Neural Computing and Applications (2024) 36:3239–3257
https://doi.org/10.1007/s00521-023-09309-y

ORIGINAL ARTICLE

Optimizing anomaly-based attack detection using classification machine learning

Hany Abdelghany Gouda[1] · Mohamed Abdelslam Ahmed[1] · Mohamed Ismail Roushdy[2]

Received: 10 April 2023 / Accepted: 16 November 2023 / Published online: 20 December 2023

Abstract

One of the significant aspects of our digital world is that data are literally everywhere, and it is increasing. On the other hand, the number of cyberattacks aiming to seize this data and use it illegally is increasing at an exponential rate, and this is the challenge. Therefore, intrusion detection systems (IDS) have attracted considerable interest from researchers and industries. In this regard, machine learning (ML) techniques are playing a pivotal role as they put the responsibility of analyzing enormous amounts of data, finding patterns, classifying intrusions, and solving issues on computers instead of humans. This paper implements two separate classification layers of ML-based algorithms with the recently published NF-UQ-NIDS-v2 dataset, preprocessing two volumes of sample records (100 k and 10 million), utilizing MinMaxScaler, LabelEncoder, selecting superlative features by recursive feature elimination, normalizing the data, and optimizing hyper-parameters for classical algorithms and neural networks. With a small dataset volume, the results of the classical algorithms layer show high detection accuracy rates for support vector (98.26%), decision tree (98.78%), random forest (99.07%), K-nearest neighbors (98.16%), CatBoost (99.04%), and gradient boosting (98.80%). In addition, the layer of neural network algorithms has proven to be a very powerful technology when using deep learning, particularly due to its unique ability to effectively handle enormous amounts of data and detect hidden correlations and patterns; it showed high detection results, which were (98.87%) for long short-term memory and (98.56%) for convolutional neural networks.

Keywords Intrusion detection · Detection techniques and methodologies · Classical Machine learning algorithms · Neural network and dataset

1 Introduction

Globalization and constant change are currently the main drivers for information system security, resulting in a significant increase in security risks such as new attacks being detected daily, which can be disastrous [1, 2]. Most human behavior patterns that involve theft, misuse, deceit, and the making of mistaken suggestions are classified as intrusion and fraud [3]. According to a report by Cisco on annual internet usage, digital transformation and technological advancements have increased the frequency of cyberattacks and the total cost of data breaches. Furthermore, the nature of the threats is constantly changing [4]. Intrusion detection constitutes one of the most important methodologies for dealing with information security incidents; it is based on selections, deployments, and operations that cover the entire spectrum of mechanisms for signs of potential attacks [5–7]. However, not all vulnerabilities have always been considered equal, and thus constant improvements in detection must be made, according to the Edgescan Vulnerability Stats Report for 2022 [8].

Utilizing machine learning (ML) is one of the major research areas being studied currently to improve intrusion detection (ID). These ML approaches aim to overcome the

✉ Hany Abdelghany Gouda
hany.abdelghany21@commerce.helwan.edu.eg

Mohamed Abdelslam Ahmed
dr.m_abdelsalam@commerce.helwan.edu.eg

Mohamed Ismail Roushdy
mohamed.roushdy@fue.edu.eg

[1] Information Systems Department, Faculty of Commerce and Business Administration, Helwan University, Cairo, Egypt

[2] Computer Science Department, Faculty of Computers and Information Technology, Future University in Egypt, Cairo, Egypt

limitations of traditional technologies and humans in addressing the massive variations and complexity of computer system events and network traffic. Many research papers have been dedicated to this context, proposing improvement models to combat intrusion for both network-based and host-based systems to protect the information system environment. As a result, a broad range of security systems have emerged [9]. However, researchers have concluded that intrusion detection systems (IDSs) using machine learning still need more research. Since no single algorithm can solve all problems, the type of problem and researcher objectives are the deciding factors in algorithm selection [10].

The majority of the IDS-related literature studies that have been published in the past three years (2019–2022) have trained machine learning algorithms on a small sample size from aged datasets like NSL–KDD, UNSW–NB15, CICIDS–2017, and ADFA–LD, which does not provide a reliable indicator of how effective those algorithms are. It is significant to note in this context that the algorithm training method on these datasets produces three essential real-world limitations:

- The first of which is on data processing like redundancy, cleansing, feature selection, and scaling.
- The second is related to anomaly traffic like diverse, sophisticated, lifetime, mutation, and generation of an old attack pattern into a new attack pattern.
- Finally, the third is related to algorithm hyper-parameter influences like under-fitting and over-fitting.

Consequently, these constraints make it impractical to deploy these models in real-world systems, which are exposed to a wide range of novelty-generated attacks, and achieving high accuracy is uncertain at all. This is a research gap that must be filled.

Hence, the importance of the research and its motives are as follows:

- The initial and most important motivation for the study is a deep interest in this domain.
- The importance of the study derives from the significant nature of the field itself.
- More studies are needed to continuously develop models to detect incursions using machine learning.
- We contributed an effective method for enhancing the capabilities of classification algorithms to detect hidden correlations and patterns.
- Presenting the results of this study with the aim of enriching scientific research in this domain.

2 Related works

Meftah et al. [11] used the UNSW-NB15 dataset, which consists of 164,673 anomaly traffic records divided into nine categories: Analysis, Backdoor, DOS, Exploits, Fuzzing, Generic, Reconnaissance, Shellcode, and Worms, to implement a two-phase anomaly-based NID algorithm. In the data preparation process for machine learning, the authors utilized feature selection based on recursive feature elimination (RFE) and random forest (RF) with a tenfold cross-validation attribute for dimensionality reduction. In the first phase, they used logistic regression (Log. Reg.) as the first algorithm to train the dataset with some functions and attributes such as R function glm for a fit model, the ANOVA filter method for selection, the Chi-Square for correlation between two variables, and CV.GLM for reliability. They then used the gradient boosting machine as the second algorithm to evaluate complexity and effects and support vector machine as the third algorithm, which was considered a constraint. In the second phase, the best result from the first phase was returned as input to three different algorithms: decision trees (DT C5.0), support vector machines (SVM), and Naïve Bayes (NB). However, the authors found weak points in the initial data preparation process, such as improper choices in cleansing, feature selection and extraction, and scaling. In addition, they failed to train and detect four categories of attacks out of the nine. The highest accuracy achieved was 86%.

Alamiedy et al. [12] estimated that approximately 20% of the NSL-KDD dataset was used, involving 36,772 records divided into a training set and a testing set, and four main classes of attacks. The data preparation occurred in the initial phase through the transformation mapping process of four symbolic features to numeric ones. Then, the feature range was scaled, and the output was filtered to different main classes, each with the same attack properties. In the next phase, the authors proposed the gray wolf optimization (GWO) algorithm to improve the optimal subset of feature selection by reducing the number of features via a multi-objective function. In the last phase, a support vector machine (SVM) was utilized to accurately predict intrusions. Finally, the highest accuracy achieved was 93.64% for DoS and 91.01% for Probe. However, weak points were found within a small dataset volume, and the detection accuracy was low: 57.72% for R2L and 53.7% for U2R.

Devan and Khare [13] worked on the NSL–KDD dataset, which involved four main classes of anomaly-based intrusion and 41 features as input data. In the data preparation stage, the authors converted the values attribute of all features to numeric, then normalized the data using the preprocessing-MinMaxScaler with feature_range attribute

0,1 to avoid bias. Then, they used the XGBoost model to eliminate the lowest 20 features in the selection process with an accuracy score of 98.91%. In the classification stage, the authors used the deep neural network (DNN) algorithm and optimized it using Adam optimization to reduce issues related to mean and variance parameters. Finally, the authors achieved the highest accuracy of 97%.

Wei et al. [14] conducted research on two datasets, namely the UNSW-NB15 and NSL-KDD datasets. The UNSW-NB15 dataset consisted of 119,341 anomaly records related to nine classes of attacks, whereas the NSL-KDD dataset included 58,630 anomaly records indicating four types of attacks. The researchers aimed to address the challenge of feature selection in the data preparation stage, which has a direct impact on training and classification processes. To solve this issue, they proposed a multi-objective immune algorithm (MOIA). In the classification stage, a neural network algorithm was used to train and classify the dataset. The experimental results showed a detection accuracy of 79.81% for the first dataset with about 16 features, which excluded some common attacks. On the other hand, the second dataset achieved a higher detection accuracy of 99.47% with a total of 24 features.

Zhang et al. [15] conducted research on proactive intrusion detection on a server machine using the ADFA-LD dataset, which is based on the Linux core. The dataset was divided into more than two-thirds for training and the rest for testing, containing two types of data (normal and attacks). The authors analyzed and preprocessed the system call trace and trace length using N-gram and evaluated the significance sequence through term frequency and inverse document frequency (TF-IDF). Six classification models were used, and the best model with an accuracy of 97.05% was the multilayer perceptron (MLP) model with a k-fold size of 10.

Masser et al. [16] conducted a comparative study based on three key performance indicators (KPIs). First, an anomaly-based detection methodology was used. Second, the CICIDS2017 dataset, which includes two types of traffic (normal and attacks) with 14 sub-anomalies and divided into three different ratios (4–6, 5–5, and 6–4). Third, seven supervised algorithms were utilized, including artificial neural network solver = ADAM, decision tree using entropy, max depth = none, and weight = balanced, *k*-nearest neighbor with $k = 1$, Naive Bayes with default values, random forest using estimators = 100, weight = balanced, and max depth = none, and support vector machine with kernel = RBF, max iter = -1, and weight = balanced. In addition, three unsupervised algorithms were used, including convolutional neural network with n-estimators = 100, *k*-means with clusters = 4, max_iter = 300, and expectation–maximization, and self-organizing maps, both using the same model by default. The study showed that KNN, DT, and NB algorithms performed best in detecting attacks. However, they had limitations in recognizing unknown intrusions, and it was recommended to use multiple algorithms against multiple attacks.

Anwer et al. [17] recommend an anomaly-based NID framework. In the third section, they identified gaps in the previous literature, such as inefficiencies in fog-based attack detection, feature selection, inaccurate algorithms on small datasets, and rating of false positives and negatives. They used the NS-KDD dataset, which was split into 80% for training and 20% for testing, with a scaling ratio of approximately 78:75. Three types of classification algorithms were utilized, including SVM with an accuracy of under 35%, GBDT with an accuracy of 78.01%, and RF, which was the most accurate with an accuracy of 85.34%.

Al-Bakaa and Al-Musawi [18], in this work, the authors applied two different feature selection methods to the UNSW-NB15 dataset. The dataset had 49 features, and in the preprocessing stage, 35 features were selected by converting all values to numeric, scaling the features to a range of (0, 1) to avoid bias, and removing 14 noisy and insignificant features. Two types of classification algorithms, decision tree (DT) and random forest (RF), were used to classify the data. The results achieved 99.9% accuracy, but the weakness of the IDS efficiency in detecting multiple classes was also noted.

Singh and Kesswani [19] conducted a study on anomaly process detection across the IoT domain using architectural feature selection. They used the NSL-KDD dataset, which includes four types of attacks, and divided it into 4 million records for training and validation and 1 million records for testing. They applied scaling and reduction to eliminate irrelevant features and mitigate bias. They used the correlation coefficient to identify the factors influencing weight, and only included numerical values for low variance features. The feature_range attribute was set to (0, 1) to decrease data dimensionality. The k-fold attribute was set to 10 for the validation phase. However, the authors did not mention the classification algorithms used. The proposed method achieved a detection accuracy of 98.4%.

Ambavkar et al. [20] present a comprehensive analysis of seven different research topics conducted over six years that involve various classification algorithms, both individual and hybrid, that are relevant to intrusion detection systems. The analysis compares several factors such as algorithms used, datasets, accuracy, strengths, and weaknesses. The authors also propose several methods to identify the most effective features for enhancing the IDS efficiency. They prioritize detection accuracy and consider only results with an accuracy of 90% or higher, which are achieved by Bayesian-based algorithms, random forest

Table 1 Literature comparison

Nos	Authors	Dataset	Accuracy	Algorithms	Strength points	Weak points
1	This paper	NF-UQ-NIDS-v2 Year 2021 Classes 20	98.26, 98.78, 99.07, 98.16, 99.04, 98.80, 98.87, 98.56%	SVC, DT, RFC, KNN, CatBoost, Gradient Boosting, LSTM, CNN	Utilizing two separate classification layers ML-based with the recently published NF-UQ-NIDS-v2 dataset, preprocessing two volumes of various sizes (100k and 10 million), and hyper-parameter optimization	Four classes with classical algorithms. In addition, two types in neural networks
2	Meftah et al. [11]	UNSW-NB15 Year 2015 Classes 9	86%	Log. Reg, Gradient Boosting, SVM, D T, and NB	Use one phase to select superlative features and the other to perform a binary classification to detect anomalous traffic	A limited volume of data, failed to train four classes of attacks, choices in cleansing, feature selection, scaling, and failed to train and detect four categories of attacks
3	Alamiedy et al. [12]	NSL-KDD Year 1999 Classes 4	93.64, 91.01%	Gray wolf optimization and SVM	Detect two classes DoS and Probe	Aging dataset, failed to train two classes of attacks, Within a small dataset volume, 57.72% for R2L and 53.7% for U2R
4	Devan and Khare [13]	NSL-KDD Year 1999 Classes 4	98.91, 97%	XGBoost-DNN	XGBoost model for feature selection	A limited volume of data from an aging dataset. Inappropriate for comparability whenever employing classical algorithms to evaluate experimental results that are on a different deep layer
5	Wei et al. [14]	UNSW-NB15 Year 2015 Classes 9 NSL-KDD Year 1999 Classes 4	99.47%	NN, and multi-objective immune algorithm (MOIA)	improved multi-objective algorithm for optimize feature selection	The experimental conditions and attack methods through the aging NSL-KDD dataset and network traffic issue could not correctly represent the recent traffic. In addition, low accuracy achieved for the UNSW-NB15 dataset, which excluded some common attacks
6	Zhang et al. [15]	ADFA-LD Year 2017 Classes 6	97.05%	SVM, DT, RF, MLP, KNN and Multi-variable NB	Developing data preprocessing and system call analysis host-based	The loading process, extracting features, and prediction should all have low computation costs. With larger frames, it takes more time to extract the features
7	Masser et al. [16]	CICIDS2017 Year 2017 Classes 14	Highest KNN, DT, then NB	ANN, DT, KNN, NB, RF, SVM, and CNN	comprehensively reviews previous studies,, the KNN, DT, and NB models obtain the best results	limitations in recognizing unknown intrusions with ANN, RF, SVM, and CNN models
8	Anwer et al. [17]	NSL-KDD Year 1999 Classes 4	85.34%	SVM, GBDT, and RF	RF	Failed to train SVM with an accuracy of under 35% and GBDT with an accuracy of 78.01% due to an aging dataset. This attempt did not substantially fill the research gap
9	Al-Bakaa and Al-Musawi [18]	UNSW-NB15 Year 2015 Classes 9	99.96%	DT and RF	Forward selection ranking (FSR) and backward elimination ranking (BER) methods. Binary class	Detection delay issue and a multi-class scenario

Table 1 (continued)

Nos	Authors	Dataset	Accuracy	Algorithms	Strength points	Weak points
10	Singh and Kesswani [19]	NSL-KDD Year 1999 Classes 4	98.4%	Proposed model	The mechanism utilizes correlation-based filtration and feature-based trust factors	Aging dataset. the authors did not mention the classification algorithms used
11	Ambavkar et al. [20]	NSL-KDD Year 1999 Classes 4	99.48, 99.87, 99.22%	Genetic Algorithm (GA), Bagged Classifier, KNN, SVM, Breadth-Forest Tree (BFTree), NB, DT, RF, MLP, and LR	A comprehensive analysis, and They prioritize to NB, RF, SVM and DT	More time was required to build the model. Additionally, U2L and R2L attacks were not effectively detected

(RF), support vector machine (SVM), and decision tree (DT).

The significance of this paper is that it contributes improved models to intrusion detection utilizing both classical and neural algorithms trained on a recently released dataset. Table 1 provides a summary of the literature comparison that highlights the strengths and weaknesses.

3 Intrusion detection

Anderson [21] made one of the earliest references to intrusion detection techniques in the 1980s, concerning the potential abuse and early detection of harmful activities. Seven years later, D. Denning proposed the concept of a real-time intrusion detection expert system as the foundation for what is now known as an IDS [22].

IDSs are pieces of software that monitor and examine the information system environment for malicious activity and commonly used to detect and recognize attacks. Specifically, because of the constant and rapid improvements that are characteristic of attacks in this domain, it deemed important to prioritize innovative techniques and improving existing system. The intention behind this section is to present widely used IDS technologies, methodologies, capabilities, and any remaining limitations [5, 23–28].

3.1 Types of ID technologies

Network-based ID (NIDS) designed—placing sensors strategically across a network—to monitor cloud, on-premises, and hybrid architectures for abnormal events that could imply a threat. NIDS could very well recognize network intrusions in real-time by analyzing data from all network traffic gathered, both incoming and outgoing, rather than from individual hosts. NIDS, on the other hand, has various issues, the most serious of which is extensibility, which means it cannot accommodate and process encrypted communication traffic.

Host-based ID (HIDS) inspects the integrity of the host itself—placed to monitor assets—by examining several features running, such as system files, system calls, services, configuration files, ports, audit logs, and abnormal activities. HIDS technologies typically require elevated privileges on the host to track the state of system layer files and execute responsive actions. For this reason, it is important that the IDS itself be secured and protected against attacks, as compromises could lead to unauthorized administrator level access. HIDS can monitor every single process, executables, kernel entries, memory utilization, and a variety of other features to report indicators of attacks.

3.2 Detection methodologies

The detection methodology is a strategy for monitoring, analyzing, determining occurrences and potential threats, and reporting actions. It comprises a management console and sensors, and each technique has its own set of pros and cons, which are presented individually under signature-based and anomaly-based.

- Signature-based: this detection method uses a list known as indicators of compromise (IOCs). It works by creating sequences and patterns of fingerprints of well-known attacks and then using them to identify intrusions when those match a particular signature, sometimes known as ‘misuse detection.’ Simpler, more performant, and providing accurate consequences of explicit knowledge of the attack with low false positive alarms. The signature database faces failure against any

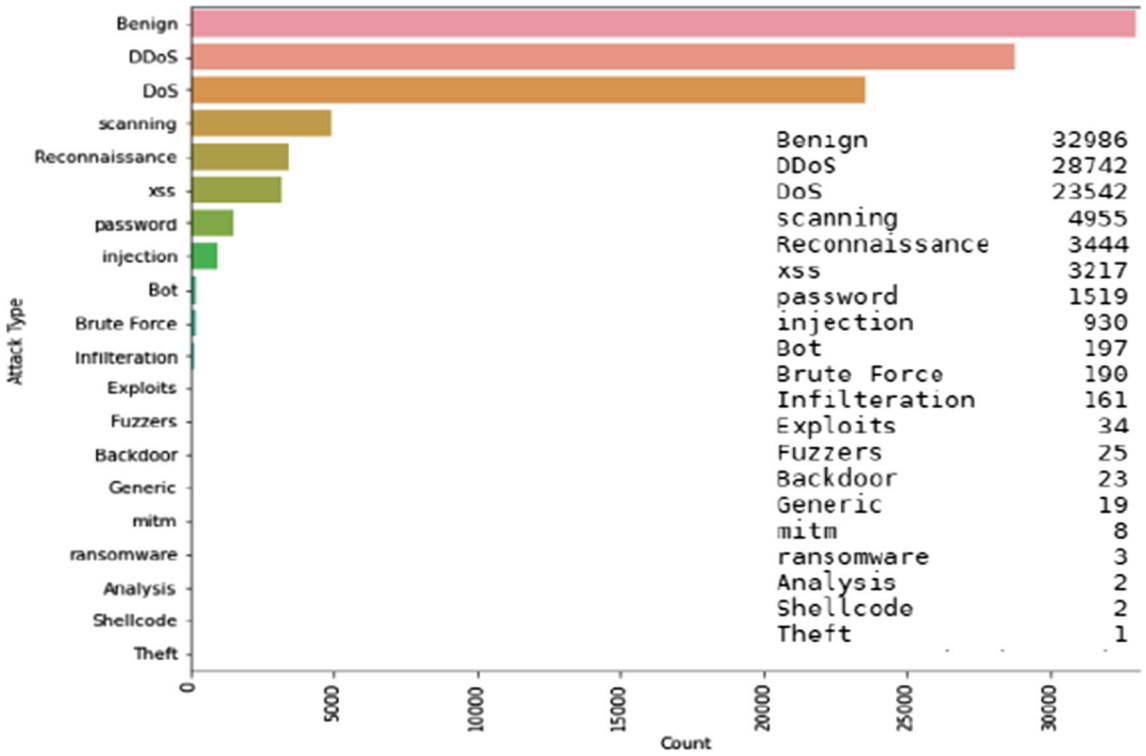


Fig. 1 Attack classes

```
['DURATION_IN',
 'LONGEST_FLOW_PKT',
 'SHORTEST_FLOW_PKT',
 'MAX_IP_PKT_LEN',
 'Label',
 'Dataset']
```

Fig. 2 Six most superlative features

new type of attack, even if it has a slight variation with the fingerprint of the attack, because it is too specific and can be altered to avoid matching. In addition, traffic may also be encrypted in order to bypass.

- Anomaly-based: its functionality is dependent on differences, at the pre-phase level, generating a baseline profile in advance as normal behavior that uses system services and resource usage for later use as a comparator issue, which is employed to detect and recognize abnormal activity of unknown threats, intrusions, or attacks. Needs continuous updating to optimize the system profile, considered both normal and abnormal, resulting in a normal change in activities that are not performed on a regular basis, ignoring important parameters of the command line or event log file, and resulting in a false positive.

3.3 Detection capabilities

Intrusion detection approaches often provide powerful features for performing in-depth analysis and evaluation of specific data. This generally provides more accurate and efficient detection and more operational flexibility in customization, modification, and tuning; however, specifying the characteristics of the types of events is absolutely essential first.

- Types of events detected: typically, at this stage, the information gathered from hosts or networks, the observed activities, the acceptable level, and the pattern profile between normal and abnormal behavior, are specified.
- Detection accuracy: IDS frequently generates false alerts and has significant issues with regard to concerns such as accuracy and efficiency. Since it is impossible to detect some missed intrusions with a single IDS, several more IDS are implemented, although they respond differently to the same packet traces and provide various alert sets. In fact, an attack unnoticed by one IDS might be discovered by another while analyzing the same data (e.g., false positive, false negative). Another major issue with accuracy is that it often requires serious customization and adaptation to accommodate the features of the monitored environment. A single sensor frequently monitors the traffic of

thousands of internal and external hosts. The number and variety of operating systems and applications used on the supervised network may be immense; additionally, operating systems and applications are constantly changing. As a result, a sensor cannot recognize all it sees. An enterprise, for example, may utilize ten distinct types and versions of Web servers (e.g., Apache, Nginx, etc.), while users could create sessions and access them using fifty various types and versions of browsers (e.g., internet explorer, Firefox, Avast, etc.). Each web server and browser may have distinct communication attributes. This could have an effect on the accuracy of the analysis.

- Tuning and customization: IDS frequently require extensive customization and tuning, to dramatically improved detection accuracy. This allows the IDS to make more informed judgements regarding preventative measures and more precisely categorize alarms.
- Technology limitations: although IDS are a valuable complement to an information security architecture, they have several limitations on some functions that prevent them from performing some tasks (e.g., determining attacks automatically without human intervention, compensating for issues in the authenticity and validity of information sources, making up for missing or inadequate security systems, and delays in alert generation while detecting freshly published attacks when the network or processing demand is heavy).

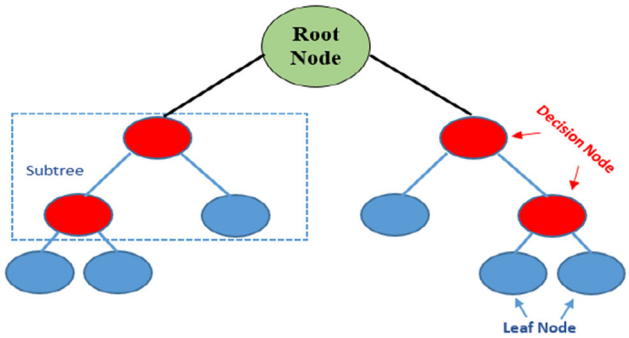


Fig. 4 DT diagram

4 NIDS dataset

Data is an essential component and the key to success in the field of ML; sufficient volumes of data allow you to train an algorithm with the goal of finding predictable patterns and validating the approach. Now, it is important to collect this data into datasets. This means that the collected data should be uniform and understandable to a machine. For this reason, after collecting the data, it is important to understand what features a proper dataset has and preprocess it by cleaning and completing it. Therefore, to practice machine learning algorithms, the researchers can use any dummy dataset, but live production data is preferred whenever possible. It is unattainable to overestimate the value of datasets in machine learning research, as they are representations of facts about the world that cannot be experienced directly and are not often replicated [29, 30].

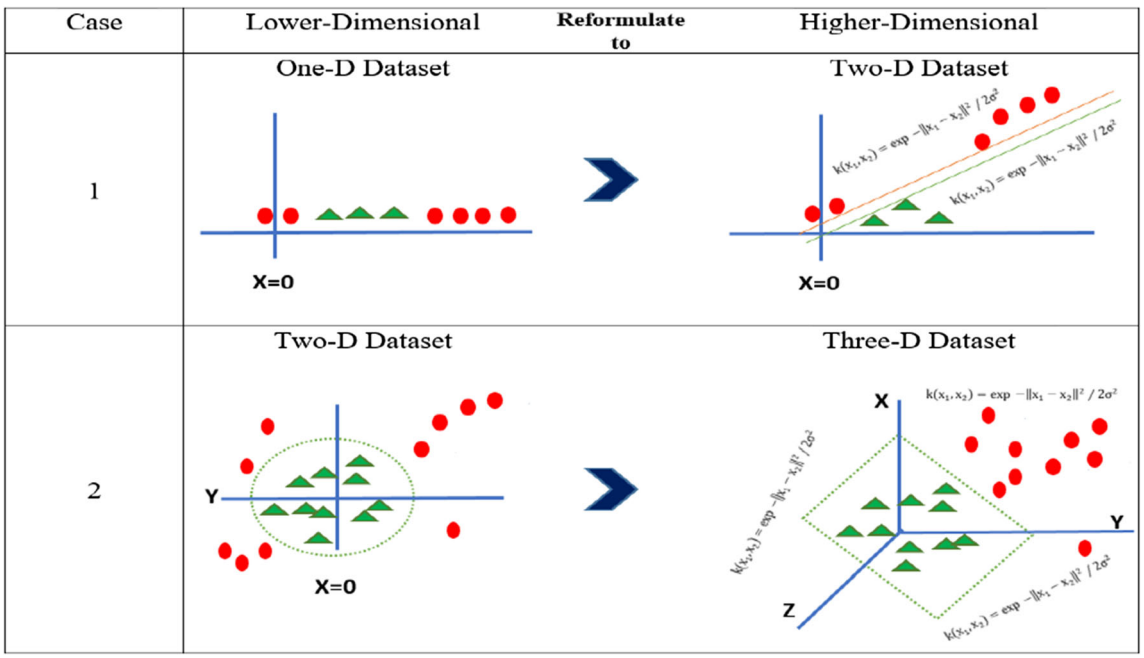


Fig. 3 Mapping dataset diagram

A popular source for machine learning datasets that are freely available on Kaggle, the university of Queensland Australia, UCI Machine Learning Repository, AWS. A number of machine learning datasets are also freely available to the public, e.g., NIDS, NF-UQ-NIDS-V2, UNSW-NB15, CICIDS 2017, ADFA-LD, NSL-KDD, KDD, and DARPA.

The University of Queensland in Australia and Kaggle have recently released the NF-UQ-NIDS-V2 dataset, which contains a wide range of variously generated, network-based anomaly attacks simulated in an environment containing events of traffic flow totaling nearly 76 million rows, and 46 columns, roughly divided into a 1:3 ratio for normal to abnormal. The dataset CSV file size is 13.73 GB [32, 32].

In this work, 100 K records were imported and loaded from the master source file of the NF-UQ-NIDS-v2 dataset to train classical algorithms. For training neural network algorithms, the dataset was then increased until it had 10 million records. And the following are the preparation stages (from 1 to 8) in which a data farm is analyzed, handled, and structured in a manner that maximizes accuracy and efficiency.

4.1 Libraries

The initial step import some libraries that are used to perform some specific functions, e.g., PANDAS to read csv files, SKLEARN for missing values and scaling, NUMPY to perform complex mathematical operations, MATPLOTLIB to plot charts, tensorflow and keras for neural network.

4.2 Data load

Determining the path of the data source, and reading it with the number of specific rows we required.

4.3 Descriptive analysis

Description of the 100 k records that were read from the data source to train classical algorithms, showing the shape of the rows, columns, headers, and data types, and verifying that there are no duplicate rows or missing values, neither NaN nor NULL, in the whole data frame. Figure 1 shows lists of count values for the attack classes generated, and there are two types of events totaling 100 k records: the first is normal with 32,986 records, and the second is anomalous with 67,014 records, containing 19 subsets. Consequently, in preparation to train neural networks, these records were increased by one 100-fold, and the attacks had reached 20 subsets.

4.4 Cleansing and imputer

We achieved it by fixing the values from the results of the previous stage, if they exist. Dropping duplicate rows and using equations for missing values such as mean, median, most frequent, and constant.

4.5 Scaling numerical values

Where the diversity of features and their distribution may have different values across the data farm, it will lead to difficulties in handling and formulating the issue, hence training the algorithm fails to work. There are many techniques in the preprocessing module from the SKLEARN library. (e.g., StandardScaler, MinMaxScaler, MaxAbsScaler, Normalizer, Binarizer, PolynomialFeatures, FunctionTransformer). We utilize (MinMaxScaler with both attributes copy = True and fere_range = 0, 1).

4.6 Encoding categorical attributes

The data farm is prepared for a machine learning algorithm that works perfectly with mathematics and numbers. From this point on, categorical variables should be encoded into

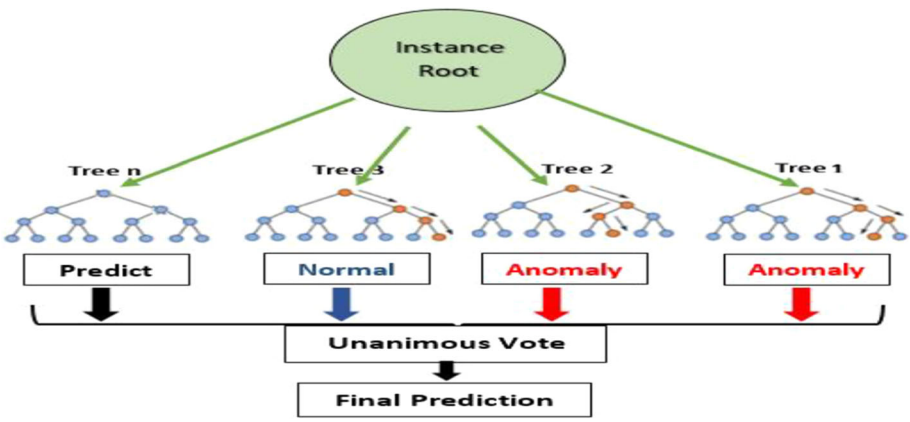


Fig. 5 RFC diagram

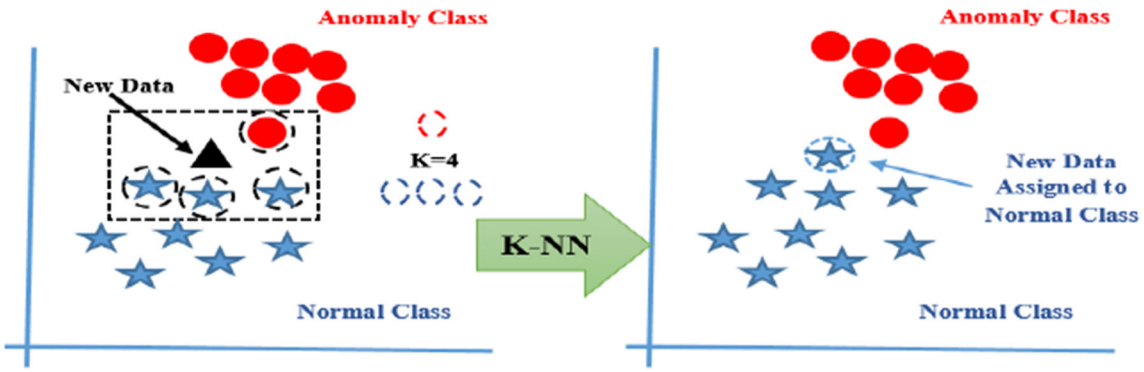
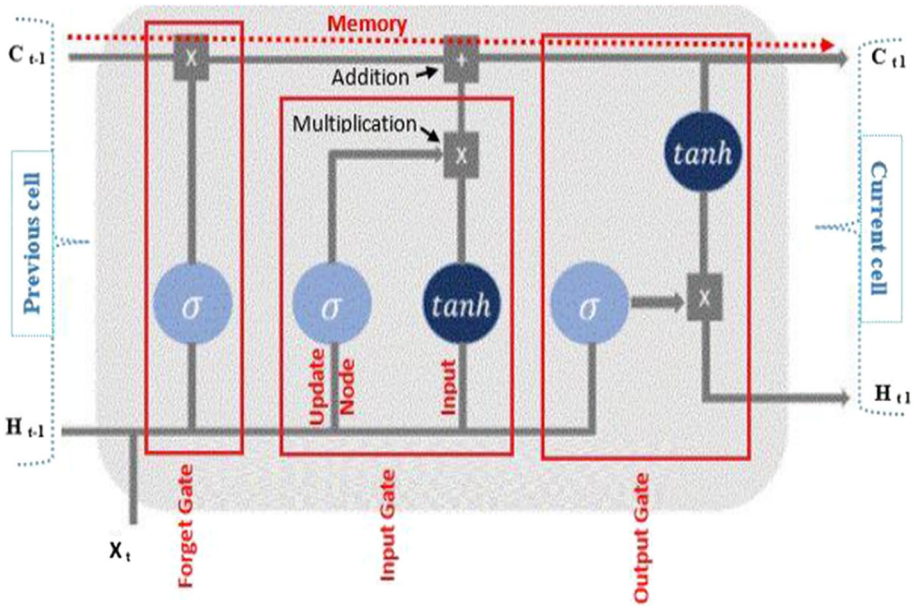


Fig. 6 KNN diagram (Colour figure online)

Fig. 7 LSTM cell diagram



numbers. In this dataset, steps taken by LabelEncoder through (select_dtypes include 'object,' en-coder.fit_tr-nasform, then separate the 'Y = output' column from encoded data to be the train shape of 100,000 rows and 45 columns).

4.7 Feature elimination

We exclusively applied it in classical algorithms only. It is a technique specific to selecting the most required and effective features and excluding the rest. Feature selection chosen based on the extent to which it relates to the output. The technique utilized was recursive feature elimination (RFE) with various attributes (e.g., estimator numbers of 100, max depth of 5, criterion of entropy, oob_score of true, and n_jobs = − 1) to select the top six superlative features, as shown in Fig. 2.

4.8 Data split

The final stage in preparing and managing a data farm, which is separated into two independent subsets, 60 and 40% of the original dataset, the first is to train the algorithm and validate its learning on this data to eventually become a model capable of accurate prediction. Then comes the second part of this stage, which is testing an algorithm on unseen data to evaluate the accuracy of the test. To verify the quality of the data and the accuracy of the model, the results of each test dataset were compared to the training dataset.

5 Machine learning algorithms

Artificial intelligence (AI) is the major domain, and machine learning (ML) is a subfield of it that concentrates on the use of datasets and algorithms to allow computers to

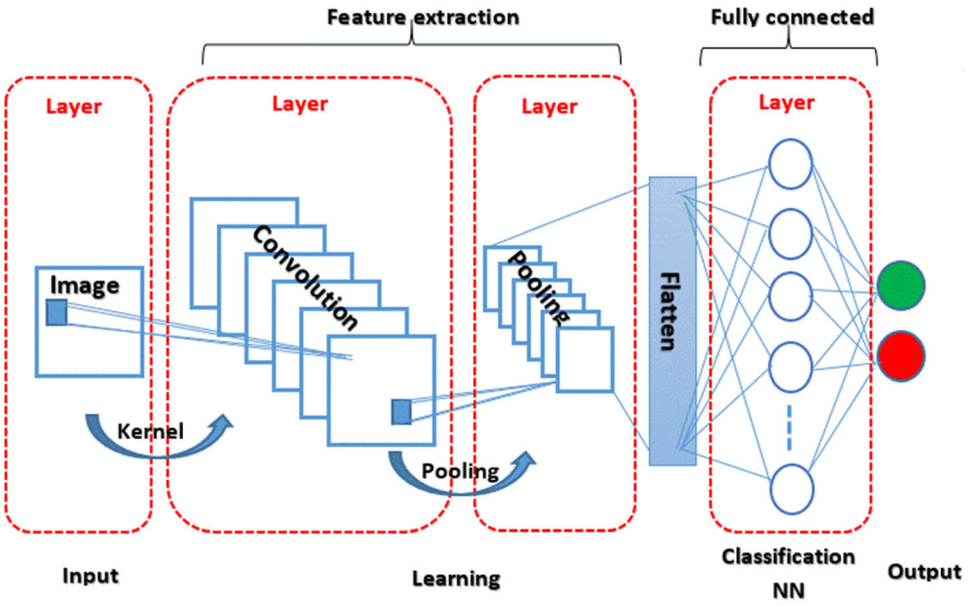


Fig. 8 CNN diagram

Table 2 Environment setup

Platform	11th Gen Intel(R) Core(TM) i7-11800H @ 2.30 GHz. RAM 32 GB. Windows 11 enterprise 64-bit
Tools	Acnaconda3-2022.10-Windows-X86_64, Spyder IDS 5.2.2, and Python 3.9.13
Libraries	Sklearn 1.0.2, Pandas 1.4.4, Numpy 1.23.5, Matplotlib 3.6.2, catboost 1.1.1, and tensorflow 2.10.0, keras 2.13.1
Modules/ layers	Pr-eprocessing, ensemble, optuna, feature_selection, model_selection, metrics, svm, neighbors, tree, Sequential, conv1D, Maxpool1D, Flatten, Dense, Dropout
Classes	MinMaxScaler, LabelEncoder, RFE, RandomForestClassifier, SVC, cross_val_score, accuracy_score, create_study, classification_report, KNeighborsClassifier, DecisionTreeClassifier, CatBoostClassifier, GradientBoostingClassifier, cross_val_predict, train_test_split, KerasClassifier, StratifiedKFold

Metrics		Formula	
Accuracy		(TP+TN)/(TP+TN+FP+FN)	TP: True Positive FP: False Positive TN: True Negative FN: False Negative
Classification Report	Precision	TP/(TP+FP)	
	Recall	TP/(TP+FN)	
	F1-score	2*((precision*recall)/(precision+recall))	

Fig. 9 Performance measures

efficiently manage the procedure of forming predictive methods and identifying patterns to simulate the manner in which individuals behave and continually improve accuracy [33–35].

ML depends on various algorithms to address data issues, which means there is no one-size-fits-all algorithm that is the optimum solution for resolving an issue. The type of issue you want to handle, the variety of variables, the appropriate model for that as well, and so on, determine the algorithm type used. Machine learning and deep learning differ in two ways: the first is the analytical model process, and the second is the learning mechanisms of each algorithm [36, 37]. Without exhaustive human-based training or intervention, machine learning models can provide IDS methodologies for detecting current, new, and sophisticated attacks. While there are several models of classification, this study will concentrate on both classical algorithms and neural networks. Feature selection and optimum attributes are essential to the core functionality of classical algorithms; these capabilities will detect patterns and correlations. The neural networks require a substantial amount of data and well-constructed layers of interconnected neurons with certain values.

5.1 Classical algorithms

Classical algorithms are characterized by training on a smaller data set, having simple correlations, and demanding human intervention to choose the features, which reduces the complexity of the data and makes patterns easier to detect for learning algorithms.

Table 3 SVC model validation and evaluation

Hyper-parameter values set (kernel = 'rbf,'cache_size = 2000,max_iter = 1000,C = 100,gamma = 'auto')									
Training model accuracy: 0.9839333333333333					Testing model accuracy: 0.982625				
	Precision	Recall	f1-Score	Support		Precision	Recall	f1-Score	Support
Analysis	0.50	1.00	0.67	2	Analysis	0.00	0.00	0.00	0
Backdoor	1.00	1.00	1.00	17	Backdoor	0.67	1.00	0.80	6
Benign	1.00	1.00	1.00	19,776	Benign	1.00	1.00	1.00	13,210
Bot	1.00	1.00	1.00	119	Bot	1.00	1.00	1.00	78
Brute force	1.00	0.97	0.99	109	Brute force	1.00	0.99	0.99	81
DDoS	0.99	0.99	0.99	17,339	DDoS	0.99	0.99	0.99	11,403
DoS	0.98	0.99	0.98	14,042	DoS	0.98	0.99	0.98	9500
Exploits	0.94	0.83	0.88	18	Exploits	0.92	0.69	0.79	16
Fuzzers	0.72	0.87	0.79	15	Fuzzers	0.50	0.60	0.55	10
Generic	1.00	0.73	0.84	11	Generic	0.88	0.88	0.88	8
Infiltration	1.00	1.00	1.00	97	Infiltration	0.98	0.83	0.90	64
Reconnaissance	0.95	0.93	0.94	2042	Reconnaissance	0.93	0.93	0.93	1402
Shellcode	1.00	1.00	1.00	1	Shellcode	0.00	0.00	0.00	1
Injection	0.78	0.57	0.66	562	Theft	0.00	0.00	0.00	1
mitm	0.00	0.00	0.00	6	Injection	0.72	0.60	0.65	368
Password	0.79	0.92	0.85	925	mitm	0.00	0.00	0.00	2
Ransomware	1.00	0.50	0.67	2	Password	0.80	0.90	0.85	594
Scanning	.99	0.99	0.99	2985	Ransomware	0.00	0.00	0.00	1
xss	95	0.95	0.95	1932	Scanning	0.99	0.99	0.99	1970
Accuracy			0.98	60,000	xss	0.95	0.94	0.94	1285
Macro avg	87	0.85	0.85	60,000	Accuracy			0.98	40,000
Weighted avg	0.98	0.98	0.98	60,000	Macro avg	0.66	0.67	0.66	40,000
Weighted avg	0.98	0.98	0.98	40,000					

5.1.1 Support vector classifier (SVC)

The capability of this module is to handle and transform highly complex data with a minimal number of training datasets; it essentially addresses the issue of nonlinear data separation through the kernel trick technique by using more than one hyperplane line, which is one of its main strengths. Utilizing the kernel attribute via the radial basis function (RBF), it can be expressed mathematically as in Eq. (1).

$$k(x_1, x_2) = \exp -x_1 - x_2^2/2\sigma^2 \quad (1)$$

where '$\|X_1 \_ X_2\|$' refer to space between 1 and 2. 'σ' refer to variance and our hyper-parameter.

We reformulate the two cases by mapping the datasets from lower-dimensional to higher-dimensional space to group each class together as shown in Fig. 3.

In case one, as the figure shows when mapping the datasets from one-dimensional to two-dimensional, one feature expressed the X-dimension, so the result is the green class mediating the red class. For a nonlinearly separable issue like those classes, we must remap the Y-dimension to a higher dimension, which means reformulating the same values in a different space by two equations, one to determine the red line and the other for the green line.

In case two, mapping the dataset from two-dimensional to three-dimensional is also the same as the issue in case 1, but with different dimensions to separate the two classes from each other through reformulating the two-dimensional (X, Y) to the three-dimensional (X, Y, Z) by three equations.

5.1.2 Decision tree classifier (DTC)

It is termed a decision tree since, similar to a hierarchy tree constructed to manage processes top-down, it initiates with the root node by utilizing the most informative feature by asking a simple question with a conclusive answer (yes or no), and then the subtrees grow down by repeating this process which means that growth creates two types of nodes that appear. The first is the decision node and

Table 4 DT model validation and evaluation

Hyper-parameter values set (def func.create study: dt_max_depth = '2,20,' n_trials = 40, direction = 'maximize')									
Training model accuracy: 0.9948333333333333					Testing model accuracy: 0.987825				
	Precision	Recall	f1-Score	Support		Precision	Recall	f1-Score	Support
Analysis	0.00	0.00	0.00	2	Backdoor	0.86	1.00	0.92	6
Backdoor	1.00	0.88	0.94	17	Benign	1.00	1.00	1.00	13,210
Benign	1.00	1.00	1.00	19,776	Bot	1.00	1.00	1.00	78
Bot	1.00	1.00	1.00	119	Brute force	0.95	0.99	0.97	81
Brute force	1.00	0.98	0.99	109	DDoS	0.99	0.99	0.99	11,403
DDoS	1.00	1.00	1.00	17,339	DoS	0.98	0.99	0.99	9500
DoS	0.99	1.00	0.99	14,042	Exploits	1.00	0.56	0.72	16
Exploits	1.00	0.72	0.84	18	Fuzzers	0.64	0.70	0.67	10
Fuzzers	0.63	0.80	0.71	15	Generic	0.88	0.88	0.88	8
Generic	1.00	0.64	0.78	11	Infiltration	1.00	0.83	0.91	64
Infiltration	1.00	0.98	0.99	97	Reconnaissance	0.97	0.95	0.96	1402
Reconnaissance	0.99	0.95	0.97	2042	Shellcode	0.00	0.00	0.00	1
Shellcode	0.00	0.00	0.00	1	Theft	0.00	0.00	0.00	1
Injection	0.99	0.90	0.94	562	Injection	0.82	0.78	0.80	368
mitm	1.00	0.17	0.29	6	mitm	0.00	0.00	0.00	2
Password	0.98	0.98	0.98	925	Password	0.91	0.89	0.90	594
Ransomware	1.00	1.00	1.00	2	Ransomware	0.50	1.00	0.67	1
Scanning	1.00	1.00	1.00	2985	Scanning	1.00	0.99	0.99	1970
xss	0.97	1.00	0.98	1932	xss	0.94	0.98	0.96	1285
Accuracy			0.99	60,000	Accuracy			0.99	40,000
Macro avg	0.87	0.79	0.81	60,000	Macro avg	0.76	0.76	0.75	40,000
Weighted avg	0.99	0.99	0.99	60,000	Weighted avg	0.99	0.99	0.99	40,000

contains branches, while the other is a normal leaf node that does not have branches. Figure 4 demonstrates the hierarchy of decision tree construction.

We explain how a decision tree works through the most commonly used choice metrics, which can be mathematically expressed as in Eqs. (2), (3).

$$\text{Entropy}(s) = -P + \log(P+) - P - \log(P-) \tag{2}$$

where (s) dataset, $P+$ probability of yes, $P-$ probability of no, *Log* number of classes

The result,

Between 0 and 1 (Zero = refers to a superlative feature that affects directly the output, and One= refers to a weak feature that does not have any effect on the outputs).

$$E(s) - (C(G1)/C(s) * E(G1) + C(G2)/C(s) * E(G2)) \tag{3}$$

where E entropy, (s) dataset, C count, $G1$ group 1, $G2$ group 2

The result,

Select the group with the highest value.

5.1.3 Random forest classifier (RFC)

It was designed by using the concept of parallel ensemble learning, which is a hierarchical combining of decision tree classifiers that uses a predefined likelihood to select the ultimate superlative feature. The random forest decides the final prediction result depending on the unanimous vote of the forecasts it receives from each tree. Consider Fig. 5.

5.1.4 K-Nearest neighbors (KNN)

It is one of the simplest types of classification algorithms in machine learning models. Figure 6 shows the method it utilizes when it is fed new data like (black triangle) to classify it under a specific class (blue or red). It first determines the K factor by assigning a specific number (k = 4). Meanwhile, the scope of nearby points, that it will work on, as it will re-sort for all neighbor points, calculate the distance between the new data and these points (3 blue and 1 red), and classify the new data according to the majority of its nearest neighbors (the new data is classified under blue).

Table 5 RFC model validation and evaluation

Hyper-parameter values set (def func.create study: rf_max_depth = '2,32,' n_trials = 30, direction = 'maximize')									
Training model accuracy: 0.9973666666666666					Testing model accuracy: 0.990775				
	Precision	Recall	f1-Score	Support		Precision	Recall	f1-Score	Support
Analysis	1.00	1.00	1.00	2	Analysis	0.00	0.00	0.00	0
Backdoor	1.00	1.00	1.00	17	Backdoor	0.67	1.00	0.80	6
Benign	1.00	1.00	1.00	19,776	Benign	1.00	1.00	1.00	13,210
Bot	1.00	1.00	1.00	119	Bot	1.00	1.00	1.00	78
Brute force	1.00	0.98	0.99	109	Brute force	1.00	0.99	0.99	81
DDoS	1.00	1.00	1.00	17,339	DDoS	1.00	0.99	0.99	11,403
DoS	0.99	1.00	1.00	14,042	DoS	0.99	0.99	0.99	9500
Exploits	1.00	0.94	0.97	18	Exploits	0.92	0.75	0.83	16
Fuzzers	1.00	1.00	1.00	15	Fuzzers	0.64	0.70	0.67	10
Generic	0.92	1.00	0.96	11	Generic	1.00	0.88	0.93	8
Infiltration	1.00	1.00	1.00	97	Infiltration	0.98	0.95	0.97	64
Reconnaissance	1.00	0.97	0.98	2042	Reconnaissance	0.98	0.95	0.96	1402
Shellcode	1.00	1.00	1.00	1	Shellcode	0.00	0.00	0.00	1
Injection	1.00	0.93	0.97	562	Theft	0.00	0.00	0.00	1
mitm	1.00	0.50	0.67	6	Injection	0.87	0.81	0.84	368
Password	1.00	0.99	0.99	925	mitm	0.00	0.00	0.00	2
Ransomware	1.00	1.00	1.00	2	Password	0.95	0.92	0.93	594
Scanning	1.00	1.00	1.00	2985	Ransomware	0.00	0.00	0.00	1
xss	0.98	1.00	0.99	1932	Scanning	1.00	1.00	1.00	1970
Accuracy			1.00	60,000	xss	0.95	0.98	0.97	1285
Macro avg	0.99	0.96	0.97	60,000	Accuracy			0.99	40,000
Weighted avg	1.00	1.00	1.00	60,000	Macro avg	0.70	0.70	0.69	40,000
					Weighted avg	0.99	0.99	0.99	40,000

5.1.5 Boost algorithms

The core idea depends on generating numerous weaker classifiers and integrating their prediction results to assemble one stringent classifier to achieve the highest accuracy. CatBoost classifier (CBC) and gradient boosting classifier (GBC) are two types of boosting algorithms.

5.2 Neural network algorithms

Neural network (NN) algorithms are characterized by mimicking the structure and function of the human cognitive system through convoluted correlations, similar to how a human would obtain conclusions. NN trains on massive data and the ability to process it in its original form, and when there is inadequate domain expertise for feature selection, it learns gradually and discovers exceptional features from the data.

5.2.1 Long short-term memory

Recurrent neural networks (RNNs) have been extended by the most powerful version that can buffer long-term dependencies in sequential data, called long short-term memory (LSTM). The LSTM structure involves a cell-chain; as shown in Fig. 7, each one of these cells has a memory and three fully connected gates (forget, input, and output), which control the information flow both inside and outside of the cell. Each LSTM cell's output is fed into the next cell in the chain.

Here, C_{t-1} C_{t-1} is hidden cell state of long-term memory, H_{t-1} is hidden state of the previous cell, short-term memory, and X_t is current input. σ sigmoid activation function output value (0 delete or 1 keep). *tanh* activation function output value ($-$ 1 and 1). H_{t1} current cell output.

The memory that makes it easier to remember values of data over a random time interval from (C_{t-1}) to (C_{t1}). The forget gate determines which information the LSTM will proceed to keep and which to delete from the previous cell state (C_{t-1}) utilizing a sigmoid activation function

Table 6 KNN model validation and evaluation

hyper-parameter values set (def func.create study: n_neighbors = '2,20,' n_trials = 2, direction = 'maximize')									
Training model accuracy: 0.98865					Testing model accuracy: 0.9816				
	Precision	Recall	f1-Score	Support		Precision	Recall	f1-Score	Support
Analysis	0.20	0.50	0.29	2	Analysis	0.00	0.00	0.00	0
Backdoor	0.94	0.88	0.91	17	Backdoor	0.50	0.83	0.62	6
Benign	1.00	1.00	1.00	19,776	Benign	1.00	1.00	1.00	13,210
Bot	0.98	1.00	0.99	119	Bot	0.97	1.00	0.99	78
Brute force	1.00	0.97	0.99	109	Brute force	0.98	0.99	0.98	81
DDoS	0.99	0.99	0.99	17,339	DDoS	0.99	0.99	0.99	11,403
DoS	0.99	0.99	0.99	14,042	DoS	0.98	0.99	0.98	9500
Exploits	0.74	0.78	0.76	18	Exploits	0.42	0.31	0.36	16
Fuzzers	0.37	0.47	0.41	15	Fuzzers	0.22	0.20	0.21	10
Generic	0.67	0.18	0.29	11	Generic	0.50	0.12	0.20	8
Infiltration	0.97	0.91	0.94	97	Infiltration	0.95	0.81	0.87	64
Reconnaissance	0.98	0.94	0.96	2042	Reconnaissance	0.96	0.92	0.94	1402
Shellcode	0.00	0.00	0.00	1	Shellcode	0.00	0.00	0.00	1
Injection	0.84	0.79	0.82	562	Theft	0.00	0.00	0.00	1
mitm	0.00	0.00	0.00	6	Injection	0.70	0.68	0.69	368
Password	0.91	0.91	0.91	925	mitm	0.00	0.00	0.00	2
Ransomware	0.00	0.00	0.00	2	Password	0.85	0.83	0.84	594
Scanning	0.99	0.99	0.99	2985	Ransomware	0.00	0.00	0.00	1
xss	0.96	0.98	0.97	1932	Scanning	0.98	0.99	0.99	1970
Accuracy			0.99	60,000	xss	0.95	0.95	0.95	1285
Macro avg	0.71	0.70	0.69	60,000	Accuracy			0.98	40,000
Weighted avg	0.99	0.99	0.99	60,000	Macro avg	0.60	0.58	0.58	40,000
					Weighted avg	0.98	0.98	0.98	40,000

(0 = delete, 1 = keep). It achieves this by assessing the new input data (X_t) and the data gained from the previously hidden state (H_{t-1}). The input gate, as a method to alter the state of memory, determines which value from the input should be utilized. The values to pass through are 0 or 1, depending on the sigmoid function. Secondly, the tanh activation function assigns weight to the values that are passed, determining their relevance on a scale between ($-$ 1 and 1). The output gate is currently in this hidden state. The newly updated cell state via a tanh function multiplied by the output passed from a sigmoid function determines the hidden state of the current cell output (H_{t1}).

5.2.2 Convolutional neural network

A convolutional neural network (CNN) broadly used for machine vision, and it was designed-based on the human visual neuron simulator. Each layer of CNN is only connected to a portion of the neurons, with the last layer at the tail of the network being fully connected. CNN applies filters separately to each pixel in an image to learn the pattern features in detail. The CNN diagram is depicted in Fig. 8, where the network architecture is constructed from four layers (input, convolutional, pooling, and fully connected). As the CNN diagram below shows, the input layer mainly feeds the dataset to the next layer. The kernel in the convolution layer works as a filter for the feature detector in the image. A pooling layer used the downscaling technique to reduce the feature map to prevent over-fitting issues by summarizing via two common methods (average pooling and max pooling). Then flattening, which converts the full matrix of pooled feature maps directly into just one column before feeding that column to the fully connected layer of the neural network. In the fully connected layer called the dense, all neurons have direct connectivity for transforming their output into any number of classes. Deep learning is composed of multiple layers of convolution and pooling. Once we proceed more deeply within CNN, it ultimately recognizes deeper characteristic features.

Table 7 CatBoost model validation and evaluation

Hyper-parameter set value to only one attribute (verbose = 0)									
Training model accuracy: 0.9941166666666666					Testing model accuracy: 0.990475				
	Precision	Recall	f1-Score	Support		Precision	Recall	f1-Score	Support
Analysis	1.00	1.00	1.00	2	Backdoor	0.86	1.00	0.92	6
Backdoor	1.00	1.00	1.00	17	Benign	1.00	1.00	1.00	13,210
Benign	1.00	1.00	1.00	19,776	Bot	1.00	1.00	1.00	78
Bot	1.00	1.00	1.00	119	Brute force	1.00	0.99	0.99	81
Brute force	1.00	0.97	0.99	109	DDoS	0.99	0.99	0.99	11,403
DDoS	0.99	1.00	1.00	17,339	DoS	0.99	0.99	0.99	9500
DoS	0.99	0.99	0.99	14,042	Exploits	0.92	0.75	0.83	16
Exploits	1.00	1.00	1.00	18	Fuzzers	0.64	0.70	0.67	10
Fuzzers	1.00	1.00	1.00	15	Generic	0.62	1.00	0.76	8
Generic	1.00	1.00	1.00	11	Infiltration	1.00	0.94	0.97	64
Infiltration	1.00	1.00	1.00	97	Reconnaissance	0.98	0.95	0.97	1402
Reconnaissance	0.99	0.96	0.97	2042	Shellcode	0.00	0.00	0.00	1
Shellcode	1.00	1.00	1.00	1	Theft	0.00	0.00	0.00	1
Injection	0.98	0.88	0.93	562	Injection	0.88	0.80	0.84	368
mitm	1.00	0.17	0.29	6	mitm	0.00	0.00	0.00	2
Password	0.98	0.98	0.98	925	Password	0.94	0.93	0.93	594
Ransomware	1.00	1.00	1.00	2	Ransomware	0.00	0.00	0.00	1
Scanning	1.00	1.00	1.00	2985	Scanning	1.00	0.99	1.00	1970
xss	0.97	1.00	0.98	1932	xss	0.96	0.99	0.97	1285
Accuracy			0.99	60,000	Accuracy			0.99	40,000
Macro avg	1.00	0.94	0.95	60,000	Macro avg	0.72	0.74	0.73	40,000
Weighted avg	0.99	0.99	0.99	60,000	Weighted avg	0.99	0.99	0.99	40,000

6 Experiment environment and results

The experiment addressed in this work conducted on dataset, tools, libraries, modules, and classes, as indicated in Table 2. They are all powerful and agile open sources.

In terms of the performance measures shown in Fig. 9 for validating and evaluating ML models, this work seeks to achieve high incident detection accuracy, essentially in the testing dataset. The model accuracy and classification report presented below are the main criteria to examine. In addition, the difference in detection accuracy between training and testing for recognizing over-fitting and under-fitting issues must be noted.

Also, six machine learning classification algorithms configured with hyper-parameter values for each one separately were trained, validated, tested, and evaluated in this environment, as shown in Tables 3, 4, 6, 7, 8, 9 and 10 which will be explained in the context of this section.

The SVC algorithm utilized hyper-parameter value sets as shown in Table 3 and achieved (98.26%) for testing model accuracy. Bot (100%) had the highest detection rate, followed by three other types (Brute Force, DDoS, and Scanning) that obtained (99%), while there are four types (Shellcode, Theft, mitm, and ransomware) that have not been detected. al model.

The DT algorithm utilized hyper-parameter value sets as shown in Table 4 and achieved 98.78% accuracy for the testing model. Bot (100%) had the highest detection rate, followed by DDoS, DoS, and scanning as having the highest detection rate, while there are three types (Shellcode, Theft, and mitm) that have not been detected due to their small number.

For the RFC algorithm, we utilized hyper-parameter value sets as depicted in Table 5 and achieved a 99.07% accuracy rate for the testing model. Bot and scanning (100%) had the highest detection rate, while Brute Force,

Table 8 Gradient boosting model validation and evaluation

Hyper-parameter set value to only one attribute (random_state = 0)									
Training model accuracy: 0.9912					Testing model accuracy: 0.98805				
	Precision	Recall	f1-Score	Support		Precision	Recall	f1-Score	Support
Analysis	0.00	0.00	0.00	2	Backdoor	0.75	1.00	0.86	6
Backdoor	1.00	0.88	0.94	17	Benign	1.00	1.00	1.00	13,210
Benign	1.00	1.00	1.00	19,776	Bot	1.00	0.99	0.99	78
Bot	1.00	1.00	1.00	119	Brute force	0.98	0.99	0.98	81
Brute force	0.98	1.00	0.99	109	DDoS	0.99	0.99	0.99	11,403
DDoS	0.99	1.00	0.99	17,339	DoS	0.99	0.99	0.99	9500
DoS	0.99	0.99	0.99	14,042	Exploits	0.92	0.69	0.79	16
Exploits	1.00	0.72	0.84	18	Fuzzers	0.00	0.00	0.00	10
Fuzzers	0.00	0.00	0.00	15	Generic	0.02	0.12	0.04	8
Generic	0.09	0.45	0.15	11	Infiltration	0.98	0.83	0.90	64
Infiltration	1.00	1.00	1.00	97	Reconnaissance	0.97	0.94	0.96	1402
Reconnaissance	0.98	0.94	0.96	2042	Shellcode	0.00	0.00	0.00	1
Shellcode	0.00	0.00	0.00	1	Theft	0.00	0.00	0.00	1
Injection	0.96	0.84	0.90	562	Injection	0.85	0.78	0.81	368
mitm	1.00	1.00	1.00	6	mitm	0.00	0.00	0.00	2
Password	0.97	0.96	0.97	925	Password	0.93	0.90	0.91	594
Ransomware	1.00	0.50	0.67	2	Ransomware	0.00	0.00	0.00	1
Scanning	1.00	1.00	1.00	2985	Scanning	1.00	1.00	1.00	1970
xss	0.97	0.99	0.98	1932	xss	0.96	0.98	0.97	1285
Accuracy			0.99	60,000	Accuracy			0.99	40,000
Macro avg	0.79	0.75	0.76	60,000	Macro avg	0.65	0.64	0.64	40,000
Weighted avg	0.99	0.99	0.99	60,000	Weighted avg	0.99	0.99	0.99	40,000

DDoS, and Dos had the next highest detection rates, while there are four types (Shellcode, Theft, mitm, and ransomware) that have not been detected because their total number did not exceed five.

The testing model had a 98.16% accuracy rate when using the KNN algorithm with hyper-parameter value sets, as demonstrated in Table 6. The detection rate for Bot, DDoS, and scanning was the highest. Four categories (Shellcode, Theft, mitm, and Ransomware) were not detected since their total number was not greater than five.

After implementing the categorical boost algorithm with hyper-parameter value sets, the testing model had a 99.04% accuracy rate, as demonstrated in Table 7. Bot anomaly event had the greatest detection rate (100%), while the accuracy rate for Brute Force, DDoS, DoS, and scanning was next. The same four anomalous events (Shellcode, Theft, Mitm, and Ransomware) that were prevalent in earlier models happened again and were detected for the same reasons.

The testing model achieved 98.80% accuracy after adopting the gradient boosting algorithm using hyper-parameter value settings, which is apparent in Table 8. Scanning anomaly had the highest detection rate (100%), followed by Bot, DDoS, and DoS. Anomalies were not recognized (Fuzzers, Shellcode, Theft, Mitm, and Ransomware). The variation in detection accuracy between testing and training is less than 0.33%, indicating that it is within the acceptable range for equity and quality.

The long short-term memory (LSTM) algorithm utilized hyper-parameter value sets as shown in Table 9 and achieved 98.87% accuracy for the testing model. Bot, Infiltration, and scanning (100%) had the highest detection rate, followed by DDoS and DoS, as having the highest detection rate, while there are two types (analysis and worms) that have not been detected.

For the convolution neural network (CNN) algorithm, we utilized hyper-parameter value sets as shown in Table 10 and achieved a 98.56% accuracy rate for the testing model. Bot (100%) had the highest detection rate, while (Brute Force, DDoS, Dos, Infiltration, and Scanning) had the next highest detection rates, while there are two types (analysis and worms) that have not been detected.

Table 9 Long short-term memory (LSTM)

Hyper-parameter set value to build: input.layers.LSTM (128), four hidden layers.Dense(256, 512,256, and 128, activation = 'relu'), output.layers.Dense(n_class, activation = 'softmax')									
Hyper-parameter set value to compile: optimizer = 'adam,' loss = 'sparse_categorical_crossentropy,' and metrics = ['accuracy']									
Hyper-parameter set value to fit: epochs = 5, and batch_size = 512									
Training model accuracy: 0.9888491666666667					Testing model accuracy: 0.98875075				
	Precision	Recall	f1-Score	Support		Precision	Recall	f1-Score	Support
Analysis	0.00	0.00	0.00	181	Analysis	0.00	0.00	0.00	132
Backdoor	0.93	0.89	0.91	1500	Backdoor	0.93	0.89	0.91	1008
Benign	1.00	1.00	1.00	1,986,660	Benign	1.00	1.00	1.00	1,324,729
Bot	1.00	1.00	1.00	11,291	Bot	1.00	1.00	1.00	7620
Brute force	0.99	0.98	0.99	9935	Brute force	0.99	0.98	0.98	6545
DDoS	0.99	0.99	0.99	1,717,661	DDoS	0.99	0.99	0.99	1,144,686
DoS	0.98	0.99	0.99	1,411,559	DoS	0.98	0.99	0.99	940,644
Exploits	0.61	0.88	0.72	2502	Exploits	0.62	0.90	0.73	1716
Fuzzers	0.65	0.72	0.68	1771	Fuzzers	0.65	0.74	0.69	1153
Generic	0.99	0.68	0.81	1293	Generic	0.99	0.65	0.79	899
Infiltration	1.00	0.99	1.00	9249	Infiltration	1.00	0.99	1.00	6205
Reconnaissance	0.99	0.95	0.97	207,690	Reconnaissance	0.98	0.95	0.97	138,469
Shellcode	0.77	0.75	0.76	103	Shellcode	0.77	0.57	0.66	82
Theft	0.67	0.87	0.76	202	Theft	0.64	0.82	0.72	120
Worms	0.00	0.00	0.00	12	Worms	0.00	0.00	0.00	9
Injection	0.91	0.77	0.83	53,772	Injection	0.91	0.77	0.83	36,338
mitm	0.93	0.31	0.47	615	mitm	0.88	0.26	0.40	386
Password	0.91	0.94	0.93	91,081	Password	0.91	0.94	0.92	60,587
Ransomware	0.87	0.92	0.89	281	Ransomware	0.88	0.88	0.88	170
Scanning	1.00	1.00	1.00	299,086	Scanning	1.00	1.00	1.00	199,258
xss	0.94	0.96	0.95	193,556	xss	0.94	0.96	0.95	129,244
Accuracy			0.99	6,000,000	Accuracy			0.99	4,000,000
Macro avg	0.82	0.79	0.79	6,000,000	Macro avg	0.81	0.77	0.78	4,000,000
Weighted avg	0.99	0.99	0.99	6,000,000	Weighted avg	0.99	0.99	0.99	4,000,000

7 Conclusion

Utilizing machine learning approaches to enhance intrusion detection has received significant attention from the research community. Previous works have demonstrated that several ID techniques have been implemented to address the large variety and number of operating systems and applications used to monitor specific technologies. However, the recentness and quality of the dataset are crucial to success in this field, and better training for ML algorithms is essential. There is no one-size-fits-all algorithm for solving all issues, but there are various types available. Additionally, two major issues in ML models are over-fitting and under-fitting, and these factors can affect the model's effectiveness and precision. In this work, the researchers implement two separate classification layers of ML-based algorithms with the recently published NF-UQ-NIDS-v2 dataset, preprocessing two volumes of sample records (100 k and 10 million), using MinMaxScaler, LabelEncoder, recursive feature elimination, normalizing the data, and identifying hyper-parameters for classical algorithms and neural networks. The results show high detection accuracy rates for SVC (98.26%), DT (98.78%), RFC (99.07%), KNN (98.16%), CatBoost (99.04%), and gradient boosting (98.80%). In addition, neural network algorithms handled enormous amounts of data and achieved high detection results, which were 98.87% for LSTM and 98.56% for CNN. However, it was observed that some attacks were not detected when the number of anomaly events was low. In the future, we aim to improve the detection of those attacks, which are characterized by a limited frequency of events, by utilizing a multilayer

Table 10 Convolution neural network (CNN)

Hyper-parameter set value to build: (multiple layers of convolution and pooling), Input layers.Conv1D(filters = 64, kernel_size = 3, activation = 'relu'), MaxPooling1D(pool_size = 2), Flatten(), Two Hidden layers.Dense(128, and 256, activation = 'relu'), Output layers.Dense(n_class, activation = 'softmax')
Hyper-parameter set value to compile: optimizer = 'adam,' loss = 'sparse_categorical_crossentropy,' and metrics = ['accuracy'], cross-validation utilize StratifiedKFold (n_splits = 21, shuffle = True, and random_state = 42)
Hyper-parameter set value to fit: epochs = 5, and batch_size = 4096

Training model accuracy: 0.9857335					Testing model accuracy: 0.98567325				
	Precision	Recall	f1-Score	Support		Precision	Recall	f1-Score	Support
Analysis	0.00	0.00	0.00	181	Analysis	0.00	0.00	0.00	132
Backdoor	1.00	0.88	0.93	1500	Backdoor	1.00	0.89	0.94	1008
Benign	1.00	1.00	1.00	1,986,660	Benign	1.00	1.00	1.00	1,324,729
Bot	1.00	1.00	1.00	11,291	Bot	1.00	1.00	1.00	7620
Brute force	1.00	0.98	0.99	9935	Brute force	1.00	0.98	0.99	6545
DDoS	0.99	0.99	0.99	1,717,661	DDoS	0.99	0.99	0.99	1,144,686
DoS	0.98	0.99	0.99	1,411,559	DoS	0.98	0.99	0.99	940,644
Exploits	0.82	0.66	0.73	2502	Exploits	0.83	0.66	0.73	1716
Fuzzers	0.57	0.76	0.65	1771	Fuzzers	0.56	0.77	0.65	1153
Generic	0.77	0.74	0.75	1293	Generic	0.75	0.71	0.73	899
Infiltration	0.99	1.00	0.99	9249	Infiltration	0.99	0.99	0.99	6205
Reconnaissance	0.96	0.93	0.95	207,690	Reconnaissance	0.96	0.93	0.95	138,469
Shellcode	0.78	0.80	0.79	103	Shellcode	0.74	0.60	0.66	82
Theft	0.33	0.24	0.28	202	Theft	0.32	0.31	0.31	120
Worms	0.00	0.00	0.00	12	Worms	0.00	0.00	0.00	9
Injection	0.75	0.76	0.75	53,772	Injection	0.75	0.76	0.76	36,338
mitm	0.92	0.30	0.46	615	mitm	0.88	0.25	0.39	386
Password	0.90	0.90	0.90	91,081	Password	0.90	0.90	0.90	60,587
Ransomware	0.95	0.69	0.80	281	Ransomware	0.91	0.61	0.73	170
Scanning	0.99	0.99	0.99	299,086	Scanning	0.99	0.99	0.99	199,258
xss	0.94	0.93	0.94	193,556	xss	0.94	0.94	0.94	129,244
Accuracy			0.99	6,000,000	Accuracy			0.99	4,000,000
Macro avg	0.79	0.74	0.76	6,000,000	Macro avg	0.79	0.73	0.74	4,000,000
Weighted avg	0.99	0.99	0.99	6,000,000	Weighted avg	0.99	0.99	0.99	4,000,000

correlation technique that is architecture-based on stacking for both classical algorithms and deep neural networks.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Data availability The data source utilized in this study is obtainable from: Sarhan, M., Layeghy, S. & Portmann, M. Towards a Standard Feature Set for Network Intrusion Detection System Datasets. Mobile Netw Appl (2021). https://doi.org/10.1007/s11036-021-01843-0. Data repositories: https://www.kaggle.com/datasets/aryashah2k/nfuq nidsv2-network-intrusion-detection-dataset.

References

1. Johnson A, Dempsey K, Ross R, Gupta S, Bailey D (2011) Guide for security-focused configuration management of information systems. Computer Security Division, Information Technology

Laboratory (National Institute of Standards and Technology), NIST Special Publication 800-128
2. Internet Crime Report 2021 (2021) Federal Bureau of Investigation (FBI), Internet Crime Complaint Center, IC3
3. Abdou H, Khalifa W, Roushdy M, Salem A (2019) Machine learning techniques for credit card fraud detection. Future Comput Inform J 4(2)
4. White Paper, Cisco Annual Internet Report (2018–2023), Cisco Public, March 9. 2020. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. Accessed 01 Dec 2022
5. Scarfone K, Mell P (2007) Guide to intrusion detection and prevention systems—IDPS. Computer Security Division, Information Technology Laboratory (National Institute of Standards and Technology), NIST Special Publication 800-94
6. (2013) Information technology—security techniques—code of practice for information security controls, 2nd edition. International Standard ISO/IEC 27002, pp 67–71
7. (2015) Information technology—security techniques—selection, deployment and operations of intrusion detection systems (IDPS), 1st edn. International Standard ISO/IEC 27039
8. Edgescan (2022) vulnerability statistics report. Smart Vulnerability Management
9. Saikushwanth V, Ramachandra G (2021) Intrusion detection system using machine learning, vol 2, issue 10. Department of Computer Science and Engineering, Chalapathi Institute of Engineering and Technology, Guntur, United International Journal for Research and Technology (UIJRT). ISSN:2582-6832
10. Bhatia V, Choudhary S, Ramkumar KR (2020) A comparative study on various intrusion detection techniques using machine learning and neural network. Department of Computer Science and Engineering, Chitkara University Institute of Engineering and Technology, Chitkara University, Punjab India, 2020 8th international conference on reliability, infocom technologies and optimization (trends and future directions) (ICRITO), Amity University, Noida, India. IEEE. 978-1-7281-7016-9/20/$31.00 ©2020
11. Meftah S, et al. (2019) Network based intrusion detection using the UNSW-NB15 dataset. Int J Comput Digit Syst. https://doi.org/10.12785/ijcds/080505. ISSN (2210-142X)
12. Alamiedy T et al (2020) Anomaly-based intrusion detection system using multi-objective grey wolf optimisation algorithm. J Ambient Intell Hum Comput 11:3735–3756. https://doi.org/10.1007/s12652-019-01569-8
13. Devan P, Khare N (2020) An efficient XGBoost–DNN-based classification model for network intrusion detection system. Neural Comput Appl 32:12499–12514
14. Wei W, et al. (2020) An improved multi-objective immune algorithm for intrusion feature selection in intrusion detecion. Appl Soft Comput 95:106522. https://www.sciencedirect.com/science/article/pii/S1568494620304610
15. Zhang X, Niyaz Q, Jahan F, Sun W (2020) Early detection of host-based intrusions in linux environment. 978-1-7281-5317-9/20/$31.00 ©2020 IEEE
16. Masser ZK, et al. (2021) Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset. IEEE Access, Open Access J. https://doi.org/10.1109/ACCESS.2021.3056614
17. Anwer M, et al (2021) Attack detection in IoT using machine learning. Eng, Technol Appl Sci Res 11(3):7273–7278. https://www.etasr.com
18. Al-Bakaa A, Al-Musawi B (2021) Improving the performance of intrusion detection system through finding the most effective features. Department of Electronics and Communication Faculty of Engineering, University of Kufa, Iraq. 978-1-6654-1224-7/21/$31.00 ©2021 IEEE
19. Singh KP, Kesswani N (2022) An anomaly-based intrusion detection system for IoT networks using trust factor. SN Comput Sci 3:168. https://doi.org/10.1007/s42979-022-01053-9
20. Ambavkar O, et al. (2022) Review on IDS based on ML algorithms. Int J Res Appl Sci Eng Technol (IJRASET). IC Value: 45.98. https://doi.org/10.22214/ijraset.2022.47284. ISSN: 2321-9653
21. Anderson JP (1980) Computer security threat monitoring and surveillance. Tech. Rep., James P. Anderson Co., Washington
22. Denning DE (1987) An intrusion-detection model. IEEE Transact Softw Eng 13(2)
23. Ozkan-Okay M, et al. (2021) A comprehensive systematic literature review on intrusion detection systems. Department of Computer Engineering, Ankara University Turkey, IEEE Access. Digital Object Identifier. https://doi.org/10.1109/ACCESS.2021.3129336
24. Lai C, et al. (2021) Review of intrusion detection methods and tools for distributed energy resources. Sandia National Laboratories, Sandia Report, SAND2021-1737, pp 15–24
25. Khraisat A et al (2019) Cybersecurity, "Survey of intrusion detection systems: techniques, datasets and challenges. Internet Commerce Security Laboratory, Federation University Australia, Mount Helen, Australia. Springer Open Access. https://doi.org/10.1186/s42400-019-0038-7
26. Sharma M, Sawant K (2021) Intrusion detection system Using Deep Learning. Int J Sci Res Eng Trends (IJSRET) 7(2): 2395–566x. ISSN (Online)
27. Malek Z, Trivedi B, Shah A (2020) User behavior pattern-signature based intrusion detection. In: 2020 4th world conference on smart trends in systems, security and sustainability (WorldS4)
28. Likhomanov D, Poliukh V (2020) Predicting malicious hosts by blacklisted IPv4 address density estimation. In: The 11th IEEE international conference on dependable systems, services and technologies, DESSERT 2020, Kyiv, Ukraine
29. Paullada A, et al. (2021) Data and its (dis)contents: a survey of dataset development and use in machine learning research. Department of Linguistics, University of Washington, Seattle, WA, USA. https://doi.org/10.1016/j.patter.2021.10033
30. Sydorenko I (2021) [Online]. https://labelyourdata.com/articles/what-is-dataset-in-machine-learning. Accessed 20 Dec 2022
31. The University of Queensland in Australia. https://staff.itee.uq.edu.au/marius/NIDS_datasets/#RA5
32. Sarhan M, Layeghy S, Portmann M (2022) Towards a Standards Feature Set for Network Intrusion Detection System Datasets, University of Queensland, Brisbane QLD 4072, Australia. Mob Netw Appl 27:357–370. https://doi.org/10.1007/s11036-021-01843-0
33. What it is machine learning [online]. https://www.ibm.com/topics/machine-learning. Accessed 10 Dec 2022
34. What it is machine learning [online]. https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/. Accessed 10 Dec 2022
35. What it is machine learning [online]. https://www.oracle.com/ng/artificial-intelligence/machine-learning/what-is-machine-learning/. Accessed 10 Dec 2022
36. Janiesch C, Zschech P, Heinrich K (2021) Machine learning and deep learning. Electron Mark 31:685–695. https://doi.org/10.1007/s12525-021-00475-2
37. Pasupa K, Sunhem W (2016) A compariosn between shallow and deep architecture classifiers on small dataset. In: 8th international conference on information technology and electrical engineering (ICITEE), Yogyakarta, Indonesia. 978-1-5090-4139-8/16/$31.00 ©2016 IEEE