DOI: 10.15849/IJASCA.230720.01

Int. J. Advance Soft Compu. Appl, Vol. 15, No. 2, July 2023
Print ISSN: 2710-1274, Online ISSN: 2074-8523
Copyright © Al-Zaytoonah University of Jordan (ZUJ)

An Optimized Network Intrusion Detection System for Attack Detection based on Supervised Machine Learning Models in an Internet-of-Things Environment

Adeeb Alhomoud

College of Science and Theoretical Studies, Saudi Electronic University, Riyadh, Saudi Arabia
e-mail: a.alhomoud@seu.edu.sa

Abstract

In this paper, an optimized classification approach based on a support vector machine (SVM) classifier is proposed to maximize the accuracy of a machine learning model employed by a network intrusion detection system to detect malicious attacks in an Internet-of-Things (IoT) environment. In addition, an experiment study based on the TON_IoT dataset is conducted in terms of classifier performance metrics, such as the false positive rate, true positive rate, precision, F-measure, recall, Matthews correlation coefficient, receiver operating characteristic area, precision-recall curve area, and a confusion matrix. It is demonstrated that the model accuracy is maximized and the number of rounds minimized for the confusion matrix to converge and correctly classify all various attacks. Finally, classifier errors are compared in terms of kernel types. This part of the investigation shows that the SVM classifier based on a polynomial kernel outperformed radial basis function and normalized polynomial kernels in terms of classifier errors, time to build the model, correctly classified instances, incorrectly classified instances, and kappa statistics.

Keywords: *Complexity parameter, confusion matrix, IoT networks, machine learning, NIDS, SVM.*

1 Introduction

The Internet of Things (IoT) is a fast-growing network of interlinked devices that communicate and exchange data over a network or the internet. Such devices can be anything from smart appliances and wearable technology to industrial control systems and are increasingly being used in a variety of settings, including homes, businesses, and industrial environments. However, the wide spread of the IoT in organizations has introduced security challenges, as the devices are often prone to cyber attacks. One of the main security concerns with an IoT in an enterprise is the potential for the unauthorized access to and control of devices. For example, an attacker could gain access to a connected industrial control system and disrupt production [1]. There is also the risk of data breaches and the potential for any sensitive data collected by IoT devices to be accessed by unauthorized parties.

Received 9 March 2023; Accepted 1 June 2023

Cyber security is a critical concern for businesses of all sizes and has become one of the top priorities for organizations due to increasing reliance on networked technology, which is accompanied by its own risks. One key aspect of cyber security is the protection of enterprise networks from unauthorized access and malicious activities. Therefore, many security procedures and practices are applied to secure the network of an organization. One approach to addressing security concerns is to use a network intrusion detection system (NIDS) to detect and prevent attacks on IoT devices [2]. Many approaches, such as supervised, unsupervised, and reinforcement learning in the field of machine learning (ML), have been used to improve the efficiency and accuracy of NIDSs [3, 4]. It remains a challenge, however, for existing intrusion detection algorithms to achieve a good level of performance in detecting malicious network attacks. Hence, the performance of these systems is often limited by the accuracy of the ML models.

This paper proposes an optimized classification approach based on a support vector machine (SVM) classifier to maximize the accuracy of the ML model used by a NIDS-based solution to detect malicious attacks in an IoT environment. In addition, by tuning the SVM classifier parameters, the confusion matrix converges in a minimum number of rounds by correctly classifying all attacks in the IoT network. An experiment study is also performed in terms of classifier performance metrics, including the true positive rate (TPR), false positive rate (FPR), precision, F-measure, recall, Matthews correlation coefficient (MCC), receiver operating characteristic (ROC) area, precision-recall curve (PRC) area, and a confusion matrix based on the TON_IoT dataset [5]. Classifier errors are also investigated in terms of SVM kernel function, comparing radial basis function, normalized polynomial, and polynomial kernels to select the best kernel function within the SVM classifier.

The rest of the paper is structured as follows. Section 2 presents the literature on intrusion detection system (IDS) types and techniques, the advantages of using ML in IDS systems, and types of IDSs for attack detection in IoT networks. Section 3 provides an overview of common types of attacks on the IoT and the datasets used. Section 4 outlines the proposed model. Section 5 outlines the methodology used, and Section 6 presents the experiments and a discussion of the results. Finally, Section 7 provides the conclusion and findings.

2 Literature Review

In the literature, various types of IDSs for attack detection are well defined and involve signature-based and behaviour-based systems. IDSs are used to identify deviations from normal network behaviour that may indicate malicious activity. Signature-based IDSs rely on a database of known attack and vulnerability signatures but are unable to detect novel malicious activity. These systems trigger an alert when a pattern being analysed matches an entry in the signature database. Behaviour-based IDSs, on the other hand, are designed to detect previously unseen attacks through the use of ML algorithms. Although they may be able to detect unknown attacks in theory, their practical application in real-world networks is limited due to the complexity of development and the high FPR they often produce. In general, IDSs process logs generated by active network equipment, such as computers or routers, to detect anomalous activity, also known as network intrusions. IDSs can also be categorized into those that are network based and host based. A NIDS is usually positioned at network gateways and routers to inspect network traffic for intrusions [6]. Host-based IDSs monitor individual nodes or devices and notify the user if any suspicious activities are detected [7].

Originally, IDSs were designed to process and analyse logs generated by various systems [8]. The same motivation has led to the use of network traffic processing for intrusion detection, with IDSs being the most common application for collecting, analysing, and classifying network traffic content. Several authors have proposed a taxonomy for IDSs that group these systems according to relevant characteristics. IDSs are widely used in network topologies to safeguard the integrity and availability of sensitive assets in protected systems. Although a variety of supervised, unsupervised, and reinforcement learning methods from the field of ML have been employed to improve the effectiveness of IDSs, it remains a challenge for current intrusion detection algorithms to accomplish a good level of performance. One reason for this is the presence of irrelevant and redundant data in high-dimensional datasets, which can interfere with the classification process of an IDS. In addition, individual classifiers may not perform well in detecting every type of attack and many models are based on outdated datasets, making them less adaptable to novel attacks. As technology advances, these attacks continue to pose a threat to the integrity, confidentiality, and availability of cyber systems, making the use of IDSs necessary to protect such systems from a different type of attack [9-11].

IDSs are deployed in several distributed systems to detect malicious activities and take swift countermeasures to stop additional intrusions and spread. IDSs can generally be divided into two categories based on their detection mechanisms: anomaly and misuse detection [12]. Anomaly detection is intended to detect malicious activities by identifying abnormalities from normal behaviour profiles, with better performance in detecting new types of attacks, but can result in a high FPR [13]. On the other hand, based on known patterns, misuse detection is effective at differentiating legitimate instances from malicious ones [14]. A comparison between anomaly and misuse detection is shown in Table 1.

Table 1: Comparison between anomaly and misuse detection

Anomaly detection	Misuse detection
Utilizes the recognized system normal behaviour profile. Discrepancy in the inbound pattern is considered suspicious.	Utilizes a technique for modelling the present well-known attack signature data. Once a match of an inbound pattern with a signature is accomplished, it is considered suspicious.
Difficulty in differentiating an attack from normal behaviour.	Difficulty in keeping an up-to-date attack signature database.
High rates of false positives.	Low rates of false positives.
Unsupervised or semi-supervised ML is suitable for anomaly detection.	Supervised ML methods are useful for misuse detection.
A very common anomaly detection approach is statistical learning.	Signature matching is a very common method in misuse detection.
Good accuracy for unknown attacks.	Very high precision in identifying known attacks.
Examples: Multiagent-based intrusion detection systems (MIDSs)	Examples: Suricata and Snort

Misuse detection can be categorized as being either knowledge based or machine learning based. In knowledge-based misuse detection, network traffic is compared to predefined rules or patterns that represent known attacks. This knowledge-based technique can be further divided into three categories: state transition analysis, signature matching, and rule-based expert systems [15]. However, this approach requires frequent updates to the knowledge database and may not be able to detect new or modified versions of attacks.

Alternatively, misuse detection can also be accomplished using ML techniques, such as a back propagation artificial neural network (BP-ANN) [16], decision trees [17], and multi-class SVMs [18].

IDSs of this type are effective at identifying known attack patterns, but they are not capable of detecting novel or modified versions of known attacks. As cyber criminals become more advanced, new vulnerabilities and threats are emerging at an alarming rate, which poses a significant risk to critical infrastructure. To address this issue, there has been a demand for more advanced IDSs that can detect and respond to novel attacks. As a result, various methods have been researched and developed to improve the performance and accuracy of IDSs in detecting such threats. One of them is ML [19-21], which can be used for both anomaly and misuse detection models. One problem that arose with the initial application of ML in IDSs was the inability of a single classifier to handle this task effectively. As a result, researchers have explored the use of ensemble classifiers as a method of enhancing the performance of IDSs [22, 23]. Ensemble learning is an ML approach that involves combining the predictions of multiple individual classifiers in order to make a more accurate classification decision about the input object. The primary objective of ensemble learning is to improve the performance of the classifier by leveraging the strengths of multiple models [24]. One example of the usefulness of ensemble learning in the context of IDSs is the ability to combine the predictions of multiple classifiers trained on different subsets of an IDS dataset. This can lead to more accurate classification performance compared to using a single classifier. The diverse range of attack types and network traffic attributes present in IDS datasets can also pose a challenge for ML algorithms, as they increase the complexity of the problem and require significant computational and time resources to process. Ensemble learning can help mitigate these challenges by aggregating the predictions of multiple classifiers [25].

The following points summarize the advantages of using an ML-based IDS over a traditional signature-based IDS:

- A machine learning-based IDS (ML-IDS) can detect variants of attacks more effectively than a signature-based IDS because it can learn the normal behaviour of traffic flow and recognize deviations from it.
- An ML-IDS does not place as much strain on computer processors as a signature-based IDS because it does not need to analyse all the signatures in a signature database.
- Some ML-IDSs, particularly those using unsupervised learning algorithms, can detect new, previously unseen attacks.
- An ML-IDS can more accurately and quickly identify attacks by considering complex characteristics of attack behaviour, compared to a signature-based IDS.
- A signature-based IDS needs continually to maintain and update its signature database to stay current, whereas an ML-IDS based on clustering and outlier detection does not require such updates.

The contribution of this paper consists of proposing an ML-NIDS based on optimizing an SVM classifier in order to maximize the accuracy and minimize the number of rounds for the confusion matrix to converge and correctly classify the various attacks. The TON_IoT dataset is also investigated to classify common attacks in an IoT environment, such as injection, man-in-the-middle (MiTM), denial of service (DoS), distributed denial of service (DDoS), password, scanning, and cross-site scripting (XSS) attacks. The next section presents the experiment and a discussion of the results.

3 Dataset and Common Attacks on the IoT

One of the most common security threats to IoT devices is DoS attacks. This type of attack involves overloading a device with excessive traffic or demand, disrupting its normal operation and causing data loss, service interruption, and reputational damage. A more serious version of a DoS attack is a DDoS attack, in which multiple entities coordinate and launch an attack on a single target, increasing the scope and impact of that attack [26]. Another security threat facing IoT devices is injection attacks. These attacks involve injecting malicious code or data into devices, resulting in data modification. Injection attacks can cause significant harm, including theft of sensitive information, data corruption, and service disruption of the IoT system [27]. The attacker can then gain full control of an IoT system [28]. XSS attacks, which are also becoming more common in IoT environments, involve injecting malicious code into a website that can be executed when unsuspecting users visit the website, potentially compromising their data [29].

To address these security threats, several datasets have been developed to evaluate and design IDSs for the IoT. The KDD'99 is a dataset that is widely used for evaluating IDSs and consists of network traffic data labelled as normal and attack cases [30]. The NSL-KDD is a version of the KDD'99 dataset that was developed by redundancy suppression and labelling all nominal values [31]. Finally, there is the TON_IoT dataset, which aims to aggregate and analyse various IoT and Industrial IoT (IIoT) data sources and contains heterogeneous data collected from various sources, including telemetry data from connected devices, Windows and Linux system logs, and system network traffic [32]. The TON_IoT dataset is based on seven attack classes: DoS, DDoS, injection, MiTM, password, XSS, and scanning attacks.

4 Proposed Model

This section shows the steps taken to collect, prepare, and utilize a dataset for training and testing to maximize the model accuracy and to minimize the number of rounds for the confusion matrix to classify all attacks correctly. Fig. 1 provides a visual representation of the steps needed to execute the various ML metrics. The evaluation of the ML involves a six-step life cycle that encompasses the selection of the dataset, pre-processing, choosing a classifier model, training the model, testing, and applying ML evaluation metrics.

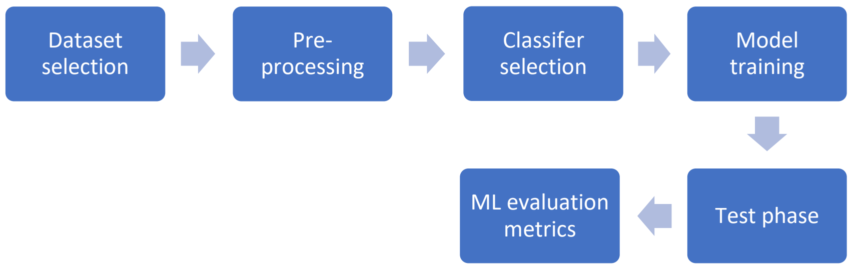

Fig. 1: Life cycle of the ML metrics evaluation

The TON_IoT dataset was selected to perform the experiment simulation. The pre-processing phase consists of selecting the pertinent attributes to maximize the model accuracy. We selected a binary sequential minimal optimization (SMO) classifier, which is an SVM that uses John Platt’s SMO algorithm. During the model training, the SVM classifier defines the hyperplane to separate the different classes with an optimization process to maximize the distance between the hyperplane and the closest points in the dataset. For the testing phase, 10-fold cross-validation was selected during the simulation to obtain the evaluation metrics based on the SVM classifier. Finally, the ML evaluation metrics were conducted on classifier performance metrics, such as the FPR, TPR, precision, F-measure, recall, MCC, ROC area, PRC area, a confusion matrix, and classifier errors.

5 Overall Methodology

This paper proposes an optimized approach to cyber-attack classification in an IoT environment based on an SVM classifier. We investigate the implementation of an ML model by using the TON_IoT dataset as input and accuracy and a confusion matrix as the output. The aim of this work is to maximize the model accuracy and minimize the number of rounds for confusion matrix convergence to classify all attacks correctly by manipulating the SVM complexity parameters (C parameters). This experiment utilized accuracy, sensitivity, precision, and MCC as performance metrics, all of which were executed within a WEKA data mining environment.

WEKA is a software suite, which is open source and freely available, and is widely utilized by researchers in the field of ML for implementing a variety of ML algorithms, including regression, classification, and clustering. WEKA 3.8 was used to execute the experiments presented in this paper.

The optimization technique is utilized through the use of the SMO classifier, known in the literature as a kind of SVM classifier, by executing and training a support vector classifier known as John Platt’s SMO algorithm. In addition, we focus on the SVM C parameters and look for the optimal value leading to the maximum accuracy. We also take the result given by the authors in [33] as a reference by taking a default SMO classifier complexity parameter equal to 1 with a polynomial kernel.

6 Experiment and Discussion of the Results

Table 2 shows a comparison of SMO metrics for various C parameter values.

Table 2: SVM metric comparison for various C parameters

Metric	C=1	C=2	C=3	C=4
Correctly classified instance	35960	35974	35975	35975
Incorrectly classified instance	15	1	0	0
Accuracy	99.9583	99.9972	100	100

As shown in Table 2, by increasing the classifier complexity parameter from 1 to 4, the correctly classified instances move from 35960 to 35975, and the incorrectly classified instances decrease from 15 to 0. Moreover, the SVM accuracy has increased from 99.9583% to 100%. It is known that the C parameter range is from 1 to infinity and is responsible for maximizing the distance separating the hyperplane and the nearest points that belong to various classes during the training process. By default, the C parameter is set to 1 and the hyperplane is determined during the training process, and the classifier succeeded in classifying 35960 instances correctly during the test phase leading to a model accuracy of 99.9583%. Then, when the C parameter is increased to 2, the classifier raises the distance between the hyperplane and the 15 nearest points during the training process and succeeded in classifying 14 new instances correctly (see Table 2, row 2: incorrectly classified instances), leading to a model accuracy of 99.9972%. Finally, by raising the C parameter to 3, the remaining incorrectly classified instances are correctly classified by increasing the distance between the hyperplane and the remaining instances, leading to an accuracy value of 100%. As a conclusion, the SVM classifier reached maximum accuracy from an SVM C parameter equal to 3.

The SVM performance metrics conducted in this study are as follows:

- Precision, defined as:

$$Precision = \frac{\text{TP}}{(\text{TP + FP})} \times 100 \tag{1}$$

- Recall (also known as sensitivity) is the proportion of actual positives which are predicted to be positive:

$$Sensitivity = \frac{\text{TP}}{(\text{TP + FN})} \times 100 \tag{2}$$

- F-measure, defined as the proportion of actual positives which are predicted to be positive:

$$F - measure = \frac{\text{TP}}{(\text{TP + FN})} \times 100 \tag{3}$$

- MCC defines the correlation between the predicted value and the actual value:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt[2]{(TP+TN)\times(TP+FN)\times(TN+FP)\times(FP+FN)}} \times 100 \tag{4}$$

- ROC area is defined as a probability curve plotting TP versus FP at different threshold values.
- PRC area represents precision versus recall.

A confusion matrix is also used as a classifier performance metric to show the predicted class with respect to the actual class for the various attack classes, such as normal, DoS, DDoS, MiTM, scanning, and XSS attacks. Finally, a comparison of SMO classifier errors is performed as a function of various kernel types.

6.1 Detailed accuracy by class for C=1

Table 3 shows the SVM performance evaluation metrics as a function of the FPR, TPR, precision, F-measure, recall, ROC area, PRC area, and MCC according to attack class for an SVM classifier C parameter equal to 1.

Table 3: SVM metrics by class for C=1

Class	FP Rate	TP Rate	Precision	F-measure	Recall	ROC Area	PRC Area	MCC
Normal	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DDoS	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DoS	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Injection	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MiTM	0	0	-	-	1.0	0.996	0.053	-
Password	0	1.0	0.996	0.998	1.0	1.0	0.996	-
XSS	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Scanning	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	0	1.0	-	-	1.0	1.0	0.999	-

6.2 Confusion matrix for C=1

Table 4 presents the confusion matrix for an SVM classifier C parameter equal to 1 as a function of various attack classes.

Table 4: Confusion matrix for SVM C=1

Normal	DDoS	DoS	Injection	MiTM	Password	XSS	Scanning
24871	0	0	0	0	0	0	0
0	4608	0	0	0	0	0	0
0	0	525	0	0	0	0	0
0	0	0	612	0	0	0	0
0	0	0	0	0	15	0	0
0	0	0	0	0	3628	0	0
0	0	0	0	0	0	1269	0
0	0	0	0	0	0	0	447

According to Table 4, we can see that a password attack has TPR and FPR values of 3628 and 15, respectively. This is due to the precision value of 0.996 for the password class given in Table 3. In addition, the SVM classifier classified 15 instances of an MiTM attack as a password attack with an error rate of 0.004.

6.3 Detailed accuracy by class for SVM C=2

Table 5 shows the SVM performance evaluation metrics as a function of the FPR, TPR, precision, F-measure, recall, ROC area, PRC area, and MCC according to attack class for an SVM classifier C parameter equal to 2.

Table 5: SVM metrics by class for C=2

Class	FP Rate	TP Rate	Precision	F-measure	Recall	ROC Area	PRC Area	MCC
Normal	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DDoS	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Dos	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Injection	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MiTM	0	0.933	1.0	0.966	0.933	0.999	0.935	0.966
Password	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
XSS	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Scanning	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 5 above illustrates that the lowest values for recall, F-measure, MCC, ROC area, and PRC area are for MiTM attacks, with values of 0.933, 0.966, 0.966, 0.999, and 0.935, respectively. In addition, the precision of password attack changes from the 0.996 given in Table 3 to 1.

6.4 Confusion matrix for C=2

Table 6 presents the confusion matrix for an SVM classifier C parameter equal to 2 as a function of various attack classes.

Table 6: Confusion matrix for SVM C=2

Normal	DDoS	DoS	Injection	MiTM	Password	XSS	Scanning
24871	0	0	0	0	0	0	0
0	4608	0	0	0	0	0	0
0	0	525	0	0	0	0	0
0	0	0	612	0	0	0	0
0	0	0	0	14	1	0	0
0	0	0	0	0	3628	0	0
0	0	0	0	0	0	1269	0
0	0	0	0	0	0	0	447

Table 6 shows that the FPR for password attacks changed from 15 to 1. Furthermore, in this iteration, the TPR for MiTM attacks changed from 0 to 14.

6.5 Detailed accuracy by class for C=3

Table 7 shows the SVM performance evaluation metrics as a function of FPR, TPR, precision, F-measure, recall, ROC area, PRC area, and MCC according to attack class for an SVM classifier C parameter equal to 3.

Table 7: SVM metrics by class for C=3

Class	FP Rate	TP Rate	Precision	F-measure	Recall	ROC Area	PRC Area	MCC
Normal	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DDoS	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DoS	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Injection	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MiTM	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Password	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
XSS	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Scanning	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 7 shows the maximum value of TPR, precision, recall, F-measure, MCC, ROC area, and PRC area for all attack classes. Likewise, according to Table 7, the FPR is the minimum for all attack classes.

6.6 Confusion matrix for C=3

Table 8 presents the confusion matrix for an SVM classifier C parameter equal to 3 versus various attack classes.

Table 8: Confusion matrix for SVM C=3

Normal	DDoS	DoS	Injection	MiTM	Password	XSS	Scanning
24871	0	0	0	0	0	0	0
0	4608	0	0	0	0	0	0
0	0	525	0	0	0	0	0
0	0	0	612	0	0	0	0
0	0	0	0	15	0	0	0
0	0	0	0	0	3628	0	0
0	0	0	0	0	0	1269	0
0	0	0	0	0	0	0	447

Table 8 shows that the FPR for password attacks changed from 1 to 0. Furthermore, in this iteration, the TPR for MiTM attacks changed from 14 to 15.

As shown in Table 8, the SVM classifier is classifying all classes correctly, and especially MiTM class attacks, as a result of raising the SVM classifier C parameter to 3. Furthermore, we obtain the same result by increasing the SVM classifier complexity beyond 3 and the SVM classifier converges to the optimized and maximized classifier evaluation metrics (i.e., precision, TPR, recall, F-measure, MCC, ROC area, and PRC area).

Table 9 shows a comparison of various kernel SMO types (radial basis function [RBF], normalized polynomial, and polynomial) as a function of time to build the model, and correctly classified instances, incorrectly classified instances, kappa statistics, and root mean error.

Table 9: SMO metric comparison for various kernel functions

Kernel	RBF	Normalized Polynomial	Polynomial
Time to build the model (s)	219.66	73.07	2.54
Correctly classified instance	99.72	99.85	100
Incorrectly classified instance	0.28	0.15	0
Kappa statistic	0.99	0.99	1
Root mean squared error	0.29	0.29	0.29

As shown in Table 9, the minimum time to build the model (i.e., 2.54 s) is for the polynomial SMO kernel, followed by the normalized polynomial kernel (73.07 s), and then the RBF kernel (219.66 s). Further, the correctly classified instances increased from 99.72% to 99.85% for the RBF and normalized polynomial kernels and reaching 100% for the polynomial SMO kernel. In addition, the incorrectly classified instances decreased from 0.28% to 0.15% for the RBF and normalized polynomial kernels, reaching 0% for the polynomial SMO kernel. The kappa statistics increased from 0.99 for the RBF and normalized polynomial kernels to 1 for the polynomial kernel. Finally, the root mean squared error has the same value for the various SMO kernel types studied.

The following SMO classifier configuration setups are used to make a comparison between RBF, normalized polynomial, and polynomial kernels:

```
weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.RBFKernel -C 250007 -G 0.01" -calibrator "weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4"
```

```
weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.NormalizedPolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4"
```

```
weka.classifiers.functions.SMO -C 3.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4"
```

Next, a comparison in terms of SMO classifier errors is performed as a function of the predicted attack classes and the types of target attacks for the RBF kernel (Fig. 2), normalized PolyKernel (Fig. 3), and polynomial kernel (Fig. 4).

According to Fig. 2, the SMO classifier with an RBF kernel makes errors in classifying DDoS, DoS, password, and scanning attacks. Further, the classifier predicted DDoS attacks as DDoS and DoS attacks. It predicted a DoS attack as a DoS and scanning attack. The classifier also predicted a password attack as password, DoS, MiTM, XSS, and scanning attacks. Finally, the RBF classifier predicted scanning attacks as scanning and DoS attacks.

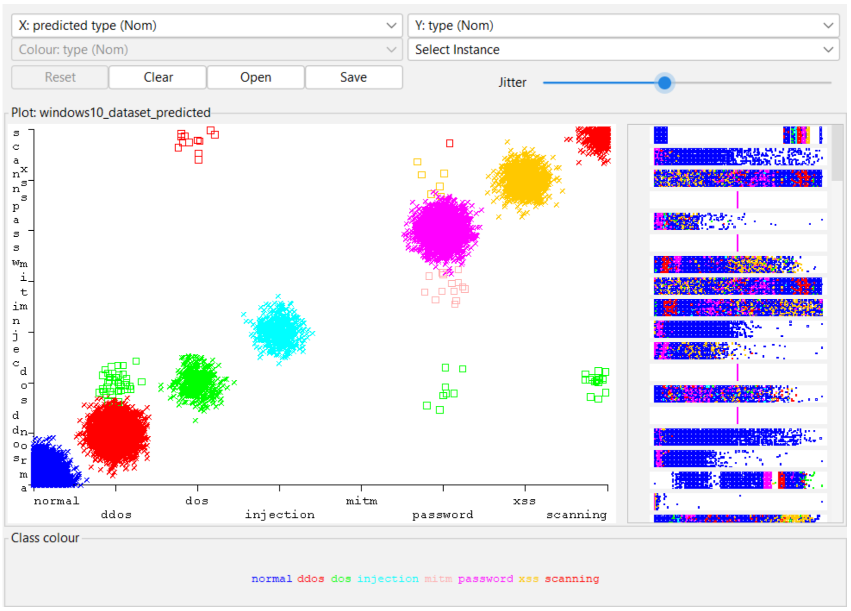
X: predicted type (Nom)
Y: type (Nom)
Colour: type (Nom)
Plot: windows10_dataset_predicted

Fig. 2: SMO classifier errors with the RFB kernel

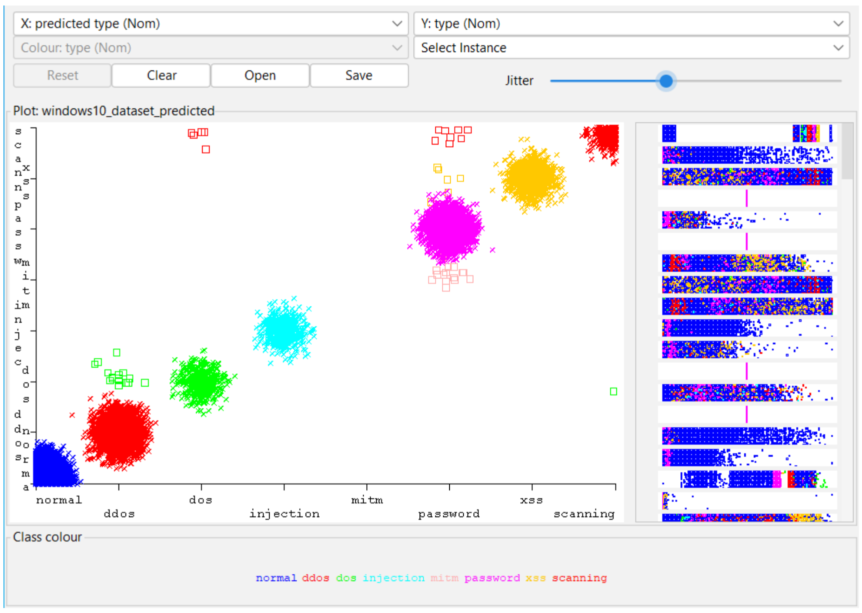
X: predicted type (Nom)
Y: type (Nom)
Colour: type (Nom)
Plot: windows10_dataset_predicted

Fig. 3: SMO classifier errors with the normalized PolyKernel

According to Fig. 3, the SMO classifier with the normalized polynomial kernel made errors in classifying DDoS, DoS, password, and scanning attacks. The normalized polynomial kernel-based classifier predicted DDoS attacks as DDoS and DoS attacks and predicted DoS attacks as DoS and scanning attacks. The classifier also predicted password attacks as password, MiTM, XSS, and scanning attacks. Finally, the SMO classifier based on a normalized polynomial kernel predicted scanning attacks as scanning and DoS attacks.

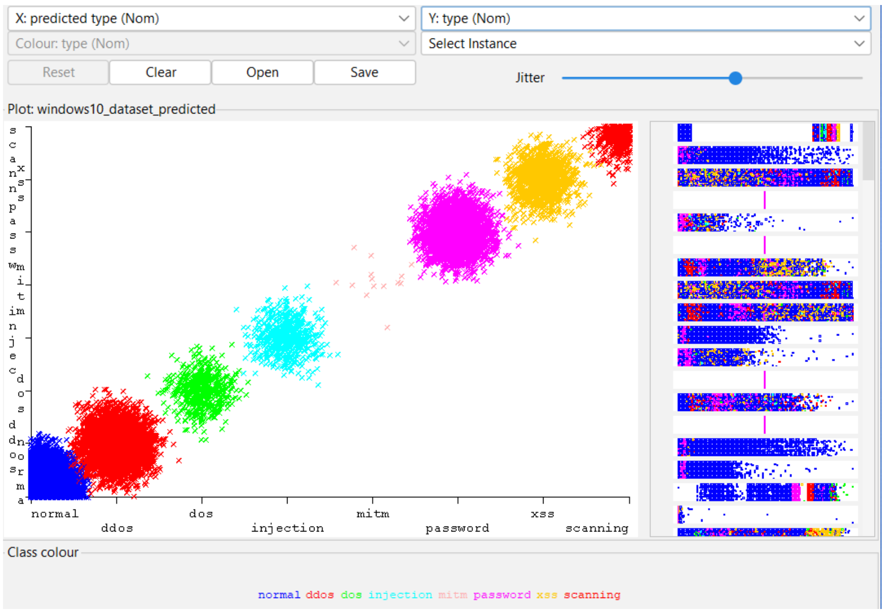

Fig. 4: SMO classifier errors with PolyKernel

According to Fig. 4, the optimized SMO classifier with a polynomial kernel made no errors in classifying normal activity or DDoS, DoS, injection, MiTM, password, XSS, and scanning attacks.

In conclusion, the SMO classifier based on a polynomial kernel outperformed the RBF and normalized polynomial kernels in terms of time to build the model, correctly classified instances, incorrectly classified instances, and kappa statistics. In contrast, the optimized SMO classifier based on the polynomial kernel is the best classifier in terms of classifier errors compared to the RBF and normalized polynomial kernels. Finally, the ML-IDS based on the optimized parameter SMO classifier (C parameter equal to 3 and a polynomial kernel) could be deployed in a future IDS solution to predict common attacks in an IoT environment.

7 Conclusion

In this work, we presented an ML-NIDS based on an optimized SVM classifier. By manipulating the SMO complexity parameter as a function of classifier performance metrics (i.e., TPR, FPR, precision, recall, F-measure, MCC, ROC area, PRC area, and a confusion matrix), an experiment study was conducted based on the TON_IoT dataset. This work has shown that the ML-NIDS based on an SVM classifier reached a maximum accuracy of 100% for a complexity parameter starting from 3. The confusion matrix also converges after only three rounds to classify all attacks in the IoT network correctly. Moreover, our research reveals that an SVM based on a polynomial kernel outperformed a classifier based on RBF and normalized kernels in terms of classifier errors. As a perspective of this work, the utilization of optimization techniques, such as genetic algorithms, ant colony optimization, heuristics, and particle swarm optimization, can be applied to enhance and improve ML layout analysis. The incorporation of optimization techniques may serve as an approach to improving ML-NIDS tool performance metrics. The performance evaluation of deep learning-based classifiers is highly dependent on the number of neurons present within the network, as well as the weight and bias values assigned to each neuron. These factors play a critical role in determining the overall accuracy and effectiveness of the classifier.

References

[1] Oueslati, N. E., Mrabet, H., Jemai, A., & Alhomoud, A. (2019, December). Comparative study of the common cyber-physical attacks in industry 4.0. In *2019 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)* (pp. 1-7). IEEE.

[2] Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C., & Faruki, P. (2019). Network intrusion detection for IoT security based on learning techniques. *IEEE Communications Surveys & Tutorials, 21*(3), 2671-2701.

[3] Jamalipour, A., & Murali, S. (2021). A Taxonomy of Machine-Learning-Based Intrusion Detection Systems for the Internet of Things: A Survey. *IEEE Internet of Things Journal, 9*(12), 9444-9466.

[4] Gyamfi, E., & Jurcut, A. (2022). Intrusion Detection in Internet of Things Systems: A Review on Design Approaches Leveraging Multi-Access Edge Computing, Machine Learning, and Datasets. *Sensors, 22*(10), 3744.

[5] Moustafa, N. (2021). A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustainable Cities and Society, 72,* 102994.

[6] Puzis, R., Klippel, M. D., Elovici, Y., & Dolev, S. (2008). Optimization of NIDS placement for protection of intercommunicating critical infrastructures. In *Intelligence and Security Informatics: First European Conference, EuroISI 2008, Esbjerg, Denmark, December 3-5, 2008. Proceedings* (pp. 191-203). Springer Berlin Heidelberg.

[7] Torkaman, A., Javadzadeh, G., & Bahrololum, M. (2013, May). A hybrid intelligent HIDS model using two-layer genetic algorithm and neural network. In *The 5th Conference on Information and Knowledge Technology* (pp. 92-96). IEEE.

[8] Fuchsberger, A. (2005). Intrusion detection systems and intrusion prevention systems. *Information Security Technical Report, 10*(3), 134-139.

[9] Elhag, S., Fernández, A., Bawakid, A., Alshomrani, S., & Herrera, F. (2015). On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems. *Expert Systems with Applications, 42*(1), 193-202.

[10] Wang, K., Du, M., Sun, Y., Vinel, A., & Zhang, Y. (2016). Attack detection and distributed forensics in machine-to-machine networks. *IEEE Network, 30*(6), 49-55.

[11] Wang, K., Du, M., Yang, D., Zhu, C., Shen, J., & Zhang, Y. (2016). Game-theory-based active defense for intrusion detection in cyber-physical embedded systems. *ACM Transactions on Embedded Computing Systems 16*(1), 1-21

[12] Joldzic, O., Z. Djuric, and P. Vuletic, A transparent and scalable anomaly-based DoS detection method. Computer Networks, 2016. 104: pp. 27-42.

[13] Papamartzivanos, D., Mármol, F. G., & Kambourakis, G. (2018). Dendron: Genetic trees driven rule induction for network intrusion detection systems. *Future Generation Computer Systems, 79,* 558-574.

[14] Kim, J., Kim, J., Thu, H. L. T., & Kim, H. (2016, February). Long short term memory recurrent neural network classifier for intrusion detection. In *2016 international conference on platform technology and service (PlatCon)* (pp. 1-5). IEEE.

[15] Pathan, A. S. K. (Ed.). (2014). *The state of the art in intrusion prevention and detection* (Vol. 44). Boca raton: CRC press.

[16] Hecht-Nielsen, R. (1992). Neural networks for perception. *Theory of the backpropagation neural network* (pp. 65-93). Academic Press.

[17] Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

[18] Cortes, C., & Vapnik, V. (1995). Support vector machine. *Machine learning, 20*(3), 273-297.

[19] Du, M., Wang, K., Chen, Y., Wang, X., & Sun, Y. (2018). Big data privacy preserving in multi-access edge computing for heterogeneous Internet of Things. *IEEE Communications Magazine, 56*(8), 62-67.

[20] Du, M., Wang, K., Xia, Z., & Zhang, Y. (2018). Differential privacy preserving of training model in wireless big data with edge computing. *IEEE transactions on big data, 6*(2), 283-295.

[21] Mishra, P., Varadharajan, V., Tupakula, U., & Pilli, E. S. (2018). A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE communications surveys & tutorials, 21*(1), 686-728.

[22] Feng, X., Xiao, Z., Zhong, B., Qiu, J., & Dong, Y. (2018). Dynamic ensemble classification for credit scoring using soft probability. *Applied Soft Computing, 65,* 139-151.

[23] Salo, F., A.B. Nassif, and A. Essex, Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection. Computer Networks, 2019. 148: pp. 164-175.

[24] Pham, N. T., Foo, E., Suriadi, S., Jeffrey, H., & Lahza, H. F. M. (2018, January). Improving performance of intrusion detection system using ensemble methods and feature selection. In *Proceedings of the Australasian computer science week multiconference* (pp. 1-6).ACM.

[25] Aljawarneh, S., Aldwairi, M., & Yassein, M. B. (2018). Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science, 25,* 152-160.

[26] Salim, M. M., Rathore, S., & Park, J. H. (2020). Distributed denial of service attacks and its defenses in IoT: a survey. *The Journal of Supercomputing, 76,* 5320-5363.

[27] Muhammad, F., Anjum, W., & Mazhar, K. S. (2015). A critical analysis on the security concerns of internet of things (IoT). *International Journal of Computer Applications, 111*(7), 1-6.

[28] Deogirikar, J., & Vidhate, A. (2017, February). Security attacks in *IoT: A survey. In 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)* (pp. 32-37). IEEE.

[29] Rodríguez, G. E., Torres, J. G., Flores, P., & Benavides, D. E. (2020). Cross-site scripting (XSS) attacks and mitigation: A survey. *Computer Networks*, *166*, 106960.

[30] Tareq, I., Elbagoury, B. M., El-Regaily, S., & El-Horbaty, E. S. M. (2022). Analysis of ToN-IoT, UNW-NB15, and Edge-IIoT Datasets Using DL in Cybersecurity for IoT. *Applied Sciences*, *12*(19), 9572.

[31] Meena, G., & Choudhary, R. R. (2017, July). A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA. In *2017 International Conference on Computer, Communications and Electronics (Comptelix*) (pp. 553-558). IEEE.

[32] Dhanabal, L., & Shantharajah, S. P. (2015). A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *International journal of advanced research in computer and communication engineering*, *4*(6), 446-452.

[33] Mrabet, H., Alhomoud, A., Jemai, A., & Trentesaux, D. (2022). A Secured Industrial Internet-of-Things Architecture Based on Blockchain Technology and Machine Learning for Sensor Access Control Systems in Smart Manufacturing. *Applied Sciences*, *12*(9), 4641.

Notes on contributor

Dr Adeeb Alhomoud holds a PhD in cyber security from the University of Bradford (2014) and a Master's degree from the University of Essex (2005) in the United Kingdom. Alhomoud is a technically astute cyber security specialist with 15+ years of diversified experience across the academic and IT sector. He has had significant exposure to the fields of cyber security, network security, malware attacks, botnets, and self-healing and offensive security. He is a Certified GRC Professional, Certified Ethical Hacker, ISO 27001 LI, with extensive knowledge of security threats, risk analysis and the development of security systems and protocols in business and corporate settings. His research interests include botnets, malware propagation, the IoT, steganography, machine learning, and artificial intelligence.