29.0.109
Zero access (zero access 1 and 2)	172.16.253.132, 192.168.248.165
Smoke bot	10.37.130.4

With the information presented above and by scrutinizing ISCX-Bot-2014 author specifications, useful features were determined in terms of the abnormal presence of packets during the request and responses from malicious sources, as shown in Table 3.

Table 3. Features examined in ISCX-Bot-2014 dataset.

	Feature	Description
1	Src_ip	Source IP Address
2	Src_port	Source Port
3	Dst_ip	Destination IP Address
4	Dst_port	Destination Port
5	Out_packets	Number of output packets
6	Out_bytes	Output byte number
7	Income_packets	Number of input packets
8	Income_bytes	Number of input bytes
9	Total_packets	Total number of transmitted packets
10	Total_bytes	Total number of transmitted bytes
11	Duration	Flow duration

The CIDDs-001 dataset is presented in plain-text format and was provided by prebuilt attributes. However, some values were missing or had insufficient information, yielding a large amount of noise. Manual inspection preceded to normalize the number of features and delete disproportional samples. Table 4 details each feature after the removal of noisy elements.

Table 4. Features used in CIDDs-001 dataset.

	Feature	Description
1	Src ip	Source IP Address
2	Src port	Source Port
3	Dest ip	Destination IP Address
4	Dest port	Destination Port
5	Duration	Duration of the flow
6	Bytes	Number of transmitted bytes
7	Packets	Number of transmitted packets
8	Class	Class label (normal, attacker, victim, suspicious, or unknown)

Consequently, samples tagged with discrete or numerous categorical values were standardized into malicious and benign classes. For the CIDDs-001 dataset, a total of 49,554 samples were labeled as benign and 105,530 as malicious, while in ISCX-Bot-2014 benign samples were over 119,287, and 56,572 were malicious.

3.3. Synthetic Minority Oversampling

After feature examination, the number of benign samples was greater than that of malicious ones, a condition described on Sections 1 and 2 upon which flows generated by botnets or intrusion attacks can be masked within the network. As a consequence, the aforementioned security appliances were unable to correctly record their activity, leading to insufficient proof of malicious activity. As an example, benign traces clearly prevailed in a great part of the observations owing to large connections between safe client-server architectures. Conversely, the malicious ones were not frequently spotted, requiring more thorough inspection. This is explained by unusual botnets and C and C conversations, odd targeted attacks exploiting some vulnerability, passive scanning, unfamiliar Denial of Service attempts, and brute-force attacks that are rarely experienced. Eventually, this problem sometimes causes ML-based sensors to deteriorate in learning potential, tending to mislead the detection of suspicious objects. To tackle the absence of malicious patterns, this works pursued to resample

previously studied datasets by applying SMOTE; in this section, we describe the steps considered develop the technique.

Consider each dataset as representation of n samples $X \in \mathbb{R}^{n \times m}$ mapped with $y_i \in \{1, 2, \dots, C\}$ labels. The unbalanced subsets are defined as $X_{min} \subset X$ for minority-class samples and $X_{maj} \subset X_{train}$ for majority observations, respectively; so, $X_{min} \cup X_{maj} = X; \forall X_{min} < X_{maj}$. Depending upon the amount of oversampling based on the extent of X_{maj} , synthetic data are generated by pointing to a space of similar features between instances belonging to $x_i \in X_{min}$ from a certain neighborhood. Then, k -nearest neighbors are computed by considering the smallest Euclidean distance between the neighbor and the rest of X_{min} instances; the closest k neighbors serve as reference points to create new in-between samples. The interpolation of those observations is described in Equation (1):

$$X_{syn} = X + (X_k - X) \cdot e, \quad (1)$$

where $X_k \in X_{min}$ is one of K -nearest neighbors ($k = 1, 2, 3, \dots, K$) from selected samples x_i , e is a random number belonging to the range of $[0, 1]$ that represent the number of instances between x_i and X_k , and X_{syn} is the new synthetic sample. In this way, SMOTE balanced both datasets as follows:

- CIDD-001: comprised 248,134 samples, where 50% corresponded to benign and the rest to malicious.
- ISCX-Bot-2014: comprised 133,226 samples, where 50% corresponded to benign and the rest to M malicious,

Finally, datasets were divided into training X_{train} and test X_{test} subsets with a random split of samples of 80% of the overall data for training purposes and 20% for testing the resulting models.

3.4. Feature Extraction and Selection

As indicated in the literature [53–57], it is essential to strengthen evidence provided by the features described on each dataset as this positively influences SL training routines by capturing maximum variability of the inputs and expanding the capacity of inference to the resulting models. If at some point ISCX-Bot-2014 and CIDD-001 features produce correlation effects, this downgrades the computation of the algorithm. With the incorporation of dimensionality reduction, features that produce wrong variability are discarded, and the remaining inputs are represented in a new subspace of lower dimension based on their variability. In this proposal, X_{train} was preprocessed via Principal Component Analysis (PCA) [45], a dimensionality reduction used to describe features in a new set of noncorrelated variables.

Principal component \mathbf{z}_1 is defined as the linear combination of the original features with maximum variation. Values in this first component are represented in Equation (2):

$$\mathbf{z}_1 = O\mathbf{a}_1, \quad (2)$$

where n is the number of samples, O is the matrix of observations with zero mean, and \mathbf{a}_1 is the vector that maximizes variance $\text{var}(\mathbf{z})$, as formulated in Equation (2).

$$\frac{1}{n}\mathbf{z}_1'\mathbf{z}_1 = \frac{1}{n}\mathbf{a}_1'O'O\mathbf{a}_1 = \mathbf{a}_1'\mathbf{S}\mathbf{a}_1, \quad (3)$$

where \mathbf{S} is the matrix of variances and covariances from the observations. In order to solve Equation (3), constraint $\mathbf{a}_1'\mathbf{a}_1 = 1$ must be established using a Lagrange multiplier:

$$M = \mathbf{a}_1'\mathbf{S}\mathbf{a}_1 - \omega(\mathbf{a}_1'\mathbf{a}_1 - 1); \quad (4)$$

maximizing Equation (4) implies deriving it with respect to the components of \mathbf{a}_1 until zero:

$$\frac{\partial M}{\partial \mathbf{a}_1} = 2\mathbf{S}\mathbf{a}_1 - 2\omega\mathbf{a}_1 = 0 \quad (5)$$

Equation (5) results in $\mathbf{S}\mathbf{a}_1 = \omega\mathbf{a}_1$, where \mathbf{a}_1 is an eigenvector of \mathbf{S} , and ω the corresponding eigenvalue.

Ultimately, a new subspace is given by the two main components for both datasets, representing new variables with 89% of captured relevance for ISCX-Bot-2014 and 94% for CIDD5-001.

3.5. Supervised Machine-Learning Algorithms

An algorithm portfolio was constructed by a set of supervised machine-learning algorithms; then, each algorithm is trained using identical set X_{train} . Therefore, the resulting classification models are subjected to rigorous evaluation against test set X_{test} with different performance metrics; the model with the best score can be selected as a candidate to solve the initial classification problem. In order to build a portfolio, the following assumption must summarize the proposed goal:

Assumption 1. *There must be a set of algorithms $a \in A$ with a P classification problem that makes it easier to identify given a selection process S , in which each algorithm can be optimally performed given set of features f on the same environment [58].*

To solve S , the following criteria were taken into account:

- Define supervised classification algorithms to use in a specific context;
- optimize each algorithm by means of hyperparameter tuning;
- evaluate each algorithm in isolation from its performance metrics;
- evaluate each algorithm in parallel in the same environment; and
- compare the performance of each algorithm with those used in the literature.

As mentioned in Section 2, the set of algorithms reported in the state of the art [59–63] for botnet and Intruder detection [64] were selected to construct the algorithm portfolio:

- K-Nearest-Neighbor (KNN);
- Support Vector Machine (SVM);
- Logistic Regression (LR);
- Decision Trees (DT); and
- Random Forest (RF).

3.6. Grid Search

Grid Search (GS) or hyperparameter search [16] is the process of iterating a set of hyperparameters λ contained in a machine-learning algorithm $a \in A$ to identify optimal subset λ^* that maximizes the performance of resulting model M and minimize loss function $L(X_{train}, M)$. Equation (6) computes the main idea behind grid search.

$$\begin{aligned} \lambda^* &= \underset{\lambda}{\operatorname{argmin}} \{L(a, X_{train}, \lambda)\} \\ &= \underset{\lambda}{\operatorname{argmin}} \{F(L(a, X_{train}, \lambda))\}' \end{aligned} \quad (6)$$

where L indicates the loss function, X_{train} is the training set, a is the chosen algorithm, λ is the set of hyperparameters to be optimized, and F is an objective function aimed to test the performance of new set of tuned hyperparameters λ^* .

Table 5 indicates used hyperparameters λ in algorithm portfolio A and their corresponding values.

Table 5. Algorithms with their inherent hyperparameters and values.

Algorithm (a)	λ	Values /Ranges
KNN	Algorithm used to compute nearest neighbors	Ball tree, KD tree, Brute Force
	No. of neighbors to use	{1,50}
	Weight function used in prediction	Uniform, By distance
SVM	Penalty parameter C of error term	{0.0001, 0.001, 0.01, 0.1}
	Decision function of shape	One-vs-one, One-vs-rest
	Kernel type to be used in algorithm	Polynomial, Linear, RBF
LR	Inverse of regularization strength of term C	{0.0001, 0.001, 0.01, 0.1}
	Norm used in penalization function	ℓ_1 , ℓ_2
	Algorithm to use in optimization problem	Linear LBFGS *, SAG †, SAGA ‡
DT	Maximum tree depth	{1,30}
	No. of features to consider for best split	{1,100}
	Strategy used to choose split at each node	Sqrt,
		Log2 Best, Random
RF	Use bootstrap samples when building trees	True, False
	Function to measure split quality	Entropy,GINI §
	Max tree depth	{1,30}
	No. of features to consider for best split	{1,30}
	Min. no. of samples required to be at leaf node	{1,30}
	Min. no. of samples required to split internal node	{1,30}
	No. of trees in forest	{1,10}

*Limited-memory BFGS; † Stochastic Average Gradient; ‡ Fast Incremental Gradient Method; § GINI Index.

4. Experiment Results

This section presents the experiment results using finest hyperparameters λ^* of each classification model resulting from A . X_{train} was submitted to an isolated trial-validation test using K-fold cross-validation (10 folds per trial) [65]. This provided better generalization in the learning process, monitoring the average results of each trial. Then, the final models were scored using each testing set X_{test} for the balanced ISCX-Bot-2014 and CIDDS datasets in terms of the following performance metrics: Accuracy, Recall, and F1-score. In order to compare the results of the improvements of the proposed methodology (SMOTE + GS), ISCX-Bot-2014 and CIDDS-001 were tested with the same algorithm portfolio but without any balancing or hyperparameter optimization techniques. Tables 6–9 show the performance scores of each dataset.

Table 6. Unbalanced CIDD-001 scores.

Algorithm	Precision	Recall	F1-Score
KNN	0.9949	0.9968	0.9958
SVM	0.8648	0.8968	0.8762
LR	0.7002	0.7012	0.7049
DT	0.9958	0.9928	0.9906
RF	0.9970	0.9922	0.9915

Table 7. Unbalanced ISCX-Bot-2014 scores.

Algorithm	Precision	Recall	F1-Score
KNN	0.9936	0.9954	0.9915
SVM	0.7832	0.7907	0.7826
LR	0.6861	0.6944	0.6895
DT	0.9947	0.9929	0.9932
RF	0.9930	0.9951	0.9947

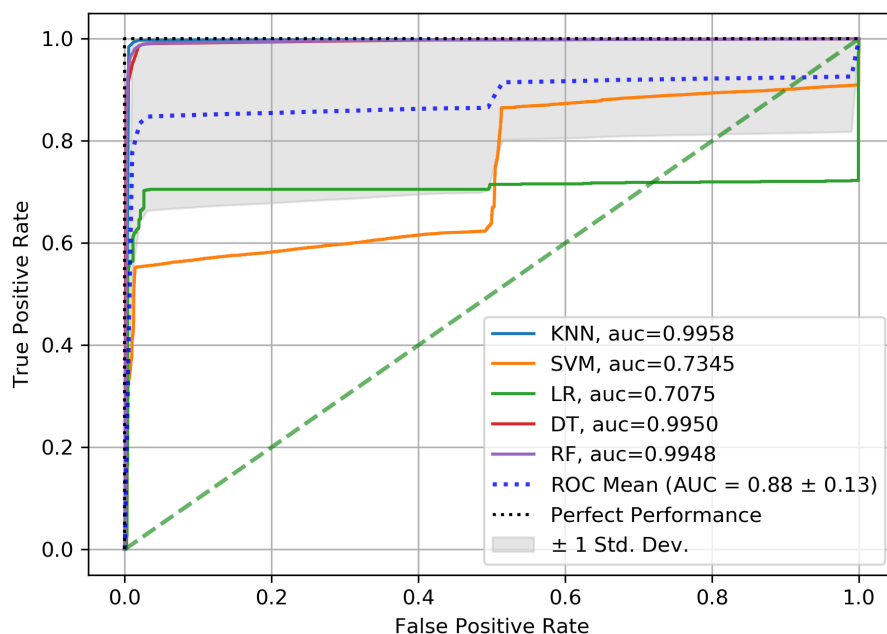
Table 8. CIDD-001 (SMOTE + GS) scores.

Algorithm	Precision	Recall	F1-Score	Best Hyperparameter/Value
KNN	0.9812	0.9817	0.9815	Algorithm used to compute nearest neighbors: Ball tree No. of neighbors to use: 4 Weight function used in prediction: By distance
SVM	0.8961	0.9344	0.9091	Penalty parameter C of error term: 0.001 Decision function of shape: One-vs-one Kernel type to be used in the algorithm: Linear
LR	0.7208	0.7188	0.7196	Inverse of regularization strength of term C: 0.0001 Norm used in penalization function: ℓ_1 Algorithm to use in optimization problem: Linear
DT	0.9816	0.9836	0.9826	Maximum tree depth: 9 No. of features to consider for best split: 12 Strategy used to choose split at each node: Best
RF	0.9814	0.9833	0.9823	Use bootstrap samples when building trees: False Function to measure split quality: GINI Max tree depth: 9 No. of features to consider for best split: 12 Min. no. of samples required to be at leaf node: 2 Min. no. of samples required to split internal node: 9 No. of trees in forest: 1

Table 9. ISCX-Bot-2014 (SMOTE + GS) scores.

Algorithm	Precision	Recall	F1-Score	Best Hyperparameter Value
KNN	0.9677	0.9676	0.9676	Algorithm used to compute nearest neighbors: Ball tree No. of neighbors to use: 4 Weight function used in prediction: by distance
SVM	0.8152	0.8261	0.8117	Penalty parameter C of error term: 0.01 Decision function of shape: One-vs-one Kernel type to be used in the algorithm: Linear
LR	0.7836	0.7789	0.7816	Inverse of regularization strength of term C: 0.0001 Norm used in penalization function: ℓ_1 Algorithm to use in optimization problem: LBFGS
DT	0.9797	0.9800	0.9799	Maximum tree depth: 9 No. of features to consider for best split: 12 Strategy used to choose split at each node: Best
RF	0.9796	0.9799	0.9798	Use bootstrap samples when building trees: False Function to measure split quality: GINI Max tree depth: 9 No. of features to consider for best split: 7 Min. no. of samples required to be at leaf node: 2 Min. no. of samples required to split internal node: 9 No. of trees in forest: 1

The efficiency of each model was determined by Receiver Operating Characteristic Area Under the Curve, which measures performance by indicating the ability of the model to distinguish classes; the higher the AUC is, the better the prediction of the model. Figures 2 and 3 depict each tuned classification model.

**Figure 2.** Receiver Operating Characteristic (ROC) for balanced subset CIDD5-001.

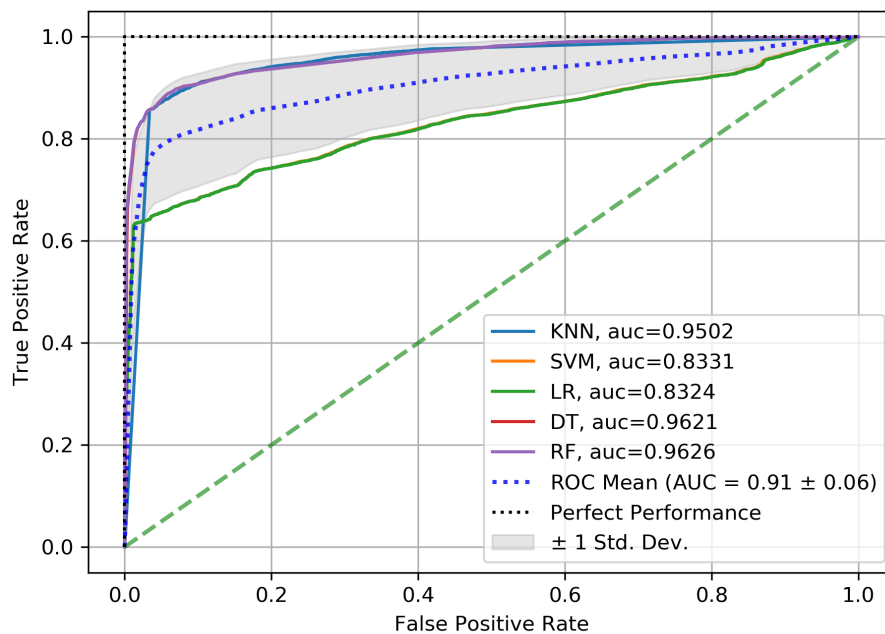


Figure 3. Receiver Operating Characteristic (ROC) for balanced subset ISCX-Bot-2014.

5. Results and Discussion

The experiment results for the botnet and IDS datasets using SMOTE + GS demonstrated significant improvement for the prediction of malicious samples in highly unbalanced datasets as compared to what the authors in [66,67] achieved by means of resampling techniques. Indeed, there was a lack of meticulous inspection with reference to sampling engineering, and an absence of algorithm evaluation regarding optimization and future assessment. Even though in [66] the performance of some SL models reached high accuracy rates, it was not explained if some extraordinary scores were the product of overfitting conditions mainly caused by a balancing factor and the employed type of resampling. In our approach, it is emphasized that by employing SMOTE and supervising forthcoming model, K-fold cross-validation overfitting can be significantly avoided; in this way, during the learning stage, X_{train} was split into a validation set of equal size of training data, making it possible to correctly verify average accuracy in different trials. As demonstrated in the results, feature extraction and selection played an important role since manual inspection was not sufficient to exploit the inherent values of each feature, but is also indispensable for a better outlook on data representations that maximize variability in a feature space. Moreover, algorithms are by default constrained by inner parameters, so that in the learning process they must be put through rigorous search to find optimal values, enhancing a more favorable sensing ratio. To present the improvements of the present work, similar proceedings are compared in Table 10.

In addition, another significant factor for the improvement of the classification models created in this work in both datasets was the fact that they were a configuration that was external to the model and whose value could not be estimated from the learned patterns, but from the performance of the algorithm in itself during the training stage (Figure 1), helping to improve the classification performance of the model, i.e., for each iteration of the hyperparameters in each algorithm of the proposed portfolio, the best were found to be their combinations, strengthening the predictions of each suggested model, as shown in the table above.

The hyperparameters for the KNN algorithm of the CIDDS-001 dataset with the best performance used four neighbors because the algorithm that calculates the closest proximity between neighbors (Ball Tree) and the function weight (Distance) used in the prediction showed that all evaluation metrics oscillated by 98% in their results. In the case for the SVM, the penalty parameter (C) showed that with a margin of error of 0.001, a linear kernel and the form-decision function (o-v-r) taking into account the

number of samples and the number of classes for this study, binary classes generated results in 90% average. For the LR algorithm, penalty rule ℓ_1 showed a very efficient regularization force (0.0001) and, with the help of the LBFGS optimization algorithm, evaluation metrics reached above 70%. With a depth of nine nodes for the DT, considering the best division for each was 12, in addition to showing favorable results in the metrics (average of 98%), reduced memory consumption, complexity, and tree size for the produced classifier model. Finally, in RF, in the same way as in DT, the depth of its nodes and the best division in the trees were nine and 12, respectively, because it is a DT derivative. By having a fully balanced dataset, sampling of features to be considered for the best division in each tree was disabled (Bootstrap = False) so that its construction took into account the whole set. Only two samples were enough to form a leaf (binary class) inside the tree, achieving results above 98%.

Table 10. Comparative analysis between related works for CIDDS-001 and ISCX-Bot-2014 datasets.

Methodology	Dataset	Algorithm	Accuracy
Verma A. et al. [66]	CIDDS-001	KNN	93.87%
SMOTE+GS	CIDDS-001	KNN	98.72%
Bijalwan A. et al. [67]	ISCX-Bot-2014	KNN	93.87%
		DT	93.37%
		Bagging with KNN	95.69%
		Ada-Boost with DT	94.78%
		Soft voting of KNN and DT	96.41%
SMOTE + GS	ISCX-Bot-2014	KNN	98.72%
		SVM	97.35%
		LR	97.89%
		DT	98.65%
		RF	98.84%

For the hyperparameters of the ISCX-Bot-2014 dataset, the algorithm in KNN (Ball Tree) calculated that the optimal number of nearest neighbors was four because the function weight (Distance) was taken into account to determine them, achieving results of more than 96% in its evaluation metrics. The form-decision function (o-v-r) in SVM, the kernel type (Linear), and the penalty parameter of 0.01 had a positive impact when generating the predictive model, reaching metrics above 80% in the classification of botnet samples. Evaluations in the performance metrics of the predictive model with LR were the lowest of all the algorithms of the used portfolio (below 79% in the classification) despite having good regularization force (0.0001); the penalty rule (ℓ_1) affected the results due to the optimization algorithm used (LBFGS). In DT, in the same way as in the CIDDS-001 dataset, the depth of the nodes was nine and their best divisions were 12 because this led to the creation of the best tree with an average performance of 97%. As already mentioned, RF is a derivative of DT, so the depth of its nodes was 9, while its best divisions were seven, taking into account the function to measure its quality (GINI) because counting in unique values is favored and therefore better classification (average of 97%). This being a binary classification problem, leaves were composed of only two samples and a single tree.

In future work, a real-time ML-based implementation of SMOTE + GS is proposed by considering the suggestions adopted in [68]. First, a specialized host is indispensable to filter, examine, and transform network packages into useful features; then, a labeling process must map samples through a botnet or intruder category and store them in a knowledge-based database. Furthermore, observations must be trained in an offline fashion for an amount of time; consecutively, an actuator layer is responsible for establishing the ML module to inspect the communication network and force to identify the type of transmitted package. Finally, a handler must decide to mitigate the identified threat, send an alarm to network administrators, or execute a defense mechanism. An important factor for the reproduction of the proposed in real applications is also to gather samples over a four-week timespan because, on average, inactive threats exhibit their functionality.

6. Conclusions

Currently, new and sophisticated techniques are developed by malicious actors (intruders) to evade detection (strong encryption in their transmitted packets, constant injection of packets that generate noise, and abnormal responses between bots and their C and C) mechanisms within security appliances, resulting in an important security concern in the fact that embedded solutions base their strength on limited actions established by signatures, anomaly recognition, and heuristic behaviors, flawed in adopting an efficient perspective for timely and accurate detection. Instead, ML-based implementations are being considered to tackle classical applications; however, a limitation occurs when the construction of datasets is prone to unbalanced information, leading to downgrading the learning stages of classification algorithms and causing a bad interpretation of malicious patterns. For this reason, we proposed in this work a data resampling technique known as SMOTE to subject to two datasets that closely resembled real botnet and IDS information (CIDDS-001 and ISCX-Bot-2014) and create a fully balanced dataset via synthetic generation. Experiment results proved that by testing different supervised-learning algorithms via the grid-search hyperparameter-optimization technique with a balanced training set, the performance of the final classification models could be substantially improved. As compared with other state-of-the-art approaches with the same set of algorithms, SMOTE + GS reached better results in terms of accuracy (KNN: 98.72%, SVM: 97.35%, LR:97.89%, DT: 98.65%, and RF:98.84%).

Author Contributions: D.G.-C. developed the conceptualization and the proposed methodology for balancing the used datasets. A.H.-S. and G.S.-P. developed the computer program to evaluate the performance of the methodology with balanced datasets. J.O.-M. and L.K.T.-M. performed analysis and the validation of the dataset balance. J.P.-P., H.M.P.-M., and A.S.-O. monitored the results and how efficient the proposed methodology was. All authors participated in the write-up and review of the article. All authors have read and agreed to the published version of the manuscript.

Acknowledgments: The authors thank the National Science and Technology Council of Mexico (CONACyT), the Instituto Politécnico Nacional, and the Grupo de Análisis, Seguridad & Sistemas (GASS) of the Universidad Complutense de Madrid for the financial support for this research.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hsu, F.H.; Ou, C.W.; Hwang, Y.L.; Chang, Y.C.; Lin, P.C. Detecting web-based botnets using bot communication traffic features. *Secur. Commun. Netw.* **2017**, *2017*, 11. doi:10.1155/2017/5960307.
2. Idhammad, M.; Afdel, K.; Belouch, M. Detection system of HTTP DDoS attacks in a cloud environment based on information theoretic entropy and random forest. *Secur. Commun. Netw.* **2018**, *2018*, 13. doi:10.1155/2018/1263123.
3. Varela-Vaca, Á.J.; Gasca, R.M.; Ceballos, R.; Gómez-López, M.T.; Torres, P.B. CyberSPL: A Framework for the Verification of Cybersecurity Policy Compliance of System Configurations Using Software Product Lines. *Appl. Sci.* **2019**, *9*, 5364.
4. Sinclair, C.; Pierce, L.; Matzner, S. An application of machine learning to network intrusion detection. In Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99), Scottsdale, AZ, USA, 6–10 December 1999; pp. 371–377.
5. Gupta, M. *Handbook of Research on Emerging Developments in Data Privacy*; IGI Global: Hershey, PA, USA, 2014; pp. 438–439.
6. Małowidzki, M.; Berezinski, P.; Mazur, M. Network intrusion detection: Half a kingdom for a good dataset. In Proceedings of the NATO STO SAS-139 Workshop, Portugal, April 2015.
7. Hochschule Coburg. Available online: <https://www.hs-coburg.de/fileadmin/hscoburg/WISENT-CIDDS-001.zip/> (accessed on 16 May 2019).
8. Canadian Institute for Cybersecurity. Botnet Dataset. Available online: <https://www.unb.ca/cic/datasets/botnet.html> (accessed on 15 May 2019).

9. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. doi:10.1016/j.future.2019.05.041.
10. Xu, Y.; Wu, C.; Zheng, K.; Niu, X.; Yang, Y. Fuzzy-synthetic minority oversampling technique: Oversampling based on fuzzy set theory for Android malware detection in imbalanced datasets. *Int. J. Distrib. Sens. Netw.* **2017**, *13*. doi:10.1177/1550147717703116.
11. Schubach, M.; Re, M.; Robinson, P.N.; Valentini, G. Imbalance-aware machine learning for predicting rare and common disease-associated non-coding variants. *Sci. Rep.* **2017**, *7*, 2959. doi:10.1038/s41598-017-03011-5.
12. Pham, T.S.; ; Hoang, T.H. Machine learning techniques for web intrusion detection—A comparison. In Proceedings of the 2016 Eighth International Conference on Knowledge and Systems Engineering (KSE), Hanoi, Vietnam, 6–8 October 2016; pp. 291–297.
13. Johnson, J.M.; Khoshgoftaar, T.M. Survey on deep learning with class imbalance. *J. Big Data* **2019**, *6*, 27. doi:10.1186/s40537-019-0192-5.
14. Seo, J.H.; Kim, Y.H. Machine-Learning Approach to Optimize SMOTE Ratio in Class Imbalance Dataset for Intrusion Detection. *Comput. Intell. Neurosci.* **2018**, *2018*, 11. doi:10.1155/2018/9704672.
15. Ma, L.; Fan, S. CURE-SMOTE algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests. *BMC Bioinform.* **2017**, *18*, 169. doi:10.1186/s12859-017-1578-z.
16. Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
17. Ring, M.; Wunderlich, S.; Grödl, D.; Landes, D.; Hotho, A. Flow-based benchmark data sets for intrusion detection. In Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS), Dublin, Ireland, 29–30 June 2017; pp. 361–369.
18. Hoang, X.; Nguyen, Q. Botnet detection based on machine learning techniques using DNS query data. *Future Internet* **2018**, *10*, 43. doi:10.3390/fi10050043.
19. Conti, M.; Dargahi, T.; Dehghantanha, A. *Cyber Threat Intelligence: Challenges and Opportunities*; Springer: New York, NY, USA, 2018; pp. 1–6.
20. Stevanovic, M.; Pedersen, J. *MMachine Learning for Identifying Botnet Network Traffic*; Technical Report; Networking and Security Section, Department of Electronic Systems, Aalborg University: Aalborg, Denmark, 2013.
21. Biradar, A.D.; Padmavathi, B. BotHook: A Supervised Machine Learning Approach for Botnet Detection Using DNS Query Data. In Proceedings of the 2019 IEEE International Conference on Computation, Communication and Engineering (ICCCE), Fujian, China, 8–10 November 2019.
22. Miller, S.; Busby-Earle, C. The role of machine learning in botnet detection. In Proceedings of the 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST), Barcelona, Spain, 5–7 December 2016; pp. 359–364.
23. Carrasco, A.; Roperio, J.; de Clavijo, P.R.; Benjumea, J.; Luque, A. A Proposal for a New Way of Classifying Network Security Metrics: Study of the Information Collected through a Honeypot. In Proceedings of the 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Lisbon, Portugal, 16–20 July 2018; pp. 633–634.
24. Bapat, R.; Mandya, A.; Liu, X.; Abraham, B.; Brown, D.E.; Kang, H.; Veeraraghavan, M. Identifying malicious botnet traffic using logistic regression. In Proceedings of the 2018 Systems and Information Engineering Design Symposium (SIEDS), Charlottesville, VA, USA, 27 April 2018; pp. 266–271.
25. Lin, K.C.; Chen, S.Y.; Hung, J.C. Botnet detection using support vector machines with artificial fish swarm algorithm. *J. Appl. Math.* **2014**, *2014*, 9. doi:10.1155/2014/986428.
26. Letteri, I.; Del Rosso, M.; Caianiello, P.; Cassioli, D. Performance of Botnet Detection by Neural Networks in Software-Defined Networks. In Proceedings of the Second Italian Conference on Cyber Security (ITASEC), Milan, Italy, 6–9 February, 2018.
27. Bonneton, A.; Migault, D.; Senecal, S.; Kheir, N. Dga bot detection with time series decision trees. In Proceedings of the 2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), Kyoto, Japan, 5 November 2015; pp. 42–53.

28. Dollah, R.F.M.; Faizal, M.A.; Arif, F.; Mas'ud, M.Z.; Xin, L.K. Machine learning for HTTP botnet detection using classifier algorithms. *J. Telecommun. Electron. Comput. Eng.* **2018**, *10*, 27–30.
29. Khan, R.U.; Zhang, X.; Kumar, R.; Sharif, A.; Golilarz, N.A.; Alazab, M. An Adaptive Multi-Layer Botnet Detection Technique Using Machine Learning Classifiers. *Appl. Sci.* **2019**, *9*, 2375. doi:10.3390/app9112375.
30. Harun, S.; Bhuiyan, T.H.; Zhang, S.; Medal, H.; Bian, L. Bot Classification for Real-Life Highly Class-Imbalanced Dataset. In Proceedings of the 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), Orlando, FL, USA, 6–10 November 2017; pp. 565–572.
31. Le, D.C.; Zincir-Heywood, A.N.; Heywood, M.I. Data analytics on network traffic flows for botnet behaviour detection. In Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI), Athens, Greece, 6–9 December 2016; pp. 1–7.
32. Kudugunta, S.; Ferrara, E. Deep neural networks for bot detection. *Inf. Sci.* **2018**, *467*, 312–322. doi:10.1016/j.ins.2018.08.019.
33. Cho, C.Y.; Shin, E.C.R.; Song, D. Inference and analysis of formal models of botnet command and control protocols. In Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS), Chicago, IL, USA, 4–8 October 2010; pp. 426–439.
34. Chowdhary, C.L. *Intelligent Systems: Advances in Biometric Systems, Soft Computing, Image Processing, and Data Analytics*; CRC Press: Boca Raton, FL, USA, 2020.
35. Zimmermann, H.J. *Fuzzy Set Theory—and Its Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2011.
36. Fernández-Cerero, D.; Varela-Vaca, Á.J.; Fernández-Montes, A.; Gómez-López, M.T.; Álvarez-Bermejo, J.A. Measuring data-centre workflows complexity through process mining: The Google cluster case. *J. Supercomput.* **2019**, 1–30; doi:10.1007/s11227-019-02996-2.
37. Basgall, M.J.; Hasperu, W.; Naiouf, M.; Fernández, A.; Herrera, F. SMOTE-BD: An Exact and Scalable Oversampling Method for Imbalanced Classification in Big Data. In Proceedings of the VI Jornadas de Cloud Computing & Big Data (JCC&BD), La Plata, Argentina, 25–29 June 2018.
38. Ramentol, E.; Caballero, Y.; Bello, R.; Herrera, F. SMOTE-RSB*: A hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory. *Knowl. Inf. Syst.* **2012**, *11*, 245–265. doi:10.1007/s10115-011-0465-6.
39. Lei, X.; Zhou, P. An intrusion detection model based on GSSVM Classifier. *Inf. Technol. J.* **2012**, *11*, 794–798. doi:10.3923/itj.2012.794.798.
40. Gonzalez-Cuautle, D.; Corral-Salinas, U.Y.; Sanchez-Perez, G.; Perez-Meana, H.; Toscano-Medina K.; Hernandez-Suarez, A. An Efficient Botnet Detection Methodology using Hyper-Parameter Optimization Trough Grid-Search Techniques. In Proceedings of the 2019 7th International Workshop on Biometrics and Forensics (IWBF), Cancun, Mexico, 2–3 May 2019; pp. 1–6.
41. Abdulhammed, R.; Faezipour, M.; Abuzneid, A.; AbuMallouh, A. Deep and Machine Learning Approaches for Anomaly-Based Intrusion Detection of Imbalanced Network Traffic. *IEEE Sens. Lett.* **2019**, *3*, 1–4. doi:10.1109/LENS.2018.2879990.
42. Putman, C.G.J.; Nieuwenhuis, L.J. Business Model of a Botnet. In Proceedings of the 2018 26th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Cambridge, UK, 21–23 March 2018; pp. 441–445.
43. Beigi, E.B.; Jazi, H.H.; Stakhanova, N.; Ghorbani, A.A. Towards effective feature selection in machine learning-based botnet detection approaches. In Proceedings of the Communications and Network Security (CNS), 2014 IEEE Conference, San Francisco, CA, USA, 29–31 October 2014; pp. 247–255.
44. Ring, M.; Wunderlich, S.; Gruedl, D.; Landes, D.; Hotho, A. Creation of Flow-Based Data Sets for Intrusion Detection. *J. Inf. Warf.* **2017**, *16*, 40–53.
45. Howley, T.; Madden, M.G.; O'Connell, M.L.; Ryder, A.G. The effect of principal component analysis on machine learning accuracy with high dimensional spectral. In Proceedings of the International Conference on Innovative Techniques and Applications of Artificial Intelligence Data, Cambridge, UK, 12–14 December 2005; pp. 209–222.

46. Zhao, D.; Traore, I.; Sayed, B.; Lu, W.; Saad, S.; Ghorbani, A.; Garant, D. Botnet detection based on traffic behavior analysis and flow intervals. *Comput. Secur.* **2013**, *39*, 2–16. doi:10.1016/j.cose.2013.04.007.
47. HoneyNet. Available online: <https://www.honeynet.org/> (accessed on 15 May 2019).
48. Szabó, G.; Orincsay, D.; Malomsoky, S.; Szabó, I. On the validation of traffic classification algorithms. In Proceedings of the International Conference on Passive and Active Network Measurement, Berlin, Germany, 26–27 March 2018; pp. 72–81.
49. Lawrence Berkeley National Laboratory and icsi, Ibln/icsi Enterprise Tracing Project. Ibln Enterprise Trace Repository. 2005. Available online: <http://www.icir.org/enterprise-tracing/> (accessed on 15 May 2019).
50. Shiravi, A.; Shiravi, H.; Tavallaee, M.; Ghorbani, A. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Comput. Secur.* **2012**, *31*, 357–374. doi:10.1016/j.cose.2011.12.012.
51. Malware Capture Facility Project. Available online: <https://mcfp.weebly.com/> (accessed on 15 May 2019).
52. Tshark. Available online: <https://www.wireshark.org/docs/man-pages/tshark.html> (accessed on 10 May 2019).
53. Marnerides, A.K.; Watson, M.R.; Shirazi, N.; Mauthe, A.; Hutchison, D. Malware analysis in cloud computing: Network and system characteristics. In Proceedings of the 2013 IEEE Globecom Workshops (GC Wkshps), Atlanta, GA, USA, 9–14 December 2013; pp. 482–487.
54. Watson, M.R.; Marnerides, A.K.; Mauthe, A.; Hutchison, D. Malware detection in cloud computing infrastructures. *IEEE Trans. Dependable Secur. Comput.* **2015**, *13*, 192–205. doi:10.1109/TDSC.2015.2457918.
55. Marnerides, A.K.; Mauthe, A.U. Analysis and characterisation of botnet scan traffic. In Proceedings of the 2016 International Conference on Computing, Networking and Communications (ICNC), Kauai, Hawaii, USA, 15–18 February 2016; pp. 1–7.
56. Venkatesh, G.K.; Nadarajan, R.A. HTTP botnet detection using adaptive learning rate multilayer feed-forward neural network. In Proceedings of the IFIP International Workshop on Information Security Theory and Practice, Egham, UK, 20–22 June 2012; pp. 38–48.
57. Su, S.C.; Chen, Y.R.; Tsai, S.C.; Lin, Y.B. Detecting p2p botnet in software defined networks. *Secur. Commun. Netw.* **2018**, *2018*, 13. doi:10.1155/2018/4723862.
58. Rice, J.R. *The Algorithm Selection Problem*; Advances in Computers; Elsevier: Amsterdam, NL, USA, 1976; Volume 15, pp. 65–118.
59. Liao, Y.; Vemuri, V.R. Use of k-nearest neighbor classifier for intrusion detection. *Comput. Secur.* **2002**, *21*, 439–448. doi:10.1016/S0167-4048(02)00514-X.
60. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. doi:10.1007/BF00994018.
61. Caesarendra, W.; Widodo, A.; Yang, B.S. Application of relevance vector machine and logistic regression for machine degradation assessment. *Mech. Syst. Signal. Process.* **2010**, *24*, 1161–1171. doi:10.1016/j.ymssp.2009.10.011.
62. Rokach, L.; Maimon, O.Z. *Data Mining With Decision Trees: Theory and Applications*; World Scientific: Singapore, 2018; Volume 69.
63. Santos, I.; Brezo, F.; Ugarte-Pedrero, X.; Bringas, P.G. Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Inf. Sci.* **2013**, *231*, 64–82. doi:10.1016/j.ins.2011.08.020.
64. Aviv, A.J.; Haeberlen, A. Challenges in experimenting with botnet detection systems. In Proceedings of the 4th Conference on Cyber Security Experimentation and Test (CSET), San Francisco, CA, USA, 8–12 August 2011; p. 6.
65. Amos, B.; Turner, H.; White, J. Applying machine learning classifiers to dynamic android malware detection at scale. In Proceedings of the 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC), Sardinia, Italy, 1–5 July 2013; pp. 1666–1671.
66. Verma, A.; Ranga, V. Statistical analysis of CIDDs-001 dataset for network intrusion detection systems using distance-based machine learning. *Procedia Comput. Sci.* **2018**, *125*, 709–716. doi:10.1016/j.procs.2017.12.091.

67. Bijalwan, A.; Chand, N.; Pilli, E.S.; Krishna, C.R. Botnet analysis using ensemble classifier. *Perspect. Sci.* **2016**, *8*, 502–504. doi:10.1016/j.pisc.2016.05.008.
68. Thamilarasu, G.; Chawla, S. Towards Deep-Learning-Driven Intrusion Detection for the Internet of Things. *Sensors* **2019**, *19*, 1977. doi:10.3390/s19091977.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).