future internet

MDPI

Article

Ensemble Classifiers for Network Intrusion Detection Using a Novel Network Attack Dataset †

Ahmed Mahfouz [1,*], Abdullah Abuhussein [2], Deepak Venugopal [1] and Sajjan Shiva [1]

[1] Department of Computer Science, University of Memphis, Memphis, TN 38152, USA; dvngopal@memphis.edu (D.V.); sshiva@memphis.edu (S.S.)
[2] Department of Information Systems, St. Cloud State University, St. Cloud, MN 56301, USA; aabuhussein@stcloudstate.edu
* Correspondence: amahfouz@memphis.edu
† This paper is an extension of Comparative Analysis of ML Classifiers for Network Intrusion Detection, originally presented at the Fourth International Congress on Information and Communication Technology.

Received: 5 October 2020; Accepted: 20 October 2020; Published: 26 October 2020

Abstract: Due to the extensive use of computer networks, new risks have arisen, and improving the speed and accuracy of security mechanisms has become a critical need. Although new security tools have been developed, the fast growth of malicious activities continues to be a pressing issue that creates severe threats to network security. Classical security tools such as firewalls are used as a first-line defense against security problems. However, firewalls do not entirely or perfectly eliminate intrusions. Thus, network administrators rely heavily on intrusion detection systems (IDSs) to detect such network intrusion activities. Machine learning (ML) is a practical approach to intrusion detection that, based on data, learns how to differentiate between abnormal and regular traffic. This paper provides a comprehensive analysis of some existing ML classifiers for identifying intrusions in network traffic. It also produces a new reliable dataset called GTCS (Game Theory and Cyber Security) that matches real-world criteria and can be used to assess the performance of the ML classifiers in a detailed experimental evaluation. Finally, the paper proposes an ensemble and adaptive classifier model composed of multiple classifiers with different learning paradigms to address the issue of the accuracy and false alarm rate in IDSs. Our classifiers show high precision and recall rates and use a comprehensive set of features compared to previous work.

Keywords: IDS; ensemble classifier; intrusion detection; ML; GTCS dataset

1. Introduction

Cyberattacks have become more widespread as intruders take advantage of system vulnerabilities for intellectual property theft, financial gain, or even the destruction of entire network infrastructures [1]. In the past few months, the Federal Bureau of Investigation released a high-impact cybersecurity warning in response to the increasing number of attacks on government targets. Government officials have warned major cities that such hacks are a disturbing trend that is likely to continue. The window for detecting a security breach can be measured in days, as attackers are aware of existing security controls and are continually improving their attacks. In many cases, a security breach is inevitable, which makes early detection and mitigation the best plan for surviving an attack.

Security professionals use different prevention and detection techniques to reduce the risk of security breaches. Prevention techniques such as applying complex configurations and establishing f Abdullah Abuhusseina strong security policy aim to make it more difficult to carry out such attacks. All security policies should maintain the three principles of the Central Intelligence Agency triad: confidentiality, integrity, and availability.

Detection techniques are either signature-based or anomaly-based. Classical security solutions such as virus scanners, intrusion detection systems (IDSs), and firewalls depend on the signature-based approach, which compares a hash of the payload to a database of known malicious signatures [1]. These techniques provide a strong defense against known attacks but fail to detect zero-day attacks. Moreover, they are by no means an adequate safeguard against skilled attackers using the latest attack techniques, who can easily bypass such security controls.

Anomaly detection techniques look for abnormal activities, including those that have not occurred before, since they view any unusual event as a potential attack. False positives can thus occur when routine activities are detected as irregular [2]. Anomaly detection requires a system trained on a model of normal system behavior.

Commercial network security appliances mainly perform misuse detection [3]. They use knowledge of previous attacks to create signatures which can precisely identify new instances of such attacks but cannot be used to identify novel attacks. A complementary anomaly detection approach can identify novel attacks, but at the expense of falsely identifying unusual activity as malicious. These limitations result in a significant number of attacks being missed, leading to the theft of intellectual property and other information from vulnerable organizations. Meanwhile, machine learning (ML) has the potential to overcome some limitations of IDSs [4]. Creating these features (in a process called feature engineering) is therefore a critical step which places bounds on the detector's capabilities. However, feature engineering is often an ad hoc process that uses trial and error to determine which traffic features are most relevant to the detection problem [5]. Such a process requires domain knowledge and is time-consuming when done iteratively. These difficulties in feature engineering have inhibited the application of ML to network security.

Moreover, due to the dearth of sufficient datasets, ML approaches for network intrusion detection suffer from a lack of accurate deployment, analysis, and evaluation. Although researchers are using some datasets—such as DARPA [6], KDD CUP 99 [7], NSL-KDD [8], and CAIDA [9]—to evaluate the performance of their proposed IDS approaches, most of those datasets are out-of-date and thus inaccurate. They lack traffic diversity, do not cover different types of attacks, and do not reflect current network traffic trends [10].

This paper offers the following contributions:

- It presents a newly generated IDS dataset called GTCS (Game Theory and Cyber Security) that overcomes most shortcomings of existing datasets and covers most of the necessary criteria for common updated attacks, such as botnet, brute force, distributed denial of service (DDoS), and infiltration attacks. The generated dataset is completely labeled, and about 84 network traffic features have been extracted and calculated for all benign and intrusive flows.
- It analyzes the GTCS dataset using Weka—an open source software which provides tools for data preprocessing and the implementation of several ML algorithms—to select the best feature sets to detect different attacks and provides a comprehensive analysis of six of the most well-known ML classifiers to identify intrusions in network traffic. Specifically, it analyzes the classifiers in terms of accuracy, true positive rates, and false positive rates.
- It proposes an adaptive ensemble learning model that integrates the advantages of different ML classifiers for different types of attacks to achieve optimal results through ensemble learning. The advantage of ensemble learning is its ability to combine the predictions of several base estimators to improve generalizability and robustness over a single estimator.

2. Background

In this section, we introduce the concepts and technologies on which our problem and methodology are based to illuminate the motivation for this work. This includes: (1) a brief overview of IDSs and their limitations; (2) ML, hyperparameter optimization, and ensemble learning; and (3) datasets, problems related to evaluative datasets, and data processing considerations.

2.1. Intrusion Detection Systems

An intrusion is a malicious activity that aims to compromise the confidentiality, integrity, and availability of network components in an attempt to disrupt the security policy of a network [11]. The National Institute of Standards and Technology (NIST) defines the intrusion detection process as "the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network" [12]. An IDS is a tool that scans network traffic for any harmful activity or policy breaches. It is a system for monitoring network traffic for malicious activities and alerting network administrators to such abnormal activities [13]. IDSs achieve this by gathering data from several systems and network sources and analyzing those data for possible threats.

Unlike firewalls, which are used at the perimeter of the network and play the role of gatekeeper by monitoring incoming network traffic and determining whether it can be allowed into the network or endpoint at all, IDSs monitor internal network traffic and mark suspicious and malicious activities. Consequently, an IDS can identify not only attacks that pass the firewall but also attacks that originate from within the network.

2.1.1. Limitations of Intrusion Detection Systems

Although IDSs are considered a key component of computer network security, they have some limitations that should be noted before deploying intrusion detection products [14]. Some of these limitations include:

- Most IDSs generate a high false positive rate, which wastes the time of network administrators and in some cases causes damaging automated responses.
- Although most IDSs are marketed as real-time systems, it may in fact take them some time to automatically report an attack.
- IDSs' automated responses are sometimes inefficient against advanced attacks.
- Many IDSs lack user-friendly interfaces that allow users to operate them.
- To obtain the maximum benefits from the deployed IDS, a skilled IT security staff should exist to monitor IDS operations and respond as needed.
- Numerous IDSs are not failsafe, as they may not be well protected from attacks or destruction.

2.1.2. Why Do We Need Machine Learning?

There are a number of reasons for researchers to consider the use of ML in network intrusion detection. One such reason is the ability of ML to find similarities within a large amount of data. The main assumption of ML-based approaches is that an intrusion creates distinguishable patterns within the network traffic and that these patterns can be efficiently detected using ML approaches [15]. These approaches promise automated data-driven detection that infers information about malicious network traffic from the vast amount of available traffic traces. Furthermore, ML can be used to discover anomalies in data without the need for prior knowledge about those data. By combining the characteristics of ML techniques and capable computing units, we can create a powerful weapon with which to respond to network intrusion threats. Moreover, we have a great deal of data nowadays, but human expertise is limited and expensive. Therefore, using ML, we can automatically discover patterns which humans may not be able to recognize due to the scale of the data. In the absence of ML, human experts would need to define manually crafted rules, which would not be scalable.

2.2. Machine Learning

ML is the field of computer science that studies algorithms that automatically improve during training and experience, enabling a computer to make accurate predictions when fed data without

being explicitly programmed [16]. ML algorithms use a subset of a larger dataset, known as training data or sample data, to build a mathematical model to make predictions or decisions based on a given problem. ML is applied in a wide variety of domains—including spam filtering, Internet search engines, recommendation systems, voice recognition, and computer vision—where conventional algorithms cannot accomplish the required tasks. Similarly, in the field of cybersecurity, several ML methods have been proposed to monitor and analyze network traffic to recognize different anomalies. Most of these methods identify anomalies by looking for deviations from a basic regular traffic model. Usually, these models are trained on a set of attack-free traffic data that are collected over a long time period.

2.2.1. Hyperparameter Optimization

Model parameters are the set of configuration variables that are internal to a model and can be learned from the historical training data. The value of these parameters is estimated from the input data. Model hyperparameters are the set of configuration variables external to the model. They are the properties that govern the entire training process and cannot be directly learned from the input data. A model's parameters specify how input data are transformed into the desired output, while the hyperparameters define the structure of the model.

Hyperparameter optimization (also known as hyperparameter tuning) is the process of finding the optimal hyperparameters for a learning algorithm in ML. A set of different measures for a single ML model can be used to generalize different data patterns. This set is known as the hyperparameters set, which should be highly optimized so that the ML model can solve the assigned problem as optimally as possible. The optimization process locates the hyperparameters tuple and produces a model that minimizes the predefined loss function for the given data. The objective function takes the hyperparameters tuple and returns the associated loss [17]. The generalization performance is often estimated using cross-validation [18]. Hyperparameter optimization techniques typically use one of several optimization algorithms: grid search, random search, or Bayesian optimization.

2.2.2. Ensemble Learning

Ensemble approaches utilize collections of ML algorithms to produce higher predictive performance than could be obtained from a single ML classifier [19]. The main idea of an ensemble approach is to combine several ML algorithms to exploit the strengths of each employed algorithm to obtain a more powerful classifier. Ensemble approaches are particularly helpful if the problem can be split into subproblems such that each subproblem can be assigned to one module of the ensemble. Depending on the structure of the ensemble approach, each module can include one or more ML algorithms. In the world of network attacks, since the signatures of different attacks are quite distinct from one another, it is normal to have different sets of features as well as different ML algorithms to detect different types of attacks. It is thus obvious that a single IDS cannot cover all types of input data or identify different types of attacks [20,21]. Many researchers have shown that classification problems can be solved with high accuracy when using ensemble models instead of single classifiers [22].

Ensemble models can be thought of as using various approaches to solve a specific problem. This resembles the process according to which patients diagnosed with a dangerous medical condition, such as a tumor, usually see more than one doctor to solicit different opinions on their case. This is a kind of cross-validation that increases the probability of receiving an accurate diagnosis. Likewise, the outputs of various ML classifiers in an intrusion detection problem can be combined to enhance the accuracy of the overall IDS. The main challenge in using ensemble approaches is how to choose the best set of classifiers to constitute the ensemble model and which decision function to use to combine the results of those algorithms [23]. Bagging, boosting, and stacking are the main three methods used to combine ML algorithms in ensemble models [24].

2.3. Datasets

One of the main challenges that researchers face in the area of intrusion detection is finding a suitable dataset that they can use to train and evaluate their proposed IDSs [25]. Although there exists a group of datasets used by researchers to train, test, and evaluate their IDS approaches, as mentioned in Section 1, most of those datasets are outdated, suffer from a lack of attack diversity, and do not reflect current trends and traffic variety. In this section, we first discuss general security issues related to finding a suitable dataset for evaluating IDSs. Next, we analyze and evaluate some of the most widely used datasets and identify deficiencies that indicate the need for a comprehensive and reliable dataset.

2.3.1. Problems with Evaluative Datasets

Researchers have reported many security issues regarding the datasets used to evaluate IDSs [26,27]. Some of these issues are as follows:

- Data privacy issues and security policies may prevent corporate entities from sharing realistic data with users and the research community.
- Getting permission from a dataset's owner is frequently delayed. Moreover, it usually requires the researcher to agree to an acceptable use policy (AUP) that includes limitations on the time of usage and the data that can be published about the dataset.
- The limited scope of most datasets does not fit various network intrusion detection researchers' aims and objectives.
- Most of the available datasets in the IDS field suffer from a lack of proper documentation describing the network environment, simulated attacks, and dataset limitations.
- Many of the accessible datasets were labeled manually.

To overcome these problems, we present a new dataset that is automatically and completely labeled. The dataset and its proper documentation are publicly available to researchers and do not require any AUP to use.

2.3.2. Data Preprocessing

Data preprocessing is the first step of the process of collecting data before they are used by any ML model. It is generally used to transform the raw data into a structure that the ML model can handle and also helps improve the quality of the model.

Normalization

Normalization is a preprocessing method that is usually applied as part of data preparation for ML models. The goal of normalization is to adjust the values of numeric data in a dataset to a range (usually 0–1) using a common scale without distorting differences in the ranges of actual values or losing information. If a dataset contains a column with values ranging from 0 to 1 and a second column with values ranging from 100,000 to 1,000,000, the enormous variance in scale among the numbers in the two columns could cause problems when attempting to combine the values as features during modeling. Normalization can help bypass these problems by creating new values that maintain the general distribution and ratios of the source data while keeping values within a scale applied across all numeric columns used in the model.

One-Hot Encoding

Most ML models require all input and output data to be in the form of numeric values. This means that if the dataset includes categorical data, it must be encoded as numbers before it can be used by ML models. One-hot encoding is the process of converting categorical data into a form that ML algorithms can use to improve prediction [28,29].

Feature Selection

Feature selection (also called attribute selection) is the process of selecting attributes of the dataset that are most relevant to the predictive modeling problem on which one is working [30]. The definition of relevance varies from method to method. Based on its understanding of significance, a feature selection technique mathematically formulates a criterion to evaluate a set of features generated by a scheme that searches over the feature space. [31] defined two degrees of relevance: strong and weak. A feature s is strongly relevant if its removal deteriorates the performance of a classifier. A feature s is weakly relevant if it is not strongly relevant and the removal of a subset of features containing s deteriorates the performance of the classifier. A feature is irrelevant if it is neither strongly nor weakly relevant. Feature selection techniques assist in creating an accurate predictive model by choosing features that will provide better accuracy and less complexity while requiring fewer data.

Feature selection techniques are generally divided into three categories: filter, wrapper, and embedded [32]. The filter method operates without engaging any information about the induction algorithm. Using some prior knowledge—for example, that a feature should be strongly correlated with the target class or that features should be uncorrelated with one another—the filter method selects the best subset of features by measuring the statistical properties of the subset to be evaluated. Alternatively, the wrapper method employs a predetermined induction algorithm to find a subset of features with the highest evaluation by searching through the space of feature subsets and evaluating the quality of selected features. The process of feature selection "wraps around" an induction algorithm. Since the wrapper approach includes a specific induction algorithm to optimize feature selection, it often provides a better classification accuracy result than the filter approach. However, the wrapper method is more time-consuming than the filter method because it is strongly coupled with an induction algorithm and repeatedly calls the algorithm to evaluate the performance of each subset of features. It thus becomes impractical to apply a wrapper method to select features from a large dataset that contains numerous features and instances [33]. Furthermore, the wrapper approach is required to re-execute its induction algorithm to select features from a dataset while the algorithm is replaced with a dissimilar one. Some researchers [34–36] have used a hybrid feature selection method. In supervised ML, an induction algorithm is typically presented with a set of training instances wherein each instance is described by a vector of feature (or attribute) values and a class label. For example, in medical diagnosis problems, features might include a patient's age, weight, and blood pressure, and the class label might indicate whether a physician had determined that the patient was suffering from heart disease. The task of the induction algorithm is to induce a classifier that will be useful in classifying future cases. The classifier is a mapping from the space of feature values to the set of class values. More information about the wrapper methods and inductive algorithm can be found in [37]. Finally, the embedded approaches include feature selection in the training process, thus reducing computational costs due to the classification process needed for each subset [38].

Dataset Imbalance

Imbalanced datasets are a common problem in ML classification, where there exists a disproportionate ratio of examples in each class. This problem is especially relevant in the context of supervised ML, which involves two or more classes. The imbalance means that the distribution of the dataset examples across the recognized classes is biased or skewed. This distribution may vary from a trivial bias to a significant imbalance where there are only a few examples in the minority class and many examples in the majority class or classes.

An imbalanced dataset poses a challenge for predictive modeling, since the majority of ML classification algorithms are designed based on the hypothesis that there are almost an equal number of examples for each class. This results in an inaccurate prediction model with poor performance, particularly for the minority class. Therefore, a balanced dataset is essential for creating a good prediction model [39].

Real-world data are usually imbalanced, which may be one of the main causes of the decrease in ML algorithms' generalization. If the imbalance is heavy, it will be difficult to develop efficient

classifiers with conventional learning algorithms. In many domains, the cost of misclassifying minority classes is greater than for the majority class for many class-imbalanced datasets. This is particularly true in the IDS domain, where malicious traffic tends to be the minority class. Consequently, there is a need for sampling methods that can handle imbalanced datasets.

Sampling techniques can be used to overcome the dataset imbalance dilemma by either excluding some data from the majority class (known as under-sampling) or by adding artificially generated data to the minority class (known as oversampling) [40]. Oversampling methods boost the number of samples in the minority class in the training dataset. The main advantage of such methods is that there will be no loss in the data from the primary training dataset, as all samples from the minority and majority classes are kept. However, there is also the drawback that the scope of the training dataset is significantly increased. Arbitrary oversampling is the simplest oversampling method, in which randomly chosen samples from the minority class are duplicated and combined with the new dataset [41]. Synthetic minority oversampling technique (SMOTE) is another oversampling method proposed by Chawla [42] wherein synthetic data are created and added to the minority class rather than simply duplicating the examples. SMOTE blindly generates synthetic data without studying the majority class data, which may lead to an overgeneralization problem [43]. Under-sampling methods, on the other hand, are used to decrease the number of examples in the majority class. They reduce the size of the majority class data to balance the class distribution in the dataset. Random under-sampling is an example of an under-sampling method that randomly selects a subset of majority class samples and merges them with a minority class sample, generating a new balanced dataset [44]. However, under-sampling a dataset by reducing the majority class results in a loss of data and overly general rules.

3. Literature Survey and Related Work

Anomaly-based IDS was first introduced by Anderson in 1980 [45]. Since then, the topic of anomaly detection has been the subject of many surveys and review articles. Various researchers have applied ML algorithms and used different publicly available datasets for their research in order to achieve better detection results [46]. Hodo et al. [47] reviewed ML algorithms and their performance in terms of anomaly detection and discussed and explained the role of feature selection in ML-based IDSs. Chandola et al. [48] presented a structured review of the research on anomaly detection in different research areas and application fields, including network IDSs. G. Meera Gandhi [49] used the DARPA-Lincoln dataset to evaluate and compare the performance of four supervised ML classifiers in detecting four categories of attacks: denial of service (DoS), remote-to-local, probe, and user-to-root. Their results showed that the J48 classifier outperformed the other three classifiers (IBK, MLP, and naïve Bayes [NB]) in prediction accuracy. In [50], Nguyen et al. conducted an empirical study to evaluate a comprehensive set of ML classifiers on the KDD99 dataset to detect attacks from the four attack classes. Abdeljalil et al. [51] tested the performance of three ML classifiers—J48, NN, and support vector machine (SVM)—using the KDD99 dataset and found that the J48 algorithm outperformed the other two algorithms. L. Dhanabal et al. [46] analyzed the NSL-KDD dataset and and used it to measure the effectiveness of ML classifiers in detecting anomalies in network traffic patterns. In their experiment, 20% of the NSL-KDD dataset was used to compare the accuracy of three classifiers. Their results showed that when correlation-based feature selection was used for dimensionality reduction, J48 outperformed SVM and NB in terms of accuracy. In [52], Belavagi et al. checked the performance of four supervised ML classifiers—SVM, RF (random forest), LR (linear regression), and NB—on intrusion detection over the NSL-KDD dataset. The results showed that the RF classifier had 99% accuracy.

In terms of feature selection, Hota et al. [53] utilized different feature selection techniques to remove irrelevant features in a proposed IDS model. Their experiment indicated that the highest accuracy result could be achieved with only 17 features from the NSL-KDD dataset by using the C4.5 algorithm along with information gain. In [54], Khammassi et al. applied a wrapper approach based on a genetic algorithm as a search strategy and logistic regression as a learning algorithm to select the best subset of features of the KDD99 and UNSW-NB15 datasets. They used three different DT

classifiers to measure and compare the performance of the selected subsets of features. Their results showed that they could achieve a high detection rate with only 18 features for KDD CUP 99 and 20 features for UNSW-NB15. Abdullah et al. [55] also proposed an IDS framework with selection of features within the NSL-KDD dataset that were based on dividing the input dataset into different subsets and combining them using the information gain filter. The optimal set of features was then generated by adding the list of features obtained for each attack. Their experimental results showed that, with fewer features, the researchers could improve system accuracy while decreasing complexity.

Besides selecting the relevant features that can represent intrusion patterns, the choice of ML classifier can also lead to better accuracy. Moreover, the literature suggests that assembling multiple classifiers can reduce false positives and produce more accurate classification results than single classifiers [56]. Gaikwad et al. [57] used REPTree as a base classifier and proposed a bagging ensemble method that provides higher classification accuracy and lowers false positives for the NSL-KDD dataset. Jabbar et al. [58] suggested a cluster-based ensemble IDS model based on the ADTree and KNN algorithms. The experimental results showed that their model outperformed most existing classifiers in accuracy and detection rates. Similarly, Paulauskas et al. [59] used an ensemble model of four different base classifiers to build a stronger learner and showed that the ensemble model produced more accurate results for an IDS.

Generally speaking, previous studies have mainly focused on comparing different ML algorithms and selecting those with the best accuracy results to improve the overall detection effect. The main optimization methods are feature selection and ensemble learning. However, there is still room to improve the results of these studies. Unlike the above studies, this paper concentrates on evaluating and comparing the performance of a group of well-known supervised ML classifiers over the full NSL-KDD dataset for intrusion detection along the following dimensions: feature selection, sensitivity to hyperparameter tuning, and class imbalance. Moreover, most of the datasets used by researchers to evaluate the performance of their proposed intrusion detection approaches are out-of-date and unreliable. Some of these datasets suffer from a lack of traffic diversity and volume or do not cover a variety of attacks, while others anonymize packet information and payload—which cannot reflect current trends—or lack feature sets and metadata. This paper produces a reliable dataset that contains benign and four common attack network flows, which meets real-world criteria. This study evaluates the performance of a comprehensive set of network traffic features and ML algorithms to indicate the best set of features for detecting certain attack categories.

4. GTCS Dataset Collection

In this section, we describe: (1) how our novel dataset was collected and stored, (2) the dataset collection testbed implementation and key design decisions, and (3) dataset features selection and extraction. Finally, we provide a dataset statistical summary to show the quality of the data collected.

4.1. Lab Setup

To generate the new dataset, benign network traffic along with malicious traffic were extracted, labeled, and stored. To mimic typical network traffic flow, we designed a complete testbed (Figure 1) composed of several normal and attacking virtual machines (VMs) that were distributed between two separate networks. The victim network consisted of a set of VMs running different versions of the most common operating systems, namely Windows Server and/or PC, Linux, and Android. The attack network was a completely separate infrastructure containing Kali 1.1 and Kali 2.0 VMs.

To generate a large amount of realistic benign traffic, we used Ostinato [60], a flexible packet generator tool that generates normal traffic with given IPs and ports. Malicious traffic was generated using Kali Linux [61], an enterprise-ready security-auditing Linux distribution based on Debian GNU/Linux [62]. The process of generating both benign and malicious traffic took eight days. All of the attacks considered for the experiments in this paper were new. In our attack scenarios, four of the most common and up-to-date attack families were considered and are briefly described below.

- Botnet attacks [63]: This attack type can be defined as a group of compromised network systems and devices that execute different harmful network attacks, such as sending spam, granting backdoor access to compromised systems, stealing information via keyloggers, performing phishing attacks, and so on.
- Brute force attacks [64]: This refers to a well-known network attack family in which intruders try every key combination in an attempt to guess passwords or use fuzzing methods to obtain unauthorized access to certain hidden webpages (e.g., an admin login page).
- DDoS attacks [65]: DoS attacks are a very popular type of network attack in which an attacker sends an overwhelming number of false requests to a target service or network in order to prevent legitimate users from accessing that service. DDoS attacks are a modern form of DoS attack wherein attackers use thousands of compromised systems to flood the bandwidth or resources of the target system.
- Infiltration attacks [66]: These attacks are usually executed from inside the compromised system by exploiting vulnerabilities in software applications such as Internet browsers, Adobe Acrobat Reader, and the like.

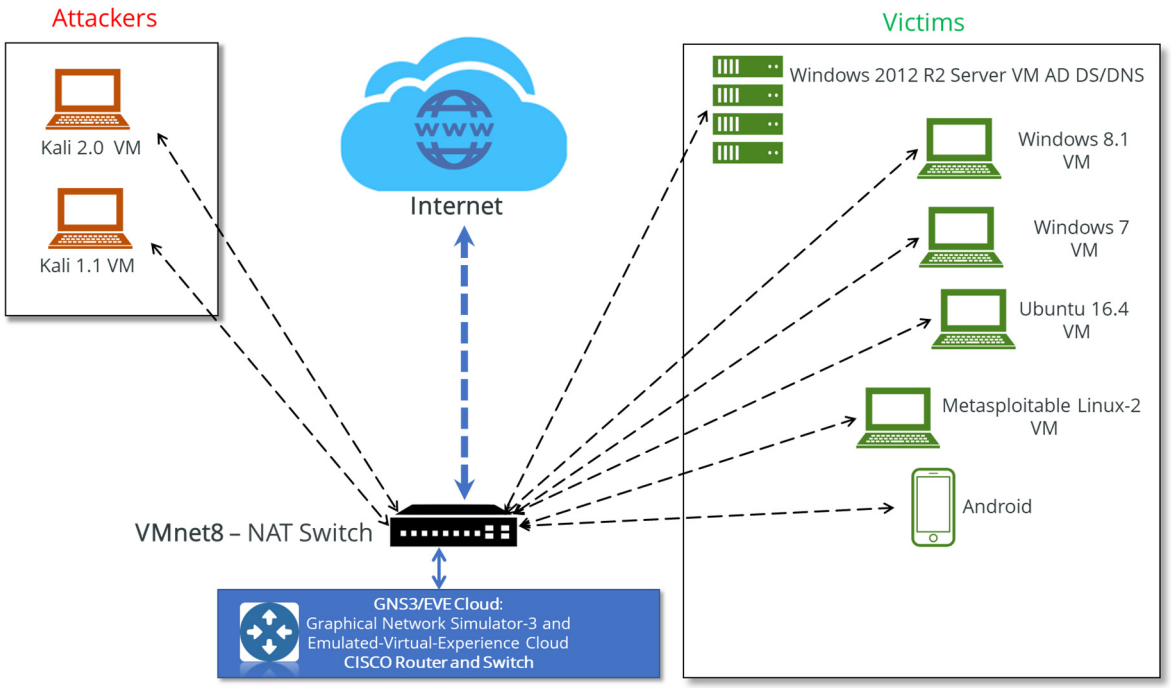

Figure 1. Testbed of GTCS dataset collection system.

4.2. Data Collection and Feature Extraction

We implemented four attack scenarios: botnet, brute force, DDoS, and infiltration. For each attack, we defined a scenario based on the implemented network topology and executed the attacks using the Kali Linux machines that were located in a separate network from the target machines. The attacking machines were Kali 1.1 and Kali 2.0.

To perform the botnet attack, we used Zbot [67], a trojan horse malware package that can be run on Microsoft Windows operating systems to execute many malicious tasks. Zbot can mainly be extended through drive-by downloads and different phishing schemes. Because it uses stealth techniques to hide from different security tools, Zbot has become the largest botnet on the Internet [68]. The victim machines in our botnet attack scenario were running Windows 7 and Windows 8. It is important to note that firewalls, Windows Defender, and automatic updating were all disabled on all Windows victim machines to enable a wide spectrum of interesting cases to be captured. The Graphical

Network Simulator-3 (GNS3) and Emulated Virtual Experience (EVE) cloud components are network software emulators that allow a combination of virtual and real devices to simulate complex networks. In addition, password complexity checks were not active, and all passwords were set to a minimum of three characters. For the brute force attack, we used the FTP module on the Kali 2.0 Linux machine to attack the machine in the victim network running Ubuntu 16.4. To carry out the DDoS attack, we used the High Orbit Ion Cannon (HOIC) tool [69], a popular free tool for performing DDoS attacks. HOIC works by flooding a target web server with junk HTTP, GET, and POST requests and can open up to 256 simultaneous attack sessions at once. For the infiltration attack, we sent a vulnerable application to the Metasploitable 2 Linux machine on the victim network. We then used the vulnerable application to open a backdoor and perform our infiltration attack.

To capture the raw network traffic data (in pcap format), we used Wireshark and tcpdump [70,71]. Wireshark is a network packet analyzer that can capture network traffic data in as much detail as possible, while tcpdump is a command line utility that helps capture and analyze network traffic. After collecting the raw network packets (i.e., pcap files), we used CICFlowMeter [72,73] to process those files and extract the features of the network flow packets. CICFlowMeter is an open source tool that generates Bi-flows from pcap files and extracts features from these flows. The extracted features are shown in Table 1. The full dataset is available on ResearchGate for researchers and practitioners [74].

Table 1. GTCS Dataset extracted features.

Feature Name	Description
Flow ID	ID which presents the unique ID calculated from the 5-tuple, i.e., Src IP, Dst IP, Src Port, Dst Port, and protocol number
Src IP	IP address of the machine from which the traffic started
Src Port	IP address of destination machine
Dst IP	IP address of destination machine
Dest Port	Destination port number
Protocol number	Number of the transaction protocol
Timestamp	Date and time of day when the flow occurred
Flow Duration	Total duration of flow
Tot Fwd/Bwd Pkts	Total packets in forward/backward direction
TotLen Fwd/Bwd Pkts	Total size of packet in forward/backward direction
Fwd/Bwd Pkt Len	Size of packet in forward/backward direction
Flow Byts/Packets	Rate of flow of bytes/packets (i.e., number of bytes/packets transferred each second)
Flow IAT	Time between two packets sent in forward/backward direction
Fwd/Bwd IAT	Time between two packets sent in forward/backward direction
Fwd/Bwd PSH	Number of times that PSH flag was set to packets moving in either forward/backward direction
Fwd/Bwd URG	Number of times that URG flag was set to packets moving in either forward/backward direction
Active	How long a flow was active before becoming idle
Idle	How long a flow was idle before becoming active
Fwd/Bwd Pkts/s	Number of transmitted packets per second in forward/backward direction
Fwd/Bwd Byts/s	Number of transmitted bytes per second in forward/backward direction
Label	Indicates whether traffic is malicious

4.3. Statistical Summary of GTCS Dataset

Each record in the GTCS dataset reveals different features of the traffic with 83 attributes plus an assigned label classifying each record as either normal or an attack. The attack types in the dataset can be grouped into four main classes (botnet, brute force, DDoS, and infiltration). The number of records associated with each class is shown in Table 2.

Table 2. Number of samples for normal and attack classes.

Class	Training Set	Occurrence Percentage
Normal	139,186	26.98%
Botnet	93,021	17.97%
Brute Force	83,858	16.20%
DDoS	131,211	25.35%
Infiltration	70,202	13.56%
Total	517,478	100.00%

5. Experiments

In this section, we present the experimental setup and results of comparing six ML classifiers from various classifier families in terms of classification accuracy, true positive rate (TPR), false positive rate (FPR), precision, recall, *F*-measure, and receiver operating characteristic (ROC) area. The selected classifiers included NB, logistic, multilayer perceptron (neural network), SMO (SVM), IBK (*k*–nearest neighbor), and J48 (decision tree). We compare the performance of the classifiers in identifying intrusions in network traffic using GTCS, the newly generated dataset. Next, we present a holistic approach for detecting network intrusions using an ensemble of the best-performing ML algorithms.

The experiment in this section was carried out using Weka on a PC with an Intel® CORE™ i5-8265U x64-based CPU running at 3.50 GHz with 16.0 GB RAM installed and a 64-bit Windows 10 OS.

5.1. Machine Learning Algorithm Performance Comparison

The experiment in this section had two phases. In the first phase, we compared the performance of the classifiers using the GTCS dataset with the full extracted. To evaluate the performance of the ML classifiers, we used precision, recall, and F1 score, which are the most common measures for evaluating the performance of anomaly detection models. Precision refers to the portion of relevant instances among the retrieved instances. Recall refers to the portion of relevant retrieved instances in the total number of relevant instances. F1 score is the harmonic mean of precision and recall. These three measures depend on the confusion matrix, where four possible situations can be defined, as shown in Table 3.

Table 3. Evaluation metrics.

Measure Metric	Explanation
True Positive Rate (TP)	Correctly classified anomalous instances as an anomaly
True Negative Rate (TN)	Correctly classified normal instances as normal
False Negative Rate (FN)	Wrongly classified anomalous instances as normal
False Positive Rate (FP)	Wrongly classified normal instances as an anomaly
Precision	$TP * (TP + FP)$
Recall	$TP * (TP + FN)$
F1 score	$(2 * Precision * Recall) (Precision + Recall)$

The results of this phase are summarized in Table 4.

In the second phase, we applied the InfoGainAttributeEval algorithm with Ranker [75] to reduce the dimensions of the dataset. InfoGainAttributeEval evaluates the worth of an attribute by measuring the information gain with respect to the class.

$$\text{InfoGain(Class,Attribute)} = \text{H(Class)} - \text{H(Class|Attribute)}$$

Tables 4 and 5 present a comprehensive comparison of the classifiers in terms of classification accuracy, precision, recall, TPR, FPR, *F*-measure, and ROC area. Table 6 presents the accuracy of each classifier in classifying different classes in the GTCS dataset in the two phases.

Table 4. Phase 1 examination results.

Classifier	Accuracy	TPR	FPR	Precision	Recall	*F*-Measure	ROC Area
NB	86.81%	0.869	0.116	0.912	0.864	0.885	0.921
Logistic	91.88%	0.936	0.097	0.923	0.931	0.924	0.930
MLP	92.90%	0.944	0.085	0.932	0.939	0.931	0.937
SMO	92.13%	0.937	0.093	0.925	0.932	0.926	0.917
IBK	94.23%	0.948	0.055	0.943	0.945	0.942	0.942
J48	94.29%	0.951	0.054	0.949	0.945	0.944	0.943

Table 5. Phase 2 examination results.

Classifier	Accuracy	TPR	FPR	Precision	Recall	*F*-Measure	ROC Area
NB	89.51%	0.881	0.109	0.942	0.894	0.915	0.948
Logistic	94.58%	0.948	0.091	0.953	0.961	0.954	0.966
MLP	95.60%	0.956	0.077	0.96	0.969	0.962	0.964
SMO	94.83%	0.949	0.086	0.955	0.962	0.956	0.944
IBK	96.93%	0.960	0.050	0.971	0.973	0.971	0.969
J48	96.99%	0.962	0.048	0.974	0.975	0.972	0.970

Table 6. Classifiers' accuracy detection for different classes of attacks.

Classifier	Class	Phase I	Phase II
NB	Normal	86.00%	86.90%
	Botnet	87.90%	88.80%
	Brute Force	80.00%	80.90%
	DDoS	83.80%	84.70%
	Infiltration	78.96%	79.86%
Logistic	Normal	92.50%	93.40%
	Botnet	92.80%	93.70%
	Brute Force	84.80%	85.70%
	DDoS	91.58%	92.48%
	Infiltration	81.30%	82.20%
MLP	Normal	93.60%	94.50%
	Botnet	93.50%	94.40%
	Brute Force	90.00%	90.90%
	DDoS	98.89%	99.79%
	Infiltration	82.30%	83.20%

Table 6. *Cont.*

Classifier	Class	Phase I	Phase II
SMO	Normal	92.80%	93.70%
	Botnet	93.60%	94.50%
	Brute Force	83.70%	84.60%
	DDoS	91.01%	92.89%
	Infiltration	72.45%	83.35%
IBK	Normal	95.50%	96.40%
	Botnet	95.60%	96.50%
	Brute Force	95.10%	96.00%
	DDoS	96.32%	97.22%
	Infiltration	79.67%	80.57%
J48	Normal	94.60%	95.10%
	Botnet	95.30%	96.20%
	Brute Force	87.70%	91.60%
	DDoS	95.53%	96.43%
	Infiltration	83.43%	84.33%

The ranker search method ranks the attributes according to their individual evaluations, after which it is possible to specify the number of attributes to retain. Using the InfoGainAttributeEval algorithm, we ranked the attributes of the GTCS dataset by their evaluations, which resulted in selecting 44 out of 84 features of the GTCS dataset. The final selected features are summarized and ranked in Figure 2. Next, we replicated the phase 1 experiment using the reduced and normalized GTCS dataset with those 44 features. The results of this phase are summarized in Table 3. Figure 3 compares the classification accuracy of each classifier in the two phases.

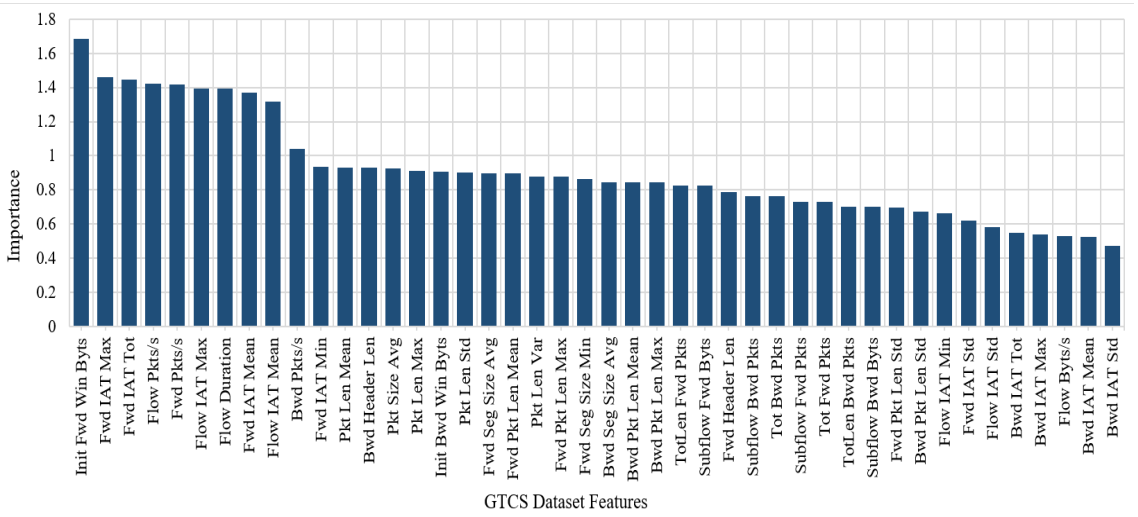

Figure 2. Selected features of GTCS dataset.

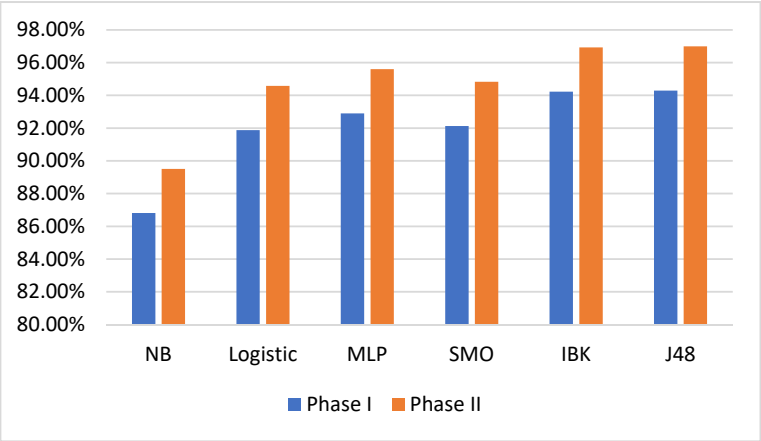

Figure 3. Classification accuracy in the two phases.

The experimental results show that IBK outperformed other classifiers and had the best accuracy in both phases. Moreover, the results shown in Table 6 indicate that IBK outperformed other classifiers in classifying normal, botnet, and brute force classes, while MLP and J48 performed best at classifying DDoS and infiltration classes, respectively. These results are illustrated in Figure 4.

5.2. A Holistic Approach for IDSs Using Ensemble ML Classifiers

In this section, we propose an ensemble classifier model (shown in Figure 5) composed of multiple classifiers with different learning paradigms to address the issue of the accuracy and false alarm rates in IDSs. The proposed model is composed of three ML classifiers from various classifier families. The selection of these classifiers is based on the results from the previous section. The selected classifiers are J48 (DT-C4.5), IBK (KNN), and MLP (NN). In the proposed model, the three classifiers work in parallel, and each classifier builds a different model of the data. The outputs of the three classifiers are combined using the majority voting method to obtain the final output of the ensemble model. The selection of the three classifiers is based on the results shown in Table 6.

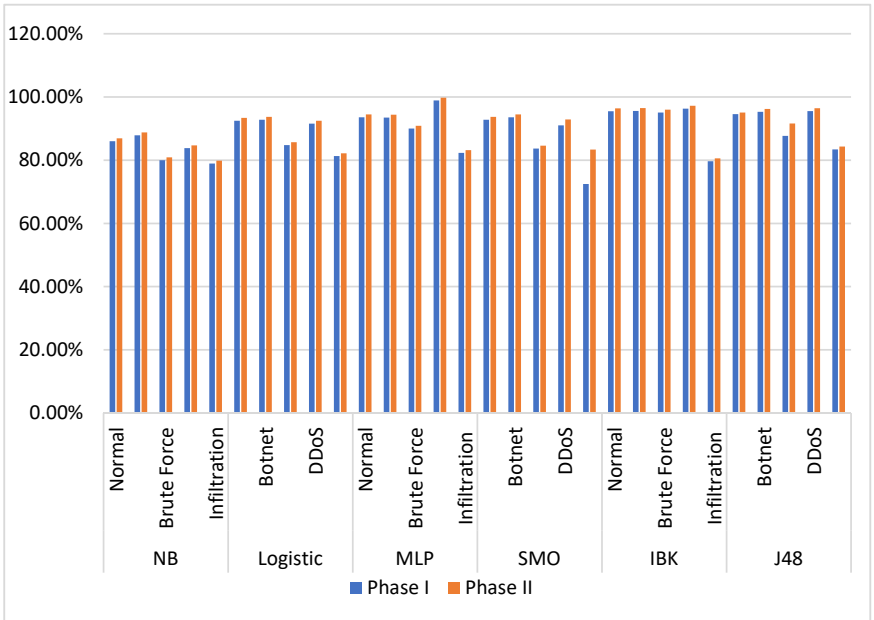

Figure 4. Accuracy of each classifier in classifying different classes in GTCS.

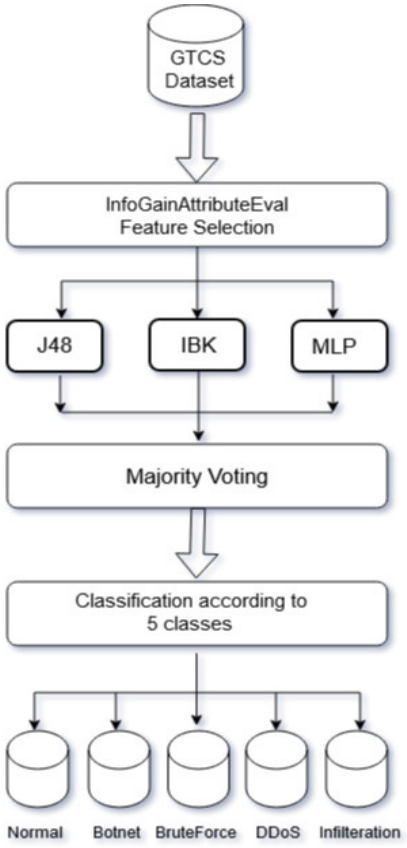

Figure 5. Ensemble classifier model flow.

In the ensemble system, each classifier builds a different model of the data based on the preprocessed dataset. To build the models, each classifier was tested using the 10-fold cross-validation technique within the dataset, wherein the dataset is divided into 10 folds or subsets. Any nine subsets were used as training sets, and the remaining subset was used as the test set. More specifically, each fold was analyzed, and the total score results determined the average performance out of the 10 folds. Majority voting is a traditional and common way to combine classifiers.

5.2.1. Experimental Results

The experimental results show that the proposed ensemble IDS model was able to outperform all single classifiers in terms of classification accuracy, as shown in Figure 6 and Table 7.

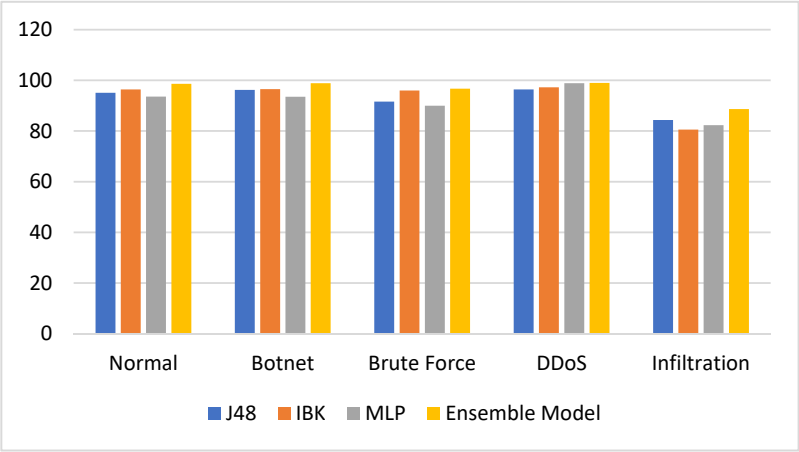

Figure 6. Accuracy of ensemble versus single classifiers.

Table 7. Individual and ensemble performances.

Class	J48			IBK			MLP			Ensemble Model		
	Acc %	TP	FP	Acc %	TP	FP	Acc %	TP	FP	Acc %	TP	FP
Normal	95.10	0.902	0.049	96.40	0.960	0.049	93.60	0.966	0.051	98.62	0.965	0.029
Botnet	96.20	0.913	0.038	96.50	0.967	0.046	93.50	0.951	0.053	98.87	0.972	0.018
Brute Force	91.60	0.876	0.068	96.00	0.960	0.050	90.00	0.816	0.068	96.70	0.901	0.031
DDoS	96.43	0.964	0.035	97.22	0.973	0.039	98.89	0.976	0.038	98.99	0.979	0.016
Infiltration	84.33	0.898	0.088	80.57	0.911	0.091	82.30	0.886	0.089	88.69	0.899	0.040

6. Conclusions and Future Work

In this paper, we presented the GTCS dataset to overcome the shortcomings of most of the existing available datasets and covered most of the necessary criteria for common updated attacks such as botnet, brute force, DDoS, and infiltration attacks. The generated dataset is completely labeled, and about 84 network traffic features have been extracted and calculated for all benign and intrusive flows. We also compared the performance of six of the most well-known ML classifiers over the new dataset and demonstrated that the ensemble of different learning paradigms can improve detection accuracy, improve TPR, and decrease FPR.

Author Contributions: Conceptualization, A.A. and A.M.; methodology, A.A. and A.M.; software, A.M.; validation, A.M., D.V.; formal analysis, A.M. and D.V.; investigation, A.M.; resources, A.A., D.V., and S.S.; writing—original draft preparation, A.M.; writing—review and editing, A.M., A.A., D.V., and S.S.; visualization, A.M.; supervision, A.M., S.S.; project administration, A.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors thank the anonymous reviewers for their insightful comments that helped improve the quality of this study.

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

References

1. Singh, R.; Kumar, H.; Singla, R.K.; Ramkumar, K. Internet attacks and intrusion detection system. *Online Inf. Rev.* **2017**, *41*, 171–184. [CrossRef]
2. Kaur, P.; Kumar, M.; Bhandari, A. A review of detection approaches for distributed denial of service attacks. *Syst. Sci. Control Eng.* **2017**, *5*, 301–320. [CrossRef]
3. Davis, J. *Machine Learning and Feature Engineering for Computer Network Security*; Queensland University of Technology: Brisbane, Australia, 2017.
4. Pacheco, F.; Exposito, E.; Gineste, M.; Baudoin, C.; Jose, A. Towards the Deployment of Machine Learning Solutions in Network Traffic Classification: A Systematic Survey. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 1988–2014. [CrossRef]
5. Zheng, A.; Casari, A. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*; O'Reilly Media, Inc.: Newton, MA, USA, 2018.
6. Bhuyan, M.H.; Bhattacharyya, D.K.; Kalita, J.K.J.I.N.S. Towards Generating Real-life Datasets for Network Intrusion Detection. *Int. J. Netw. Secur.* **2015**, *17*, 683–701.
7. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
8. Deshmukh, D.H.; Ghorpade, T.; Padiya, P. Improving classification using preprocessing and machine learning algorithms on NSL-KDD dataset. In Proceedings of the 2015 International Conference on Communication, Information and Computing Technology (ICCICT), Mumbai, India, 15–17 January 2015.

9. Nehinbe, J.O. A critical evaluation of datasets for investigating IDSs and IPSs researches. In Proceedings of the 2011 IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS), London, UK, 1–2 September 2016.
10. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *ICISSP*; University of New Brunswick: Fredericton, NB, Canada, 2018; pp. 108–116.
11. Huang, H.; Al-Azzawi, H.; Brani, H. Network traffic anomaly detection. *arXiv* **2014**, arXiv:1402.0856.
12. Lazarevic, A.; Kumar, V.; Srivastava, J. Intrusion Detection: A Survey, in Managing Cyber Threats. 2005. Available online: https://www.researchgate.net/publication/226650646_Intrusion_Detection_A_Survey (accessed on 21 October 2020).
13. Azeez, N.A.; Bada, T.M.; Misra, S.; Adewumi, A.; Van Der Vyver, C.; Ahuja, R. *Intrusion Detection and Prevention Systems: An Updated Review*; Springer Science and Business Media LLC: Berlin, Germany, 2019; pp. 685–696.
14. Yeo, L.H.; Che, X.; Lakkaraju, S. Understanding Modern Intrusion Detection Systems: A Survey. *arXiv* **2017**, arXiv:1708.07174.
15. Fadlullah, Z.M.; Tang, F.; Mao, B.; Kato, N.; Akashi, O.; Inoue, T.; Mizutani, K. State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2432–2455. [CrossRef]
16. Shalev-Shwartz, S.; Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*; Cambridge University Press: Cambridge, UK, 2014.
17. Yuan-Fu, Y. A Deep Learning Model for Identification of Defect Patterns in Semiconductor Wafer Map. In Proceedings of the 30th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC), Saratoga Springs, NY, USA, 6–9 May 2019.
18. Claesen, M.; De Moor, B. Hyperparameter search in machine learning. *arXiv* **2015**, arXiv:1502.02127.
19. Ryu, J.; Kantardzic, M.; Walgampaya, C. Ensemble Classifier based on Misclassified Streaming Data. In Proceedings of the 10th IASTED International Conference on Artificial Intelligence and Applications, Innsbruck, Austria, 15–17 February 2010.
20. Elmomen, A.A.; El Din, A.B.; Wahdan, A. Detecting Abnormal Network Traffic in the Secure Event Management Systems. In *International Conference on Aerospace Sciences and Aviation Technology*; The Military Technical College: Cairo, Egypt, 2011.
21. BalaGanesh, D.; Chakrabarti, A.; Midhunchakkaravarthy, D. Smart Devices Threats, Vulnerabilities and Malware Detection Approaches: A Survey. *Eur. J. Eng. Res. Sci.* **2018**, *3*, 7–12. [CrossRef]
22. Hansen, L.K.; Salamon, P. Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.* **1990**, *12*, 993–1001. [CrossRef]
23. Ho, T.K. *Multiple Classifier Combination: Lessons and Next Steps*; World Scientific: Singapore, 2002; pp. 171–198.
24. Wang, G.; Hao, J.; Ma, J.; Jiang, H. A comparative assessment of ensemble learning for credit scoring. *Expert Syst. Appl.* **2011**, *38*, 223–230. [CrossRef]
25. Koch, R.; Golling, M.; Rodosek, G.D. Towards comparability of intrusion detection systems: New data sets. In Proceedings of the TERENA Networking Conference, Dublin, Ireland, 19–22 May 2014.
26. Paxson, V.; Floyd, S. Why we don't know how to simulate the Internet. In Proceedings of the 29th Conference on Winter Simulation, Atlanta, GA, USA, 7–10 December 1997.
27. Ghorbani, A.A.; Lu, W.; Tavallaee, M. *Network Intrusion Detection and Prevention*; Springer Science and Business Media LLC: Berlin, Germany, 2009; Volume 47.
28. Lee, K.-C.; Orten, B.; Dasdan, A.; Li, W. Estimating conversion rate in display advertising from past erformance data. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Beijing, China, 12–16 August 2012.
29. Beck, J.E.; Woolf, B.P. High-Level Student Modeling with Machine Learning. In *Proceedings of the Lecture Notes in Computer Science*; Springer: Berlin, Germany, 2000; pp. 584–593.
30. Karimi, Z.; Kashani, M.M.R.; Harounabadi, A. Feature Ranking in Intrusion Detection Dataset using Combination of Filtering Methods. *Int. J. Comput. Appl.* **2013**, *78*, 21–27. [CrossRef]
31. Kohavi, R.; John, G.H. Wrappers for feature subset selection. *Artif. Intell.* **1997**, *97*, 273–324. [CrossRef]
32. John, G.; Kohavi, R.; Pfleger, K. Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the Eleventh International Conference*; Morgan Kaufmann: Burlington, MA, USA, 1994.

33. Biesiada, J.; Duch, W. *Feature Selection for High-Dimensional Data: A Kolmogorov-Smirnov Correlation-Based Filter*; Springer: Berlin, Germany, 2008; pp. 95–103.
34. Araújo, N.; De Oliveira, R.; Ferreira, E.; Shinoda, A.A.; Bhargava, B. Identifying important characteristics in the KDD99 intrusion detection dataset by feature selection using a hybrid approach. In Proceedings of the 2010 17th International Conference on Telecommunications, Doha, Qatar, 4–7 April 2010; pp. 552–558.
35. Chebrolu, S.; Abraham, A.; Thomas, J.P. Hybrid Feature Selection for Modeling Intrusion Detection Systems. In *Proceedings of the Computer Vision*; Springer: Berlin, Germany, 2004; pp. 1020–1025.
36. Guennoun, M.; Lbekkouri, A.; El-Khatib, K. Optimizing the feature set of wireless intrusion detection systems. *Int. J. Comput. Sci. Netw. Secur.* **2008**, *8*, 127–131.
37. Talavera, L. An Evaluation of Filter and Wrapper Methods for Feature Selection in Categorical Clustering. Available online: https://www.cs.upc.edu/~{}talavera/_downloads/ida05fs.pdf (accessed on 21 October 2020).
38. Moradi, P.; Gholampour, M. A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy. *Appl. Soft Comput.* **2016**, *43*, 117–130. [CrossRef]
39. Liu, Y.; Yu, X.; Huang, J.X.; An, A. Combining integrated sampling with SVM ensembles for learning from imbalanced datasets. *Inf. Process. Manag.* **2011**, *47*, 617–631. [CrossRef]
40. 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*, 1–11. [CrossRef]
41. Zhai, Y.; Ma, N.; Ruan, D.; An, B. An effective over-sampling method for imbalanced data sets classification. *Chin. J. Electron.* **2011**, *20*, 489–494.
42. Chawla, N.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
43. Yen, S.-J.; Lee, Y.-S. Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Syst. Appl.* **2009**, *36*, 5718–5727. [CrossRef]
44. Hasanin, T.; Khoshgoftaar, T.M.; Leevy, J.L.; Seliya, N. Investigating Random Undersampling and Feature Selection on Bioinformatics Big Data. In Proceedings of the 2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService), Newark, CA, USA, 4–9 April 2019; pp. 346–356.
45. Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M. A Deep Learning Approach for Network Intrusion Detection System. In Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (Formerly BIONETICS), New York, NY, USA, 3–5 December 2016.
46. Dhanabal, L.; Shantharajah, S.P. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *Int. J. Adv. Res. Comput. Commun. Eng.* **2015**, *4*, 446–452.
47. Hodo, E.; Bellekens, X.; Hamilton, A.; Tachtatzis, C.; Atkinson, R. Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey. *arXiv* **2017**, arXiv:1701.02145.
48. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv.* **2009**, *41*, 1–58. [CrossRef]
49. MeeraGandhi, G. Machine learning approach for attack prediction and classification using supervised learning algorithms. *Int. J. Comput. Sci. Commun.* **2010**, *1*, 11465–11484.
50. Nguyen, H.A.; Choi, D. Application of Data Mining to Network Intrusion Detection: Classifier Selection Model. In *Proceedings of the Computer Vision*; Springer: Berlin, Germany, 2008.
51. Darshan, V.S.; Raphael, R. Real Time Call Monitoring System Using Spark Streaming and Network Intrusion Detection Using Distributed WekaSpark. *J. Mach. Intell.* **2017**, *2*, 7–13. [CrossRef]
52. Belavagi, M.C.; Muniyal, B. Performance Evaluation of Supervised Machine Learning Algorithms for Intrusion Detection. *Procedia Comput. Sci.* **2016**, *89*, 117–123. [CrossRef]
53. Hota, H.S.; Shrivas, A.K. Decision Tree Techniques Applied on NSL-KDD Data and Its Comparison with Various Feature Selection Techniques. In *Advanced Computing, Networking and Informatics*; Springer: Berlin, Germany, 2014; Volume 1, pp. 205–211.
54. Khammassi, C.; Krichen, S. A GA-LR wrapper approach for feature selection in network intrusion detection. *Comput. Secur.* **2017**, *70*, 255–277. [CrossRef]
55. Abdullah, M.; Alshannaq, A.; Balamash, A.; Almabdy, S. Enhanced intrusion detection system using feature selection method and ensemble learning algorithms. *Int. J. Comput. Sci. Inf. Secur.* **2018**, *16*, 48–55.
56. Chebrolu, S.; Abraham, A.; Thomas, J.P. Feature deduction and ensemble design of intrusion detection systems. *Comput. Secur.* **2005**, *24*, 295–307. [CrossRef]

57. Roli, F.; Kittler, J. *Multiple Classifier Systems: Third International Workshop, MCS 2002, Cagliari, Italy, 24–26 June 2002. Proceedings*; Springer Science & Business Media: Berlin, Germany, 2002; Volume 2364.
58. Hansen, J.V.; Lowry, P.B.; Meservy, R.D.; McDonald, D.M. Genetic programming for prevention of cyberterrorism through dynamic and evolving intrusion detection. *Decis. Support Syst.* **2007**, *43*, 1362–1374. [CrossRef]
59. Koza, J.R.; Poli, R. A Genetic Programming Tutorial. 2003. Available online: https://www.researchgate.net/publication/2415604_A_Genetic_Programming_Tutorial (accessed on 21 October 2020).
60. Srivats, P. Ostinato Packet Generator. 2018. Available online: https://ostinato.org (accessed on 11 November 2019).
61. Najera-Gutierrez, G.; Ansari, J.A. *Web Penetration Testing with Kali Linux: Explore the Methods and Tools of Ethical Hacking with Kali Linux*; Packt Publishing Ltd.: Birmingham, UK, 2018.
62. Sousa, O.F. Analysis of the package dependency on Debian GNU/Linux. *J. Comput. Interdiscip. Sci.* **2009**, *1*, 127–133. [CrossRef]
63. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-baiot—Network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22. [CrossRef]
64. Arzhakov, A.V.; Silnov, D.S. Analysis of Brute Force Attacks with Ylmf-pc Signature. *Int. J. Electr. Comput. Eng.* **2016**, *6*, 1681–1684.
65. Sharma, K.; Gupta, B.B. Taxonomy of Distributed Denial of Service (DDoS) Attacks and Defense Mechanisms in Present Era of Smartphone Devices. *Int. J. E Serv. Mob. Appl.* **2018**, *10*, 58–74. [CrossRef]
66. Kirda, E. Getting Under Alexa's Umbrella: Infiltration Attacks Against Internet Top Domain Lists. In Proceedings of the Information Security: 22nd International Conference (ISC 2019), New York, NY, USA, 16–18 September 2019.
67. Yan, G.; Brown, N.; Kong, D. Exploring discriminatory features for automated malware classification. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*; Springer: Berlin, Germany, 2013.
68. Lawrence, D. The Hunt for the Financial Industry's Mostwanted Hacker. 2015. Available online: https://www.bloomberg.com/news/features/2015-06-18/the-hunt-for-the-financial-industry-s-most-wanted-hacker (accessed on 21 October 2020).
69. Nagpal, B.; Sharma, P.; Chauhan, N.; Panesar, A. DDoS tools: Classification, analysis and comparison. In Proceedings of the 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 11–13 March 2015.
70. Goyal, P.; Goyal, A. Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark. In Proceedings of the 2017 9th International Conference on Computational Intelligence and Communication Networks (CICN), Girne, Cyprus, 16–17 September 2017; pp. 77–81.
71. Ndatinya, V.; Xiao, Z.; Manepalli, V.R.; Meng, K.; Xiao, Y. Network forensics analysis using Wireshark. *Int. J. Secur. Netw.* **2015**, *10*, 91–106. [CrossRef]
72. Draper-Gil, G.; Lashkari, A.H.; Mamun, M.S.I.; Ghorbani, A.A. Characterization of encrypted and vpn traffic using time-related. In Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP), Rome, Italy, 19–21 February 2016; pp. 407–414.
73. Lashkari, A.H.; Draper-Gil, G.; Mamun, M.S.I.; Ghorbani, A.A. Characterization of tor traffic using time based features. In Proceedings of the ICISSP, Porto, Portugal, 19–21 February 2017; pp. 253–262.
74. Mahfouz, A.; Abuhussein, A.; Shiva, S. GTCS Network Attack Dataset 2020. Available online: https://www.researchgate.net/publication/344478320_GTCS_Network_Attack_Dataset (accessed on 21 October 2020).
75. Amrita, M.A. Performance analysis of different feature selection methods in intrusion detection. *Int. J. Adv. Res. Comput. Eng. Technol.* **2013**, *2*, 1725–1731.