Journal Article

Detection and Classification of DDoS Flooding Attacks on Software-Defined Networks: A Case Study for the Application of Machine Learning

Sangodoyin, A. O., Akinsolu, M.O., Pillai, P. and Grout, V.

Recommended citation:

Sangodoyin, A. O., Akinsolu, M.O., Pillai, P. and Grout, V. (2021) 'Detection and Classification of DDoS Flooding Attacks on Software-Defined Networks: A Case Study for the Application of Machine Learning,' *IEEE Access*, vol. 9, pp. 122495 - 122508, August 2021, doi: 10.1109/ACCESS.2021.3109490

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 3, 2021, accepted August 22, 2021, date of publication August 31, 2021, date of current version September 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3109490

Detection and Classification of DDoS Flooding Attacks on Software-Defined Networks: A Case Study for the Application of Machine Learning

ABIMBOLA O. SANGODOYIN[1], (Member, IEEE), MOBAYODE O. AKINSOLU[2], (Member, IEEE), PRASHANT PILLAI[1], (Senior Member, IEEE), AND VIC GROUT[2]

[1]School of Mathematics and Computer Science, University of Wolverhampton, Wolverhampton WV1 1LY, U.K.
[2]Faculty of Arts, Science and Technology, Wrexham Glyndŵr University, Wrexham LL11 2AW, U.K.

Corresponding authors: Abimbola O. Sangodoyin (asangodoyin@ieee.org) and Mobayode O. Akinsolu (m.o.akinsolu@ieee.org)

This work was supported in part by the Faculty of Arts, Science and Technology, Wrexham Glyndŵr University, U.K.

ABSTRACT Software-defined networks (SDNs) offer robust network architectures for current and future Internet of Things (IoT) applications. At the same time, SDNs constitute an attractive target for cyber attackers due to their global network view and programmability. One of the major vulnerabilities of typical SDN architectures is their susceptibility to Distributed Denial of Service (DDoS) flooding attacks. DDoS flooding attacks can render SDN controllers unavailable to their underlying infrastructure, causing service disruption or a complete outage in many cases. In this paper, machine learning-based detection and classification of DDoS flooding attacks on SDNs is investigated using popular machine learning (ML) algorithms. The ML algorithms, classifiers and methods investigated are quadratic discriminant analysis (QDA), Gaussian Naïve Bayes (GNB), k-nearest neighbor (k-NN), and classification and regression tree (CART). The general principle is illustrated through a case study, in which, experimental data (i.e. jitter, throughput, and response time metrics) from a representative SDN architecture suitable for typical mid-sized enterprise-wide networks is used to build classification models that accurately identify and classify DDoS flooding attacks. The SDN model used was emulated in Mininet and the DDoS flooding attacks (i.e. hypertext transfer protocol (HTTP), transmission control protocol (TCP), and user datagram protocol (UDP) attacks) have been launched on the SDN model using low orbit ion cannon (LOIC). Although all the ML methods investigated show very good efficacy in detecting and classifying DDoS flooding attacks, CART demonstrated the best performance on average in terms of prediction accuracy (98%), prediction speed (5.3×10^5 observations per second), training time (12.4 ms), and robustness.

INDEX TERMS SDN security, DDoS flooding attack, machine learning, network security.

I. INTRODUCTION

Networking technologies such as IoT are growing at steady rates in terms of users, intermediate systems (i.e. network devices), and applications. Recent statistics also suggest that the number of connected devices will continue to grow towards several billions by 2025 [1], [2]. The increasing emphasis on seamless and distributed connectivity, cloud-based applications, and real-time network monitoring and automation also suggests that these trends are likely to continue [3]. In line with this growth and the quality of service delivery to accommodate the increasing demands of network users, there is a synergy between research communities and industries in exploring innovative ways of modeling network architectures [4], [5]. A particular area of interest in this regard is software-defined networks (SDNs) that decouple their control planes from their data planes [6]. This distinct feature of SDNs has led to many state-of-the-art IoT network architectures such as 5G and beyond network architectures being SDN-based [7].

As a fast-growing technology, SDN simplifies network administration, management, and monitoring by using

The associate editor coordinating the review of this manuscript and approving it for publication was Yuan Zhuang.

OpenFlow protocol [8]. With the help of the controller, SDN equips network administrators with a global view of the network and supports the interactive programming of the underlying data plane elements, simultaneously. Hence, the SDN controller is the intelligence of the network that exercises direct control over data plane devices through the application programming interface (API) [8]. Despite the programmability, flexibility, and decentralized control of SDNs, which are very critical to their successful deployment, these features tend to be at odds with making SDNs more secure [9]. Consequently, many SDN architectures still suffer from several security breaches which include (but are not limited to) increased potential for DDoS flooding attacks [10], [11].

DDoS flooding attacks are coordinated attacks launched using a large number of compromised hosts to usurp available network bandwidth or render target nodes or end-user devices on the network completely unavailable [12]. They constitute the most dangerous malicious traffic on the internet [13]. Typically, perpetrators of DDoS flooding attacks surreptitiously employ entire networks of infected devices (i.e., botnet) to orchestrate attacks [14]. In this way, end users of the device nodes on the attacked network are often unaware attacks are being launched from their devices and IP addresses. Since the internet does not mandate any requirements for flow control aside from end hosts, IoT devices are naturally highly prone to coordinated attacks such as DDoS flooding attacks [13]. This vulnerability can be attributed to the exponential growth in the number of IoT devices while the security and protection of these devices are still being enhanced gradually [15].

There is no unified rationale behind DDoS flooding attacks; they can be motivated by competition or, political and monetary reasons, amongst other factors. Due to the sophistication of DDoS flooding attack tools (e.g., HTTP unbearable load king (HULK), High orbit ion canon (HOIC), etc.) and cheap online stress booters, there is a crucial need for an up-to-date defense mechanism to counter both known and unknown variants of DDoS flooding attacks. To detect and mitigate DDoS flooding attacks such as HTTP, TCP, and UDP flooding attacks, many solutions have been proposed [16]–[20].

Identifying both known and unknown attack traffic promptly, with the introduction of minimal overhead, is crucial for the continued operation of critical network devices and systems on SDNs. To achieve this, machine learning (ML) techniques have found numerous successful applications in the intrusion detection and prevention systems (IDPSs) of SDNs. Even though ML-based approaches have assisted in the successful detection and mitigation of DDoS flooding attacks [21], [22], sniffing network data for further exploitation, launching DDoS flooding attacks and reprogramming the entire SDN by malicious users is still a complex area requiring increased security hardening [23]. To address these problems, a rapid low-cost method of identifying traffic scenarios on SDNs to guarantee their availability and reliability while scrutinizing and legitimizing requests from network devices and systems is required.

In this paper, as an initiation of addressing the problems discussed above, the low-cost ML-based detection and classification of DDoS flooding attacks on SDNs is investigated using empirical network data. The dataset is generated within a controlled simulation environment and covers three types of DDoS flooding attacks: HTTP flooding attack, TCP flooding attack, and UDP flooding attack. HTTP, TCP, and UDP flooding attacks are applied since these protocols represent a large portion of web applications' traffic usage and characteristics [24]. Additionally, the network model developed and emulated in this paper is similar to those obtainable in modern enterprise-wide network architectures.

A custom modeled SDN architecture based on a tree topology for the extraction of real-time SDN traffic data before and during DDoS flooding attacks has been adopted in this paper to make the contributions summarized as follows:

- Generic vulnerability testing of the SDN by launching coordinated TCP, UDP, and HTTP flooding attacks on the SDN server at the data plane to render the SDN unavailable.
- Detection and classification of UDP, TCP, and HTTP flooding attacks on the SDN to illustrate the application of popular ML algorithms in the detection and classification of DDoS flooding attacks in SDNs.
- Comparative analysis and comprehensive evaluation of the popular ML algorithms to assess the performance of their prediction models in detecting and classifying DDoS flooding attacks in SDNs over a number of statistical runs.

The case study for applying ML algorithms to detect and classify DDoS flooding attacks using metrics from an emulated SDN presented in this work, demonstrates the suitability of the investigated ML algorithms as viable add-ons or complementary programs which can be featured in the SDN controller to expedite the practical detection and classification of present and potential DDoS flooding attacks on real-world SDNs. The remainder of this paper is organized as follows: Section II presents related works on DDoS attack detection methods in SDNs. Section III presents the proposed SDN architecture and the emulated DDoS flooding attack scenarios. The classification algorithms or methods are elaborated in Section IV. In Section V, the details of the experimental setup are provided. The results and discussions from experimentation are detailed in Section VI and concluding remarks are provided in Section VII.

II. RELATED WORKS

There are various current approaches to DDoS flooding attack detection based on test data, attack types, and modeled or controlled simulation scenarios. Efficient DDoS flooding attack detection mechanisms typically operate by distinguishing malicious traffic packets and patterns from legitimate ones [25]. These mechanisms are widely deployed in conventional networks, SDNs, and ML-assisted SDNs. [20] offers, a comprehensive overview of DDoS flooding attacks

and suitable defence mechanisms. Here the defence mechanisms (i.e. preventive, reactive, and cooperative) presented are broadly categorised based on the level of defence activity and the degree of cooperation between autonomous, cooperative and interdependent entities on the internet. In [26], dynamic resource allocation is shown to aid in the accurate detection of DDoS flooding attacks. In particular, a dynamic way of utilizing reserved cloud resources to cloud customers under DDoS flooding attack is demonstrated by employing a queueing theory-based model which ensures the availability of cloud services to benign users.

ML is another popular way of detecting and classifying DDoS flooding attacks [21], [27], [28]. ML is a wide interdisciplinary area of research that involves learning patterns from datasets for computers to perform specific tasks without explicit instructions. Some of the most popular ML methods for classification include C4.5, k-Means, support vector machine (SVM), Apriori, Expectation–maximization (EM), PageRank, AdaBoost, k nearest neighbour (k-NN), Naïve Bayes (NB), and classification and regression tree (CART) [29]. For example, in [27], a radial-basis-function neural network model built and trained using some statistical network features is employed to describe and classify DDoS flooding attacks in terms of behavior. Even though the detection rates reported are relatively high, the monitoring system is passive. To better detect DDoS flooding attacks, by recognising the phases of the attacks, a clustering-based method, employing the entropy values on select attributes of the network, is proposed in [30].

Considering the flow classes obtainable on a data center network, a DDoS flooding attack detection approach is proposed in [31] through the use of a correlation analysis model. The model works by predicting flow classes based on the k-closest training data points in the feature space and an evaluation of their influence is then performed using correlation analysis. In [32], flow events are collected from the switch interface of the network and a sequential probability ratio test, having a bounded false positive and false negative error rates threshold, is applied for decision making and identification of compromised interfaces. The use of network entropy values has also been explored in [33]–[35] to identify anomalous network traffic flows from normal network traffic flows. When used with suitable window sizes and appropriate thresholds, these entropy values provide a measure of the randomness of benign traffic and malicious traffic together with good detection accuracies [34].

Rate limiting has also been applied in DDoS flooding attack detection and mitigation in SDNs [36], [37]. A primary advantage of rate limiting is that it shields SDNs from complete outages during DDoS flooding attacks. However, all the flows within the SDNs are still affected by this approach and the response time of SDNs increase for legitimate traffic. A support vector machine approach was applied to the existing defence advanced research projects agency (DARPA) dataset to classify DDoS attacks [21]. However, there is no comparison with the dataset received from the emulated SDN environment. Self-organizing maps (SOMs) based on Artificial Neural Network (ANNs) have also been applied to detect DDoS flooding attacks in SDNs [38]. Using six-tuple features to train the traffic flow, the ANN-based SOM showed better performance compared to other methods involving the KDD-99 dataset. An application of the extreme gradient boosting (XGBoost) algorithm for the detection and classification of DDoS flooding attacks can be found in [39]. These results show better performance compared to random forest, support vector machine, and gradient boosting decision tree when applied in an SDN-based cloud network. Deep learning algorithms may also be employed to detect DDoS flooding attacks in SDNs [22], [28].

As discussed, although several ML techniques have been applied to DDoS flooding attack detection and classification in SDNs, most of the available approaches employing them make use of KDD and NSL-KDD datasets with only a few making use of real-time experimental network data from SDN configuration as carried out in this paper. The work presented in this paper addresses this apparent research gap. Many DDoS flooding attack detection techniques rely on identifying key features (i.e. traffic flow metrics) relevant to distinguishing malicious network traffic from benign network traffic. In this work, we compare the efficacy of four popular ML methods (i.e. Gaussian NB (GNB), quadratic discriminant analysis (QDA), k-NN, and CART) for the ML-based detection and classification of TCP, UDP, and HTTP DDoS flooding attacks using the jitter, throughput, and response time metrics from emulated SDN scenarios. In this manner, the emulated SDN scenarios focus on key real-time traffic features obtainable in typical mid-sized enterprise SDN architectures. The aim of this approach, in contrast to the related works discussed above, is to demonstrate the viability of popular ML algorithms in the detection and classification of DDoS flooding attacks considering emulated SDN scenarios that are representative of real-world SDN scenarios.

III. PROPOSED SDN ARCHITECTURE AND EMULATION OF ATTACK SCENARIOS

In this work, a custom SDN architecture based on a tree topology is designed using the Mininet emulator [40] and the vulnerability of the SDN is assessed by introducing DDoS flooding attacks. In particular, a tree topology is considered because it can be easily adapted for wide area networks and supports scalability. The custom SDN topology (Figure 1) was implemented on a 32.0 GB RAM Intel Xeon E3-1220 processor with Kali Linux as the base operating system. The floodlight controller [41] for the SDN was deployed in Oracle VM VirtualBox running Ubuntu 18.10 LTS, while the Mininet software was deployed in Oracle VM VirtualBox running Ubuntu 16.10 LTS.

The modeled SDN comprises 10 OpenFlow switches and 16 hosts, connected using a 100 Mbps link. The essential software tool includes ‘‘iperf’’, which is used to create the client-server relationship and the Low Orbit Ion Canon (LOIC) [42], which is used to generate the DDoS flooding

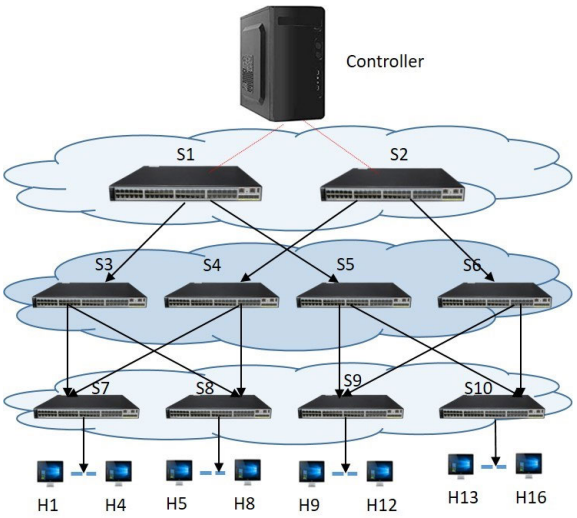


FIGURE 1. Modelled SDN tree architecture.

attacks. By using ''iperf'' and ''ping'' commands to generate legitimate traffic between the host and server, system properties such as response time, throughput, and jitter values generated were recorded every second for a duration of 15 minutes. For the attack scenarios, an assumption (based on conventional DDoS flooding attack strategy on SDNs shown in Figure 2 and the work presented in [43]) that attacks emanate sequentially from an internal source was made as follows:

- An attacker scans the SDN for vulnerable active hosts.
- The vulnerable hosts are then compromised for exploitation.
- Finally, compromised hosts are used to launch attacks on victims.

Based on the above strategy, compromised hosts within the SDN were used to launch HTTP, TCP, and UDP flooding attacks on the SDN server for 15 minutes, respectively, and the corresponding data for response time, throughput, and jitter were recorded. This amounted to a total data size of 1.6 GB. The data recorded during legitimate traffic between the host and server, and when the SDN was attacked with HTTP, TCP, and UDP flooding attacks, are converted to .txt files, and Knime is used to derive the response time, jitter, and throughput metrics (and remove duplicate records). The refined dataset, with no missing data, is then inserted into a data array. The data array holds the dataset for the construction of the classification models used to investigate the ML-based detection and classification of the DDoS flooding attacks launched on the SDN.

This dataset captures both normal and abnormal profiles of the SDN, as required for ML-based anomaly detection in SDNs [44], by modeling modern DDoS flooding attacks in SDNs. The dataset collected and analysed addresses the following limitations of existing benchmark SDN datasets [45], [46]:

1. The lack of a modern footprint attack model obtainable in SDNs.
2. Variations in normal traffic records due to fewer connected devices a few years ago compared to the present IoT era.
3. Variations in the distribution of attacks of old benchmark testing set and new training set [47].

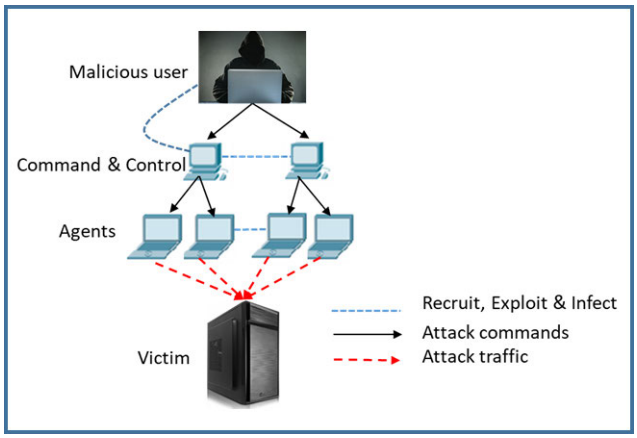


FIGURE 2. DDoS attack strategy.

To further clarify how the DDoS flooding attacks have been emulated on the SDN (to degrade the performance and availability of the targeted equipment [48]), the typical DDoS flooding attack process used in the emulation is shown in Figure 3. Here, the malicious user crafts a packet and sends it to the victim. In a situation where there is no match entry rule in the table of the OpenFlow switch, the OpenFlow switch encapsulates the header of the packet and sends a ''packet_in'' message to the controller for instruction. The controller decrypts the ''packet_in'' message and determines the appropriate route to the destination address based on the installed rule (e.g., drop, forward packet to port, send) on the controller. A ''packet_out'' message is then generated and a new flow rule is installed in the OpenFlow switch table. The OpenFlow switch, in turn, forwards ''packet_out'' to the destination address.

It is important to note that, if a malicious user spoofs the source IP address and generates spurious ''packet_in'' messages at specified intervals, more ''packet_in'' messages will be sent to the controller. As a result, the data and control plane resources become vulnerable to DDoS flooding attacks depending on the attack strategies deployed [49]. This is why building and employing robust DDoS flooding attack detection algorithms are vital for SDN controllers.

IV. THE CLASSIFICATION ALGORITHMS

To undertake low-cost detection and classification of DDoS flooding attacks on SDNs using historical and recurrent values or thresholds for throughput (T_p), response time (R_t), and jitter (J_t) data for a given event or scenario, four classic machine learning algorithms or methods are employed in this work. These well-established methods are the Discriminant Analysis (DA) [50], Decision Tree (DT) [51], [52], Naïve Bayes (NB) [53], [54], and k-Nearest Neighbours (k-NN) [55]. These methods are used due to their popularity

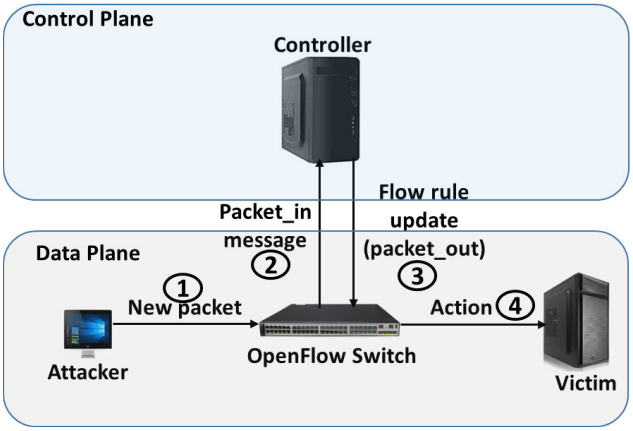


FIGURE 3. DDoS attack process in SDN.

and global recognition in machine learning applications such as data mining [29].

A. DISCRIMINANT ANALYSIS (DA) CLASSIFIER

DA is a statistical learning method, often used to find the optimum combination of features that characterizes two or more classes of observations. The DA classifier could be linear DA (LDA) or quadratic DA (QDA). LDA and QDA are widely used primarily due to their inherent capacity to address many multiclass problems without the requirement to tune hyperparameters to improve their classification accuracy [56]–[58].

The classification model for the DA method works as follows [50]: For N classes (e.g., $y = [y_1, y_2, \ldots, y_N]$), each class generates data (e.g., $x = [x_1, x_2, \ldots, x_d]$) where d is the number of observations or the dimensional space according to a multivariate normal distribution. The classification model has the same covariance matrix for all the classes for LDA, whereas both the means and covariances vary in each of the classes for QDA. LDA may suffer from high bias if a difference exists in the covariance matrices [59]. To avoid this, QDA is employed in this work.

Typically, DA carries out predictive classification by using Bayes' rule and minimizing the expected classification cost as follows [50]:

$$\widehat{y} = \underset{y=1,\ldots,N}{\arg\min} \sum_{N=1}^{N} \widehat{K}(n|x)E(y|n) \tag{1}$$

where $\widehat{y}$ is the predicted class, $\widehat{K}(n|x)$ is the posterior probability of class n for the observation x and $E(y|n)$ is the cost of classifying or predicting an observation as y when its true class is n. The class n which maximizes $\widehat{K}(n|x)$ is selected.

Typically, the $\widehat{K}(n|x)$ that x belongs to class n is the multiplication of the multivariate normal density and the prior probability. For x-data having d-dimensions, the density function of the multivariate normal having a mean of μ_n (1-by-d matrix) and covariance of σ_n (d-by-d matrix) at a 1-by-d point x can be expressed as follows [60]:

$$K(n|x) = \frac{1}{\sqrt{((2\pi)^d|\sigma_n|)}} e^{-(-\frac{1}{2}(x-\mu_n)\sigma_n^{-}1(x-\mu n)^T)} \tag{2}$$

where $|\sigma_n|$, is the determinant of σ_n, ${\sigma_n}^{-1}$ is its inverse matrix and $(x-\mu_n)^T$ is the transpose of $(x-\mu_n)$. If $K(n)$ represents the prior probability of class n, then the posterior probability that an observation x belongs to class n is stated as follows:

$$\widehat{K}(n|x) = \frac{K(n|x)K(n)}{K(x)} \tag{3}$$

$K(x)$ is the sum over n of $K(x|n)K(n)$ and it is often referred to as a normalization constant. In practice, since $K(x)$ does not depend on n and values of the features of x are known or given, $K(x)$ is effectively a constant and $K(x|n)K(n)$ is a joint probability model.

As shown in Table 1, DA has no control parameters or hyperparameters. More details on DA and its variants can be found in [50], [61].

B. NAIVE BAYES (NB) CLASSIFIER

As with the DA classifier, a typical NB classifier also relies on Bayes' theorem and applies probability density information to the training data [53], [54]. NB assumes (simply) that the predictors for each given class are conditionally independent [54]. Even though this conditional mutual independence assumption does not usually hold for most practical cases [54], Bayes' classification often provides robust posterior distributions for biased class density estimates at the decision boundary (i.e. where the posterior probability is 50%) [62]. Bayes' classifier is scalable and only a small number of training data is required for the estimation of the parameters used for classification [54], [63].

NB primarily works by assigning observations to the most probable class in accordance with the maximum a posteriori (MAP) decision rule. After estimating the densities of the predictors within all the classes, NB models posterior probabilities according to Bayes' rule to have the independent feature model or NB probability model that assigns the class label $\widehat{y} = n$ for some observation as follows [64]:

$$\widehat{y} = \underset{n\in(1,\ldots,N)}{\arg\max}\ \widehat{K}(n) \prod_{i=1}^{d} \widehat{K}(x_i|n) \tag{4}$$

To ensure the generality of the NB classification in this work, the continuous features associated with each class are assumed to have a normal (Gaussian) distribution. Hence, the Gaussian NB (GNB) is employed in this work. For example, given some data that contains a continuous attribute such as x (the observations stated above), for GNB, the data is initially segmented by the class, then the mean and variance of x is evaluated in each class. If the mean and Bessel corrected variance of the values in x associated with class n are μ_n and ${\sigma_n}^2$, respectively, the probability distribution for an observation value, x_v, from x given a class n, can

be obtained as follows [65]:

$$\widehat{K}(x_v|n) = \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{(x_v-\mu_k)^2}{2\sigma_n^2}} \tag{5}$$

As with QDA, GNB has no control parameters or hyperparameters as shown in Table 1. More details on NB classification and its variants (multinominal NB model and multivariate NB model) can be found in [66].

C. DECISION TREE

Generally, a decision tree (DT) is an acyclic graph (i.e. flowchart-like structure) which can be employed to make decisions such as classification. A DT model works by predicting the target feature based on several input features for a given classification problem where each element of the domain of the target feature is a class [51], [52]. For DTs, a specific input feature is examined at each internal branching (non-leaf) node and made to follow either the left branch or right branch after a split [51], [52]. This process is repeated recursively, and is known as recursive partitioning [51]. Each leaf of the DT has a class label or a probability distribution over all possible classes. The decision about the class for an instance of input feature, is made once a leaf node is reached and splitting no longer improves the predictions, or the specified maximum number of decision splits (or branch nodes) has been reached [51], [52].

In this work, the DT learning employs classification and regression tree (CART). CART is a non-parametric DT learning technique that hybridizes classification trees and regression trees and works as follows [51]: Given a set of predictors or input features $x = [x_1, x_2, \ldots, x_d]$, at each node, evaluate the weighted impurity in the node and estimate the probability that $x_v \in x$ is in the node using:

$$K(T) = \sum_{j\in T}^{\forall j} w_j \tag{6}$$

where T is the set of all the indexes of observation in the node and w_j is the weight of the observation j. At each node, the observations in the node are sorted and the best direction of split (i.e. left or right) is determined by maximizing the impurity gain over all possible candidates for splitting (i.e. for all splitting candidates in x).

Assuming x does not contain any missing values, the impurity gain (I_G), for the current splitting candidate can be estimated as follows:

$$I_G = K(T) - K(T_L) - K(T_R) \tag{7}$$

where $K(T_L)$ and $K(T_R)$ are the probabilities associated with a particular splitting candidate in the sets containing observation indexes in the sets T_L and T_R, respectively, after splitting all the observations at the node. In this work, Gini's diversity index (gdi), which is very popular with CART has been used to estimate the impurity [51] and is estimated as follows:

$$gdi = 1 - \sum_{n=1}^{N} (K_n)^2 \tag{8}$$

where N is the number of classes at the node and K_n is the probability of a feature being classified to belong to a particular class.

As with QDA and GNB, CART has no hyperparameters. However, the stopping criterion (e.g. the maximum number of splits) and the pruning method directly influence the DT classifier's performance [51], [67]. For our experiments we have set the maximum number of splits to be 100 as shown in Table 1. For simplicity and to ensure fairness in comparisons to other methods, the DT classifier is not pruned in this work.

D. k-NEAREST NEIGHBOURS (k-NN)

By convention, k-Nearest Neighbours (k-NN) is a non-parametric ML algorithm [68]. k-NN is widely used to address classification problems [69]. Given a new observation or point (x_{new}), k-NN finds the k nearest neighbours (or closet points) to x_{new} in the training data and returns a classification label that has the largest posterior probability among the response values for the k points to determine $\widehat{y}$. A special case of k-NN referred to as NN is when the nearest neighbour (i.e. $k = 1$) to x_{new} is used to determine $\widehat{y}$. In k-NN, the closeness of two points is estimated using a distance function. Several popular distance functions such as Euclidean distance, Minkowski distance, and cosine similarity are suitable for k-NN classifiers [70]. The primary control parameters for k-NN are k and the distance function shown in Table 1. To ensure simplicity and fairness in comparisons with other methods, these parameters are not tuned for the tests carried out. Particularly, $k = 1$ and the Euclidean distance function stated as follows are employed for the k-NN classifier in this work:

$$D(x_{new}, x_{near}) = \sqrt{\sum_{i=1}^{d} (x_{new}^i - x_{near}^i)^2} \tag{9}$$

where x_{near} is a point close to x_{new} in d-space and D(x_{new}, x_{near}) is the Euclidean distance between them.

TABLE 1. Table of control parameters or hyperparameters.

Method	Hyperparameters
QDA	Not Applicable (N/A)
GNB	Not Applicable (N/A)
CART	Maximum number of splits = 100
k-NN	k=1 Distance Function: Euclidean

V. EXPERIMENTAL SETUP

The built-in functions ''fitcdiscr'', ''fitcnb'', ''fitctree'', and ''fitcknn'' (available in the MATLAB software) have been used to implement QDA, GNB, CART and k-NN, respectively. To determine the control parameters for all methods classifiers, where applicable, MATLAB's default settings are

used as shown in Table 1. It is assumed that these default settings in MATLAB are unlikely to be altered by most network engineers, not experienced in machine learning. Generally, machine learning algorithms or methods perform better when their control parameters or hyperparameters are tuned or optimized. However, since hyperparameter optimization is not the goal of this work, this is not investigated. To examine the predictive accuracy of the fitted models from all methods, the validation scheme used is cross-validation. The choice of this validation scheme is guided by the total number of observations (less than 10,000). Following the popular approach for typical cross-validation schemes used to address classification problems [71], a five-fold nested cross-validation has been used by all the classifiers for all experiments.

To compare the classification models constructed and trained by all methods, the validation accuracy, training time, and prediction speed of all methods are used over 50 independent statistical runs which construct and train a model for each method in each run. These statistical runs allow for statistical analysis and comparisons of the efficiencies and robustness of the methods. Also, since the sample size is sufficiently large (i.e. 50 in this case), a z-statistic can be used to estimate the probability values for hypothesis tests (see Section VI-E). All experiments have been carried out on a workstation with Intel 6-core i7-8700 3.20 GHz CPU and 32.0 GB RAM, except where stated otherwise. Elapsed times reported are elapsed real times.

A. DATASET AND CLASSIFICATION PROBLEM

ML is inherently data-driven; ML algorithms work by learning information directly from data without depending on any predetermined relation as a model. As a result, ML algorithms tend to improve their performance as the number of data samples used for learning increases. Naturally, this makes ML models from larger training datasets often generalize well for new data.

To demonstrate the feasibility of the ML algorithms employed in our work to address a classification problem, an overly large dataset is not necessarily required. This is because, conventionally, the practicality of these classifiers have been demonstrated using popular datasets in literature with fewer observations or records per class or category. For example, the dataset from [78] which is well-known in the ML community for testing classifiers [79], [80], contains a set of 150 records with five features and three classes (i.e., 50 observations per class), as opposed to our work, where 3,600 records (900 observations per class) have been used.

For the classification problem, the primary predictors are the throughput (T_p), jitter (J_t) and response time (R_t) for four classes of SDN events or scenarios (i.e. 'Normal', 'HTTP', 'UDP', and 'TCP' classes). The 900 samples or observations used for each class (i.e., 900 observations for Normal, 900 observations for HTPP, 900 observations for UDP and 900 observations TCP for a total of 3600 independent observations) offer a low-cost ML implementation and balanced dataset for better accuracy in our work. Each observation in the dataset is a vector associated with a specific class of SDN event or scenario. The dataset can be expressed as follows:

$$U_{SDN} = \{T_p^{o_1}, J_t^{o_1}, R_t^{o_1}, \ldots, T_p^{o_{3600}}, J_t^{o_{3600}}, R_t^{o_{3600}}\} \quad (10)$$

where U_{SDN} can be viewed mathematically as the universal dataset for all the 3600 independent observations and o_i is the i^{th} independent observation in the dataset for $i \in [1, 3600]$. As a result, there are four classes of SDN events or scenarios to which each observation or vector element (i.e., $T_p^{o_i}$, $J_t^{o_i}$, $R_t^{o_i}$) in U_{SDN} may be associated with exclusively (i.e. each observation belongs to only one class exclusively and can be viewed mathematically as datasets of U_{SDN}).

As discussed in III, U_{SDN} is an emulated dataset. The method used to generate U_{SDN} has been shown to be representative of real-world SDN scenarios and the metrics in U_{SDN} have been studied in terms of their sensitivity based on extensive experiments [72]. Note that it is not unconventional practice to use emulated network scenarios to typify real-world SDN scenarios in terms of traffic behaviour and other metrics [75]. Even though such virtual networks must be validated using real-world network data (i.e., PCAPs) to have purpose-built methods that generalize well to legitimate and malicious traffics [75], this is not the goal of our work. A demonstration of the general usage of popular classifiers using a case involving the detection and classification of DDoS flooding attacks by considering T_p, J_t and R_t metrics from an emulated SDN is the focus of this paper. More details on the classes or types of SDN events and scenarios which constitute the classes (i.e. 'Normal', 'TCP', 'UDP' and 'HTTP') for the classification problem addressed in this work can be found in [72].

The dataset for the SDN classes of events are described as follows:

$$U_{SDN}^{Normal} = \{T_P^{o_1}, J_T^{o_1}, R_T^{o_1}, \ldots, T_P^{o_{900}}, J_T^{o_{900}}, R_T^{o_{900}}\} \quad (11)$$

$$U_{SDN}^{TCP} = \{T_P^{o_{901}}, J_T^{o_{901}}, R_T^{o_{901}}, \ldots, T_P^{o_{1800}}, J_T^{o_{1800}}, R_T^{o_{1800}}\} \quad (12)$$

$$U_{SDN}^{UDP} = \{T_P^{o_{1801}}, J_T^{o_{1801}}, R_T^{o_{1801}}, \ldots, T_P^{o_{2700}}, J_T^{o_{2700}}, R_T^{o_{2700}}\} \quad (13)$$

$$U_{SDN}^{HTTP} = \{T_P^{o_{2701}}, J_T^{o_{2701}}, R_T^{o_{2701}}, \ldots, T_P^{o_{3600}}, J_T^{o_{3600}}, R_T^{o_{3600}}\} \quad (14)$$

where U_{SDN}^{Normal} is the data subset containing all 900 independent observations associated with the normal operating condition of the SDN (i.e. 'Normal' class), U_{SDN}^{TCP} is the dataset containing all 900 independent observations associated with the TCP flooding attack of the SDN (i.e. 'TCP' class), U_{SDN}^{UDP} is the dataset containing the 900 independent observations for the UDP flooding attack of the SDN (i.e. 'UDP' class) and U_{SDN}^{HTTP} is the dataset containing the 900 independent observations for HTTP flooding attack of the SDN (i.e. 'HTTP' class).

The datasets described above are used to build and train classification models to address the multinominal classification problem shown in Figure 4 for any instantaneous

observation (o_i) on the SDN. In a way, this constitutes an offline classification. In practice, ML or prediction models for SDN monitoring are built using collected or historical SDN data, which are mostly populated in repositories [76], [77]. As a result, most robust approaches hybridize offline and online methods whereby prediction models built using static data (typically, from repositories of past SDN operations) are regularly updated with new data from the current operations of the SDNs to improve the accuracy of the prediction models [81].

The hybrid approach mentioned above is not within the scope of our work. To serve as a paradigm for the SDN research community, our work mainly focuses on demonstrating the feasibility of employing popular classifiers in the detection and classification of DDoS flooding attacks in SDNs. Hence, emulated SDN data (U_{SDN}) that would normally be available from a repository of SDN operations has been used in our work.

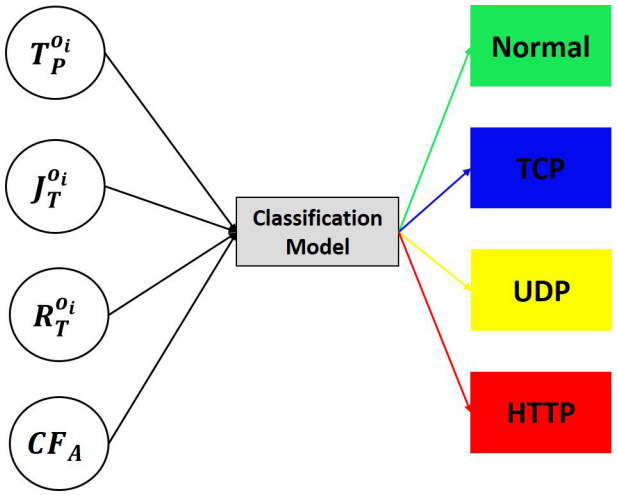


FIGURE 4. Classification problem using four predictors.

B. DATA PRE-PROCESSING AND FEATURE ENGINEERING

Feature engineering primarily involves the process of transforming raw data sets into data sets with highly informative feature or features with high predictive power [73]. Depending on the nature of the raw data and the intended prediction or classification, there are several methods available for feature engineering. These methods include (but are not limited to), binning or bucketing, one-hot encoding, normalization, standardization and data imputation [73]. For our experiments, the raw datasets are standardized using the z-scores of their respective data points. An adaptive univariate mathematical cost function CF_A is also engineered for all independent observations as an additional predictor giving a total of four predictors as shown in Figure 4.

The z-scores for the raw sample datasets are calculated as follows [73]:

$$z = \frac{(x - \bar{x})}{SD} \tag{15}$$

where z is the z-score of a data point x in a given sample data set with mean $\bar{X}$ and standard deviation SD.. Compared with the min-max normalization often used to normalize network parameters [72], standardization (z-score normalization) is preferred in this work since the choice of the classifiers is largely influenced by the assumption that the values in the features of the datasets have a normal distribution or are distributed close to a normal distribution [65]. By inspecting the descriptive statistics of the z-scores shown in Figure 5, the following a posteriori inference can be made for the tendencies of the z-scores when the SDN is operating normally (i.e. not under attack):

$$\text{z-scores ('Normal SDN scenarios')} \rightarrow \begin{cases} +T_P^{o_i,z} \\ -J_T^{o_i,z} \\ -R_T^{o_i,z} \end{cases} \tag{16}$$

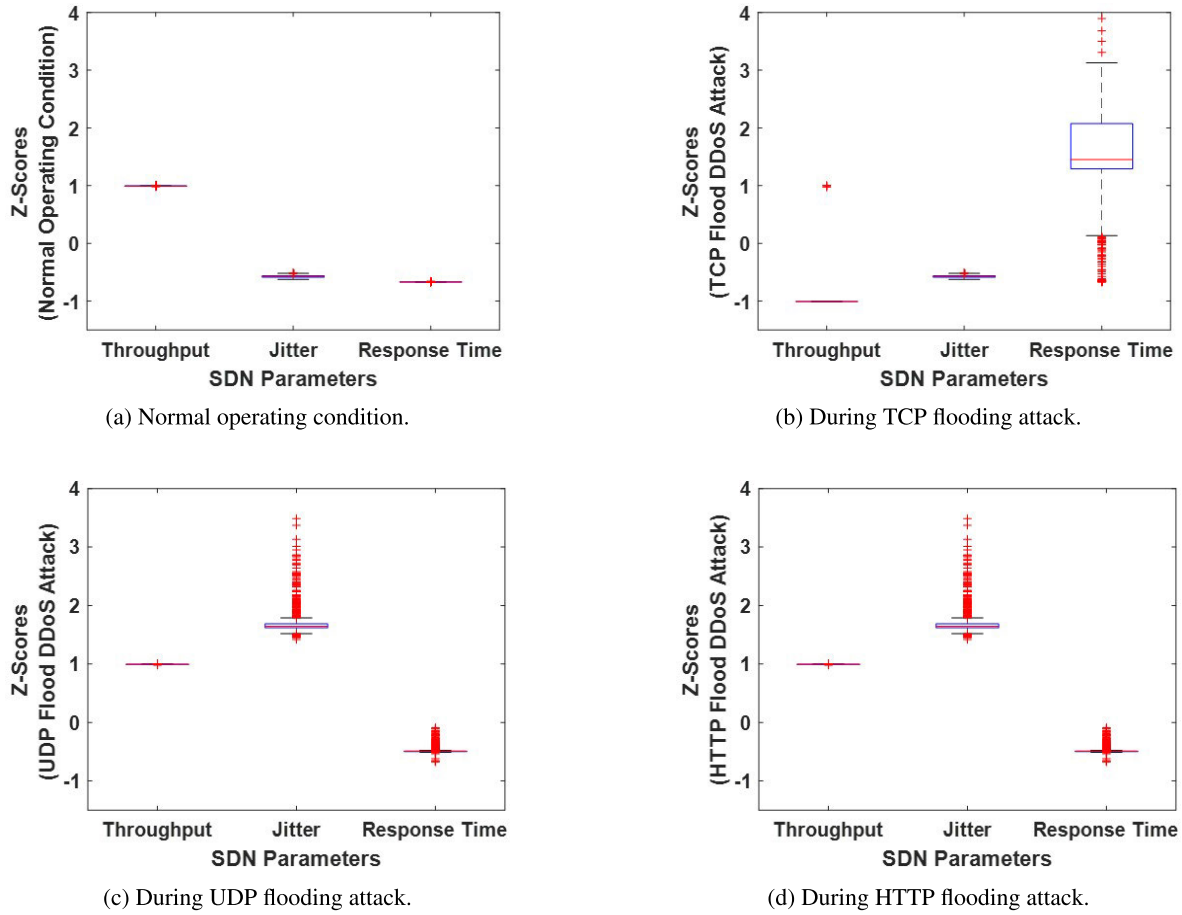
where $T_P^{o_i,z}$, $J_T^{o_i,z}$ and $R_T^{o_i,z}$ are the z-scores for $T_P^{o_i}$, $J_T^{o_i}$, and $R_T^{o_i}$, respectively. Figure 5 also reveal that the sign conventions of $T_P^{o_i,z}$, $J_T^{o_i,z}$ and $R_T^{o_i,z}$ vary with the class or type of SDN attack. In other words, a sign-based feature (CF_A) may be engineered for another predictor that complements the existing predictors.

Generally, model performance assessment in ML is presented in terms of recall and precision. Recall is the ratio of the correct positive predictions to the total of positive observations in the test set (set of observations excluded from the training or learning), while precision is the ratio of correct positive predictions to the total of positive predictions [73]. For the classification problem addressed in our work, a high precision is desired to ensure that all attack scenarios are classified as attack scenarios to guarantee the availability and non-disruption of the SDN. However, a lower recall can be tolerated from a practical viewpoint. This is because normal network events that are seldom classified as attacks on the SDN by the ML model will not disrupt the SDN and such tolerable misclassifications can be easily ratified by network administrators or operators after subsequent investigations.

Since ML models are only as reliable as the data and methods used to train them, spurious correlations that are interpretable by human analysts upon examination can be expected. So in practice, a trade-off between high precision and high recall is always expected because it is typically impossible to have an optimality of both [73]. Hyperparameter tuning which is not within the scope of this paper is one of a number of ways to optimize (maximize) either the precision or the recall of the validation set (set of observations used to tune or optimize the hyperparameters). As a workaround in this work, a mathematically deduced cost function (CF_A) that mimics an artificial boundary between normal events and attack events is introduced and employed as a predictor in the classification problem.

To ensure (CF_A) distinguishes between the SDN events reasonably and non-discriminatorily (see Figure 6), (CF_A) is derived as follows based on the a posteriori inferences made from Figures 5:

$$CF_A = (T_P^{o_i,z} - (J_T^{o_i,z} + R_T^{o_i,z})) \times w \tag{17}$$



(a) Normal operating condition.

(b) During TCP flooding attack.

(c) During UDP flooding attack.

(d) During HTTP flooding attack.

FIGURE 5. Box plots showing the distribution of standardized scores from the dataset.

where $w \in (0,1)$ is a uniformly distributed random weight used to adaptively penalize CF_A according to the operating condition the ith SDN event or scenario is associated with. w assumes its values according to the following criterion:

$$w = \begin{cases} 0 < w < 0.5; & \text{If SDN scenario is 'Normal'.} \\ 0.5 < w < 1; & \text{Otherwise.} \end{cases} \quad (18)$$

A plot of CF_A against SDN events or scenarios is given in Figure 6 showing that $CF_A \to +$ when the SDN is operating normally (i.e. not under attack) and $CF_A \to -$ when the SDN is under attack. This observation is clearly consistent with the intention of adopting CF_A as a predictor (see Figure 4) that can create an artificial boundary between normal and attack events or scenarios. From Figure 6, it can be seen that $CF_A = 0$ imitates such a boundary.

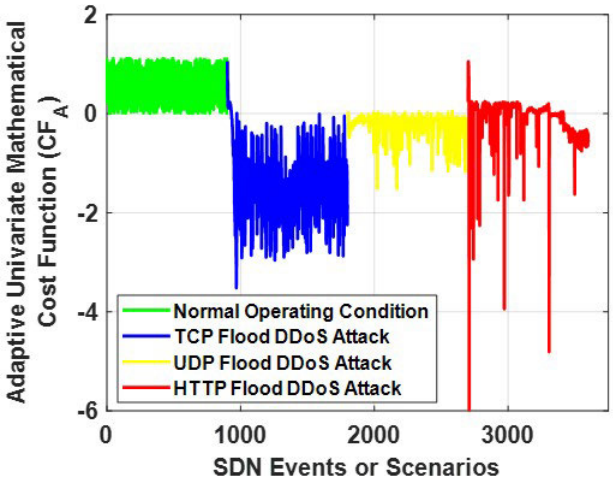


FIGURE 6. Plot of CF_A against SDN events.

VI. RESULTS AND DISCUSSIONS

A. PREDICTION OR VALIDATION ACCURACY

The prediction accuracy of a classifier is usually a ratio of correctly classified observations to the total number of classified observations. The prediction accuracy of all methods can be inferred from their respective confusion matrices. Figure 7 shows the confusion matrices for all methods for a typical independent run. From a given confusion matrix, the prediction accuracy ($Pred_A$) of the classifier can be estimated as:

$$Pred_A = \frac{TP + TN}{TP + TN + FP + FN} \quad (19)$$

where TP is the true positive (i.e. observations classified to be X^* that were actually X^*), TN is the true negative

TABLE 2. Prediction or validation accuracy for all methods (over 50 statistical runs).

Method	Best	Worst	Average	Median	SD
CART	98.9%	98.4%	98.7%	98.7%	1.1e-3
GNB	96.2%	95.9%	96.1%	96.1%	6.1e-4
k-NN	96.0%	95.3%	95.6%	95.6%	1.6e-3
QDA	96.1%	95.6%	95.9%	95.9%	9.3e-4

(i.e. observations classified not to be X^* and were not X^*), FP is the false positive (i.e. observations classified to be X^*, but were not X^*) and FN is the false negative (i.e. observations classified not to be X^*, but were actually X^*).

Since all methods considered return confidence scores of predictions (i.e. posterior probabilities), their prediction accuracies' summaries can also be viewed using their receiver operating characteristic (ROC) curves and the areas under the ROC curves (AUCs). Figure 7 shows ROC curves for all methods for a typical independent run. For a given ROC curve, the prediction accuracy summary is provided using a combination of the true positive rate (TPR, also called the recall) and false positive rate (FPR) and by discretizing the confidence scores of the classification model. The discrete scores are then used as prediction thresholds to predict the classes of the observations in the data set.

For a typical ROC curve, TPR and FPR are estimated as follows:

$$\text{TPR or Recall} = \frac{TP}{TP + FN} \tag{20}$$

$$\text{FPR} = \frac{FP}{FP + TN} \tag{21}$$

From Table 2, the following observations are made for the prediction or validation accuracies of all methods: (1) The prediction or validation accuracies of all methods are generally higher than 95% for the classification problem. (2) CART shows the best average and median prediction or validation accuracies of 98.7% each, over the 50 independent runs. (3) GNB and QDA perform similarly based on their nearly equal average prediction or validation accuracies of 96.1% and 95.9%, respectively, and their nearly equal median prediction or validation accuracies of 96.1% and 95.9%, respectively, over the 50 independent runs. (4) k-NN obtains the lowest average and median prediction or validation accuracies of 95.6%each, over the 50 independent runs.

All methods show very good robustness for the classification problem considering their very low standard deviations over the 50 independent runs, as shown in Table 2. From Table 2, it can be seen that: (1) QDA has the least standard deviation of 9.3e-4 over all the 50 independent runs. (2) k-NN has the highest standard deviation of 1.6e-3 over the 50 independent runs. Hence, in terms of prediction or validation accuracy robustness for the classification problem, QDA and k-NN are the most and least robust methods, respectively, compared to other methods.

B. TRAINING TIME

Training time is the time taken by each method to build and train a classification model. From Table 3, the following observations are made for the training times of all methods: (1) Training times of all methods are generally more than 12 ms for the classification problem. (2) CART shows the best average and median training times of 12.4 ms and 12.4 ms, respectively, over the 50 independent runs. (3) k-NN and QDA perform similarly based on their near-equal average and median training times of 12.8 ms and 12.7 ms, respectively, over the 50 independent runs. (4) GNB shows the worst average and median training times of 16.9 ms each, over all 50 independent runs.

All methods show very good robustness for the classification problem based on their very low standard deviations over all runs as shown in Table 3, from which it can be seen that: (1) CART has the lowest standard deviation of 4.5e-4. (2) k-NN has the highest standard deviation of 4.5e-4. Hence, in terms of training time stability for the classification problem, CART is the most robust method, and k-NN is the least robust method.

TABLE 3. Training time (ms) for all methods (over 50 statistical runs).

Method	Best	Worst	Average	Median	SD
CART	12.1	13.6	12.4	12.2	3.3e-4
GNB	16.4	18.2	16.9	16.9	3.6e-4
k-NN	12.4	14.2	12.8	12.7	4.5e-4
QDA	12.2	14.2	12.8	12.7	3.4e-4

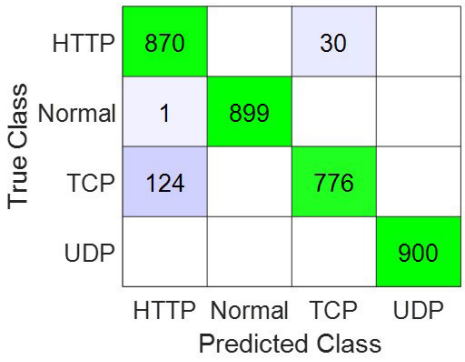
C. PREDICTION SPEED

The prediction speed is the total number of predictions or classifications made by a method divided by the time taken to make the predictions. It is measured in observations per second (obs/s). Table 4 allows the following observations for the prediction speeds of all methods: (1) The prediction speeds of all methods are generally higher than 2.6e5 obs/s for the classification problem. (2) CART shows the best average and median prediction speeds of 5.3e5 obs/s and 5.3e5 obs/s, respectively, over all runs. (3) GNB (average of 4.1e5 obs/s and median of 4.2e5 obs/s) performs better than QDA (average and median of 3.5e5 obs/s each). (4) k-NN obtains the lowest average and median prediction speeds of 2.9e5 obs/s each.

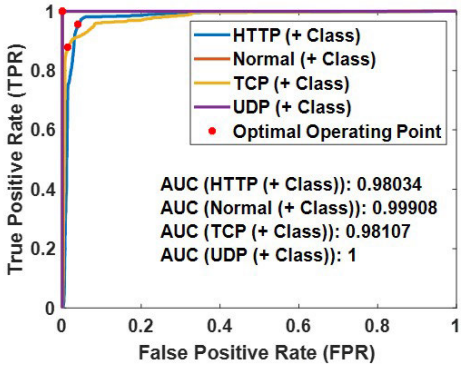
All methods show very good robustness for the classification problem based on their very low standard deviations over all runs, as shown in Table 4. Here, it can be seen that: (1) CART has the least standard deviation of 1.0e-4. (2) k-NN has the most standard deviation of 8.8e-4. Hence, in terms of prediction speed stability for the classification problem, CART is the most robust method and k-NN the least robust method, compared to other methods.

D. RANKING

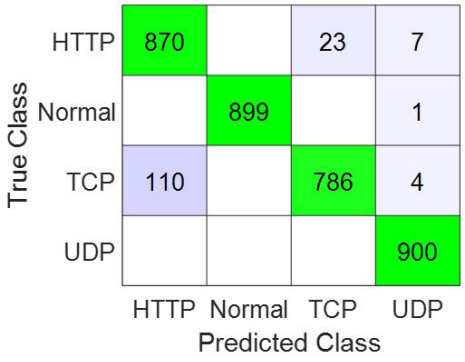
To quickly rank all methods inferentially, rewards are assigned to each method according to their average performances for prediction accuracy, training time and prediction speed. Points are assigned: 4, 3, 2, 1 in descending order of performance. Table 5 suggests how, the total sum of points obtained by each method can be used to deduce the overall ranking.



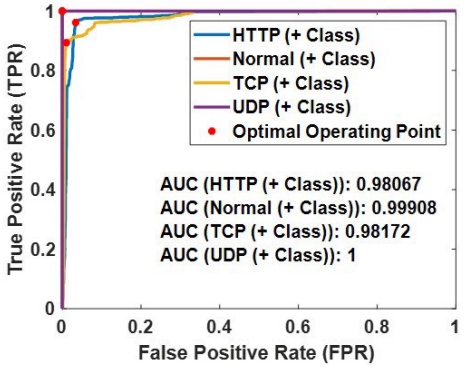
(a) Confusion matrix for QDA.



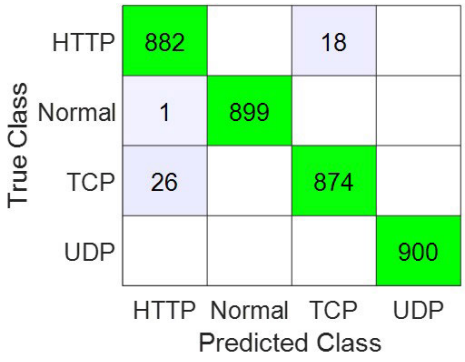
(b) ROC curve for QDA.



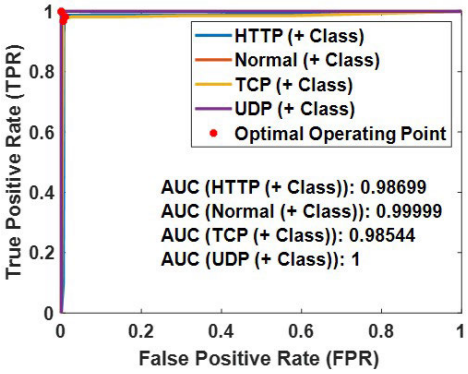
(c) Confusion matrix for GNB



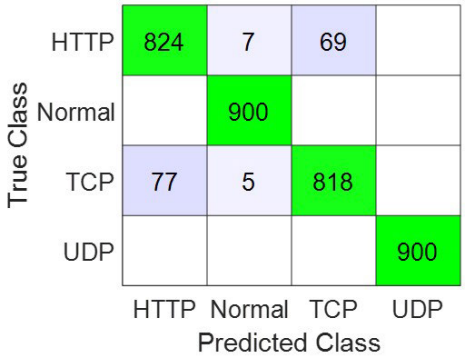
(d) ROC curve for GNB.



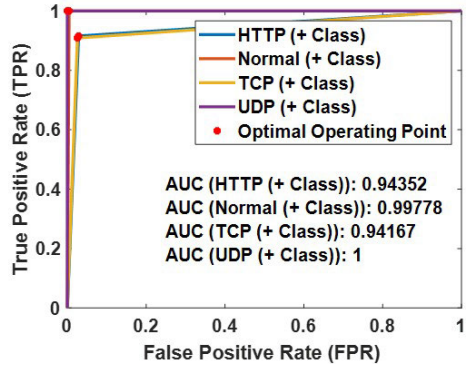
(e) Confusion matrix for CART.



(f) ROC curve for CART.



(g) Confusion matrix for k-NN.



(h) ROC curve for k-NN.

FIGURE 7. Typical confusion matrices and ROC curves for all methods.

TABLE 4. Prediction speed (obs/s) for all methods (over 50 statistical runs).

Method	Best	Worst	Average	Median	SD
CART	5.4e5	4.0e5	5.3e5	5.4e5	1.0e-4
GNB	4.2e5	3.9e5	4.1e5	4.2e5	7.5e-3
k-NN	3.0e5	2.6e5	2.9e5	2.9e5	8.8e-3
QDA	3.6e5	3.2e5	3.5e5	3.5e5	6.9e-3

TABLE 5. Ranking of all methods.

Method	Prediction Accuracy	Prediction Speed	Training Time	Overall Rank
CART	Rank 1 (4 points)	Rank 1 (4 points)	Rank 1 (4 points)	Rank 1 (12 points)
GNB	Rank 2 (3 points)	Rank 2 (3 points)	Rank 4 (1 point)	Rank 2 (7 points)
k-NN	Rank 4 (1 point)	Rank 4 (1 point)	Rank 2 (3 points)	Rank 4 (5 points)
QDA	Rank 3 (2 points)	Rank 3 (2 points)	Rank 2 (3 points)	Rank 2 (7 points)

TABLE 6. Hypothesis test: CART vs other methods.

Method	Prediction Accuracy (p-values)	Prediction speed (p-values)	Training Time (p-values)
GNB	5.9e-18	7.1e-18	7.1e-18
k-NN	6.6e-18	7.1e-18	2.1e-18
QDA	6.5e-18	7.1e-18	4.0e-18

Table 5 indicates that CART ranks first with a total or 12 points, GNB and QDA both rank second with seven, and k-NN ranks fourth with five. This suggests that the overall performance of CART is much better than QDA, GNB and k-NN for this classification problem based on the settings used. To statistically verify that CART's overall performance ranks better than the other methods, hypothesis tests are carried out in the next subsection using the Wilcoxon test [74].

E. HYPOTHESIS TEST

To carry out the hypothesis test (i.e. Wilcoxon test [74]), the results obtained by all methods for prediction accuracy, prediction speed and training time over the 50 independent statistical runs are used as data samples. The null hypothesis is that the data samples of CART and the other methods (i.e. GNB, k-NN and QDA) have equal medians at 5% significance level (i.e. 95% confidence level). If the resultant probability value (i.e. p-value) of the hypothesis test is less than or equal to 0.05, a strong evidence exists against the null hypothesis and it is therefore rejected.

The results for the hypothesis tests are shown in Table 6. According to the p-values in Table 6, the null hypothesis is rejected in all the cases. This shows that the prediction accuracy, prediction speed and training time of CART is significantly better than the prediction accuracy, prediction speed and training time of other methods. Therefore, the ranking earlier established in Table 5 is statistically verified.

VII. CONCLUSION

Machine learning-based detection and classification of flooding DDoS attacks has been implemented and investigated in this paper using four popular supervised learning methods (GNB, QDA, k-NN and CART). All methods performed well and show suitability for application in the classification and detection of such attacks. However, in terms of stability, prediction accuracy, training time, and prediction speed, CART outperforms others based on the investigations carried out. It is good to note that for the methods having control parameters (e.g., CART and k-NN), their hyperparameters have not been tuned or optimized because hyperparameter tuning and optimization is not within the scope of this work. In future, the relative usefulness of the predictors and the engineered feature will be investigated using computational intelligence techniques such as global sensitivity analysis and parallel coordinates. The hyperparameters of some of the methods will also be investigated for tuning and optimization, and a larger dataset (e.g. over 10,000 observations) will be generated to further investigate how these machine learning algorithms or methods can assist in the expedited detection and classification of DDoS flooding attacks. It is envisaged that future investigations will culminate in the design and development of a machine learning-based add-on that complements existing software tools for DDoS flooding attack detection and classification.

REFERENCES

[1] D. Puthal and X. Zhang, "Secure computing for the Internet of Things and network edges: Protecting communication in the worldwide network of devices," *IEEE Consum. Electron. Mag.*, vol. 7, no. 6, pp. 29–30, Nov. 2018.
[2] T. Alam, *A Reliable Communication Framework and Its Use in Internet of Things (IoT)*, document CSEIT1835111| Received 10 2018, pp. 450–456.
[3] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future internet," *Comput. Netw.*, vol. 75, pp. 453–471, Dec. 2014.
[4] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.
[5] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "GENI: A federated testbed for innovative network experiments," *Comput. Netw.*, vol. 61, pp. 5–23, Mar. 2014.
[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.
[7] A. K. Sarica and P. Angin, "Explainable security in SDN-based IoT networks," *Sensors*, vol. 20, no. 24, p. 7326, Dec. 2020.
[8] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.
[9] A. Sangodoyin, T. Sigwele, P. Pillai, Y. F. Hu, I. Awan, and J. Disso, "DoS attack impact assessment on software defined networks," in *Proc. Int. Conf. Wireless Satell. Syst.* Cham, Switzerland: Springer, 2017, pp. 11–22.
[10] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 36–43, Jul. 2013.
[11] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 55–60.
[12] D. K. Bhattacharyya and J. K. Kalita, *DDoS Attacks: Evolution, Detection, Prevention, Reaction, and Tolerance*. Boca Raton, FL, USA: CRC Press, 2016.
[13] Y. Cao, Y. Gao, R. Tan, Q. Han, and Z. Liu, "Understanding internet DDoS mitigation from academic and industrial perspectives," *IEEE Access*, vol. 6, pp. 66641–66648, 2018.
[14] M. Cirillo, M. D. Mauro, V. Matta, and M. Tambasco, "Botnet identification in DDoS attacks with multiple emulation dictionaries," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3554–3569, 2021.

[15] B. Liao, Y. Ali, S. Nazir, L. He, and H. U. Khan, "Security analysis of IoT devices by using mobile computing: A systematic literature review," *IEEE Access*, vol. 8, pp. 120331–120350, 2020.
[16] J. Ashraf and S. Latif, "Handling intrusion and DDoS attacks in software defined networks using machine learning techniques," in *Proc. Nat. Softw. Eng. Conf.*, Nov. 2014, pp. 55–60.
[17] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "DDoS attack protection in the era of cloud computing and software-defined networking," *Comput. Netw.*, vol. 81, pp. 308–319, Apr. 2015.
[18] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS attack detection and mitigation using SDN: Methods, practices, and solutions," *Arabian J. Sci. Eng.*, vol. 42, no. 2, pp. 425–441, Feb. 2017.
[19] K. S. Sahoo, D. Puthal, M. Tiwary, J. J. P. C. Rodrigues, B. Sahoo, and R. Dash, "An early detection of low rate DDoS attack to SDN based data center networks using information distance metrics," *Future Gener. Comput. Syst.*, vol. 89, pp. 685–697, Dec. 2018.
[20] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39–53, Apr. 2004.
[21] R. T. Kokila, S. Thamarai Selvi, and K. Govindarajan, "DDoS detection and analysis in SDN-based environment using support vector machine classifier," in *Proc. 6th Int. Conf. Adv. Comput. (ICoAC)*, Dec. 2014, pp. 205–210.
[22] C. Li, Y. Wu, X. Yuan, Z. Sun, W. Wang, X. Li, and L. Gong, "Detection and defense of DDoS attack-based on deep learning in OpenFlow-based SDN," *Int. J. Commun. Syst.*, vol. 31, no. 5, p. e3497, Mar. 2018.
[23] C. Yoon, T. Park, S. Lee, H. Kang, S. Shin, and Z. Zhang, "Enabling security functions with SDN: A feasibility study," *Comput. Netw.*, vol. 85, pp. 19–35, Jul. 2015.
[24] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area internet traffic patterns and characteristics," *IEEE Netw.*, vol. 11, no. 6, pp. 10–23, Nov./Dec. 1997.
[25] G. Carl, G. Kesidis, R. R. Brooks, and S. Rai, "Denial-of-service attack-detection techniques," *IEEE Internet Comput.*, vol. 10, no. 1, pp. 82–89, Jan. 2006.
[26] S. Yu, Y. Tian, S. Guo, and D. O. Wu, "Can we beat DDoS attacks in clouds?" *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2245–2254, Sep. 2014.
[27] D. Gavrilis and E. Dermatas, "Real-time detection of distributed denial-of-service attacks using RBF networks and statistical features," *Comput. Netw.*, vol. 48, no. 2, pp. 235–245, Jun. 2005.
[28] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep recurrent neural network for intrusion detection in SDN-based networks," in *Proc. 4th IEEE Conf. Netw. Softwarization Workshops (NetSoft)*, Jun. 2018, pp. 202–206.
[29] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2008.
[30] K. Lee, J. Kim, K. H. Kwon, Y. Han, and S. Kim, "DDoS attack detection method using cluster analysis," *Expert Syst. Appl.*, vol. 34, no. 3, pp. 1659–1665, Apr. 2008.
[31] P. Xiao, W. Qu, H. Qi, and Z. Li, "Detecting DDoS attacks against data center with correlation analysis," *Comput. Commun.*, vol. 67, pp. 66–74, Aug. 2015.
[32] P. Dong, X. Du, H. Zhang, and T. Xu, "A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.
[33] R. Wang, Z. Jia, and L. Ju, "An entropy-based distributed DDoS detection mechanism in software-defined networking," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, vol. 1, Aug. 2015, pp. 310–317.
[34] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2015, pp. 77–81.
[35] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*. Berlin, Germany: Springer, 2011, pp. 161–180.
[36] M. Kuerban, Y. Tian, Q. Yang, Y. Jia, B. Huebert, and D. Poss, "FlowSec: DOS attack mitigation strategy on SDN controller," in *Proc. IEEE Int. Conf. Netw., Archit. Storage (NAS)*, Aug. 2016, pp. 1–2.
[37] S. Lim, S. Yang, Y. Kim, S. Yang, and H. Kim, "Controller scheduling for continued SDN operation under DDoS attacks," *Electron. Lett.*, vol. 51, no. 16, pp. 1259–1261, Aug. 2015.
[38] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Proc. IEEE Local Comput. Netw. Conf.*, Oct. 2010, pp. 408–415.
[39] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, and J. Peng, "XGBoost classifier for DDoS attack detection and analysis in SDN-based cloud," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Jan. 2018, pp. 251–256.
[40] *Team Mininet*. Accessed: Nov. 21, 2018. [Online]. Available: http://www.mininet.org/download/
[41] *Floodlight*. Accessed: Nov. 21, 2018. [Online]. Available: http://www.projectfloodlight.org/Floodlight controller
[42] *Praetox Technologies Low Orbit Ion Cannon*. Accessed: Nov. 21, 2018. [Online]. Available: https://github.com/NewEraCracker/LOIC/
[43] A. Sangodoyin, B. Mohammed, M. Sibusiso, I. Awan, and J. P. Disso, "A framework for distributed denial of service attack detection and reactive countermeasure in software defined network," in *Proc. 7th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2019, pp. 80–87.
[44] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Secur. Privacy*, May 2010, pp. 305–316.
[45] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*. Berlin, Germany: Springer-Verlag, 2003, pp. 220–237.
[46] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln laboratory," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, 2000.
[47] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Inf. Secur. J.: Global Perspective*, vol. 25, nos. 1–3, pp. 18–31, Apr. 2016.
[48] S. Behal and K. Kumar, "Characterization and comparison of DDoS attack tools and traffic generators: A review," *IJ Netw. Secur.*, vol. 19, no. 3, pp. 383–393, 2017.
[49] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "SPHINX: Detecting security attacks in software-defined networks," in *Proc. NDSS*, vol. 15, 2015, pp. 8–11.
[50] G. J. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*, vol. 544. Hoboken, NJ, USA: Wiley, 2004.
[51] J. Praagman, *Classification and Regression Trees* (Stone the Wadsworth Statistics/Probability Series), B. Leo, H. F. jerome, A. O. Richard, and J. Charles, Eds. Wadsworth, OH, USA: Belmont, 1984, p. 358.
[52] O. Z. Maimon and L. Rokach, *Data Mining With Decision Trees: Theory and Applications*, vol. 81. Singapore: World Scientific, 2014.
[53] D. G. Stork, R. O. Duda, P. E. Hart, and D. Stork, *Pattern Classification*. Hoboken, NJ, USA: Wiley, Nov. 2000.
[54] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Mach. Learn.*, vol. 29, no. 2, 1997, pp. 131–163.
[55] M. Bressan and J. Vitrià, "Nonparametric discriminant analysis and nearest neighbor classification," *Pattern Recognit. Lett.*, vol. 24, no. 15, pp. 2743–2749, Nov. 2003.
[56] S. Guo and H. Tracey, "Discriminant analysis for radar signal classification," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 4, pp. 3134–3148, Aug. 2020.
[57] C. Zhuang, H. Zhao, C. Sun, and W. Feng, "Detection and classification of GNSS signal distortions based on quadratic discriminant analysis," *IEEE Access*, vol. 8, pp. 25221–25236, 2020.
[58] S. Pang, S. Ozawa, and N. Kasabov, "Incremental linear discriminant analysis for classification of data streams," *IEEE Trans. Syst., Man Cybern., B (Cybern.)*, vol. 35, no. 5, pp. 905–914, Oct. 2005.
[59] A. Tharwat, "Linear vs. quadratic discriminant analysis classifier: A tutorial," *Int. J. Appl. Pattern Recognit.*, vol. 3, no. 2, p. 145, 2016.
[60] S. J. Prince, *Computer Vision: Models, Learning, and Inference*. Cambridge, U.K.: Cambridge Univ. Press, 2012.
[61] H. Yan and Y. Dai, "The comparison of five discriminant methods," in *Proc. Int. Conf. Manage. Service Sci.*, Aug. 2011, pp. 1–4.
[62] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning*, vol. 1. New York, NY, USA: Springer, 2001.
[63] L. Jiang, L. Zhang, C. Li, and J. Wu, "A correlation-based feature weighting filter for Naive Bayes," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 201–213, Feb. 2019.
[64] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, vol. 31. New York, NY, USA: Springer, 2013.

[65] C. Forbes, M. Evans, N. Hastings, B. Peacock, *Statistical Distributions*. Hoboken, NJ, USA: Wiley, 2011.
[66] A. McCallum and K. Nigam, "A comparison of event models for Naive Bayes text classification," in *Proc. AAAI Workshop Learn. Text Categorization*, vol. 752, 1998, pp. 41–48.
[67] S. Gey and E. Nedelec, "Model selection for CART regression trees," *IEEE Trans. Inf. Theory*, vol. 51, no. 2, pp. 658–670, Feb. 2005.
[68] S. A. Dudani, "The distance-weighted K-nearest-neighbor rule," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, no. 4, pp. 325–327, Apr. 1976.
[69] X. Gao and G. Li, "A KNN model based on Manhattan distance to identify the SNARE proteins," *IEEE Access*, vol. 8, pp. 112922–112931, 2020.
[70] L.-Y. Hu, M.-W. Huang, S.-W. Ke, and C.-F. Tsai, "The distance function effect on K-nearest neighbor classification for medical datasets," *SpringerPlus*, vol. 5, no. 1, p. 1304, Dec. 2016.
[71] P. Memar and F. Faradji, "A novel multi-class EEG-based sleep stage classification system," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 26, no. 1, pp. 84–95, Jan. 2018.
[72] A. O. Sangodoyin, M. O. Akinsolu, and I. Awan, "A deductive approach for the sensitivity analysis of software defined network parameters," *Simul. Model. Pract. Theory*, vol. 103, Sep. 2020, Art. no. 102099.
[73] A. Burkov, *The Hundred-Page Machine Learning Book*, vol. 1. Québec, QC, Canada: Andriy Burkov, 2019.
[74] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in Statistics*. New York, NY, USA: Springer, 1992, pp. 196–202.
[75] R. U. Rasool, U. Ashraf, K. Ahmed, H. Wang, W. Rafique, and Z. Anwar, "Cyberpulse: A machine learning based link flooding attack mitigation system for software defined networks," *IEEE Access*, vol. 7, pp. 34885–34899, 2019.
[76] W. Queiroz, M. A. M. Capretz, and M. Dantas, "An approach for SDN traffic monitoring based on big data techniques," *J. Netw. Comput. Appl.*, vol. 131, pp. 28–39, Apr. 2019.
[77] D. Javeed, T. Gao, and M. T. Khan, "SDN-enabled hybrid DL-driven framework for the detection of emerging cyber threats in IoT," *Electronics*, vol. 10, no. 8, p. 918, Apr. 2021.
[78] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, Sep. 1936.
[79] X. Mengfan, H. Lei, and P. Dongliang, "A combined algorithm of K-means and MTRL for multi-class classification," *J. Syst. Eng. Electron.*, vol. 30, no. 5, pp. 875–885, Oct. 2019.
[80] I. C. Goknar, M. Yildiz, S. Minaei, and E. Deniz, "Neural CMOS-integrated circuit and its application to data classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 5, pp. 717–724, May 2012.
[81] T. V. Phan, N. K. Bao, and M. Park, "A novel hybrid flow-based handler with DDoS attacks in software-defined networking," in *Proc. Int. IEEE Conf. Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People, Smart World Congr. (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, Jul. 2016, pp. 350–357.

ABIMBOLA O. SANGODOYIN (Member, IEEE) received the B.Sc. degree in electrical engineering from the University of Ibadan, Nigeria, in 2010, and the M.Sc. degree in personal, mobile, and satellite communications and the Ph.D. degree in computer science from the University of Bradford, U.K., in 2013 and 2020, respectively. His Ph.D. degree was in cybersecurity and his Ph.D. thesis focused on design and analysis of anomaly detection and mitigation schemes for DDoS attacks in SDN. He is currently a Postdoctoral Research Fellow in cybersecurity with the University of Wolverhampton. He is also a Registered Electrical Engineer with COREN.

MOBAYODE O. AKINSOLU (Member, IEEE) received the M.Sc. degree with distinction in electrical and electronic engineering from the University of Bradford, U.K., in 2014. After his undergraduate studies and compulsory national service in Nigeria, he received his master's degree. He then worked as a Research Fellow (Industrial Attaché) and a Visiting Researcher at the National Space Research and Development Agency, Nigeria, and the RFID Research Centre, African University of Science and Technology, Nigeria, respectively, until 2016. His Ph.D. degree was in applied artificial intelligence and his Ph.D. thesis focused on electromagnetic design automation using surrogate model-assisted evolutionary algorithms. In 2020, he received a commendation from the University of Chester, U.K. (through the partnership with Wrexham Glyndŵr University) for his publication record (over 20 publications) relating to his Ph.D. work. He is currently a Lecturer (an Assistant Professor) in electronic and communication engineering at Wrexham Glyndŵr University. He is a member of IET and a Registered Electrical Engineer with COREN. From 2016 to 2019, he was a Ph.D. Scholarship Awardee in recognition of a joint project between Wrexham Glyndŵr University, U.K., and the University of Birmingham, U.K.

PRASHANT PILLAI (Senior Member, IEEE) is currently an Award-Winning Academic Leader and a Visionary, who has over 18 years of U.K. Higher Education experience. He is also a Globally Renowned Scholar and an Outstanding University Educator, who provides academic direction and leadership to research, business engagement, and education within the subject area of computer science at the University of Wolverhampton. He is also a Professor of cyber security and the Director of Wolverhampton Cyber Research Institute. The institute comprises of over ten academic staffs and focuses of research in the area of cyber security and intelligent and secure cyber physical systems/the IoT. He is also the Head of the School of Mathematics and Computer Science, University of Wolverhampton. He was recently appointed as a Founding Member of the IEEE Special Interest Group on Big Data for Cyber Security and Privacy. He was recently appointed as the Co-Chair of the 5G-Satellite Subgroup of the IEEE 5G Technology Roadmap Working Group.

VIC GROUT received the degree in mathematics and computing and the Ph.D. degree in engineering. He is currently a Professional Futurist/Futurologist at Wrexham Glyndŵr University, researching emerging and future technologies (artificial intelligence, big data analytics, the Internet of Things, robotics, automation, and high-speed communications) and their wider social (ethical, legal, political, environmental, and economic) impact. With his degrees, his current role is a Professor of computing futures, having previously been a Professor of network algorithms. His past and current projects range from using internet hardware and software to enable the elderly and disabled to lead longer independent lives, through the projects, such as effective online age-gating, medical diagnosis and using artificial intelligence, and machine learning for efficient minefield clearance, to advising the European Commission on ethical concerns relating to funded research projects. He has authored over 400 research articles and four books on these various topics, edits the Turing's Radiator blog on technological philosophy and appears regularly on radio and TV. His first science-fiction novel *Conscious*, in 2017.