Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.039255
Article

A Novel Ensemble Learning System for Cyberattack Classification

Óscar Mogollón-Gutiérrez*, José Carlos Sancho Núñez, Mar Ávila Vegas and Andrés Caro Lindo

Department of Computer and Telematic Systems Engineering, Universidad de Extremadura, School of Technology, Cáceres, 10005, Spain

*Corresponding Author: Óscar Mogollón-Gutiérrez. Email: oscarmg@unex.es
Received: 18 January 2023; Accepted: 13 April 2023; Published: 23 June 2023

Abstract: Nowadays, IT systems rely mainly on artificial intelligence (AI) algorithms to process data. AI is generally used to extract knowledge from stored information and, depending on the nature of data, it may be necessary to apply different AI algorithms. In this article, a novel perspective on the use of AI to ensure the cybersecurity through the study of network traffic is presented. This is done through the construction of a two-stage cyberattack classification ensemble model addressing class imbalance following a one-vs-rest (OvR) approach. With the growing trend of cyberattacks, it is essential to implement techniques that ensure legitimate access to information. To address this issue, this work proposes a network traffic classification system for different categories based on several AI techniques. In the first task, binary models are generated to clearly differentiate each type of traffic from the rest. With binary models generated, an ensemble model is developed in two phases, which allows the separation of legitimate and illegitimate traffic (phase 1) while also identifying the type of illegitimate traffic (phase 2). In this way, the proposed system allows a complete multiclass classification of network traffic. The estimation of global performance is done using a modern dataset (UNSW-NB15), evaluated using two approaches and compared with other state-of-art works. Our proposal, based on the construction of a two-step model, reaches an F1 of 0.912 for the first level of binary classification and 0.7754 for the multiclass classification. These results show that the proposed system outperforms other state-of-the-art approaches (+0.75% and +3.54% for binary and multiclass classification, respectively) in terms of F1, as demonstrated through comparison together with other relevant classification metrics.

Keywords: Intrusion detection; ensemble learning; two-phase model; UNSW-NB15; cybersecurity

1 Introduction

The volume of data handled by information systems (IS) is growing exponentially every year [1]. There is an increasing amount of information collected by organizations, and each of them use it in following their business plans. Generally, IS, depending on their main action line, may provide access

to stored data or enable them to improve their internal processes. For this reason, it is undeniable that the nature of the data that can be managed by an IS is heterogeneous and depends on the sector where it is implemented: education [2], financial [3], food [4], health [5] or computer vision [6].

Given the enormous amount of data handled by IS, it is necessary to use mechanisms for processing large volumes of data [7]. Artificial Intelligence (AI) can play a key role in this process. According to [8], AI is the ability given to a system to perform cognitive tasks, such as learning or decision making. The use of these technologies on traditional IS makes it possible to provided them with "intelligence" and offer new ways of interacting with users. AI applications in IS are vast [9–11].

Regardless of the nature of store data in IS, AI can also be applied to guarantee or increase security by implementing intrusion detection systems (IDS). Intrusions into IS are aimed at exploiting vulnerabilities to gain access to them and, subsequently, make fraudulent use of systems. These systems help to identify malicious actions or variations through network traffic monitoring and classification which is generated by clients attempting to access these systems.

In recent years, models that build IDS with different perspectives have emerged in the scientific community. Intrusion tasks may be performed by profiling the system's behaviour to be protected and notifying detected outliers [12]. Another approach consists in analyzing incoming and outcoming traffic through a network using AI (machine learning [13,14] or deep learning [15,16]). Depending on the number of labeled network trace samples used to train an IDS, model generation can be carried out in two different ways: supervised or unsupervised. Supervised learning consists of building models from a labeled dataset. On the other hand, unsupervised learning can generate a behavioral pattern from an unlabeled dataset. Federated learning applications are an emerging trend addressing security from an unsupervised perspective [17]. By combining these two approaches, semi-supervised learning can leverage the benefits of both and improve the accuracy and generalization of intrusion detection models [18]. This approach is particularly relevant in security domains where labeled data is scarce.

Despite their technical differences, all approaches should deal appropriately with the inherent complexity of network data. Class imbalance is a prevalent problem in the field of network traffic intrusion detection, where one class of network traffic (normal traffic) is often much more prevalent than the other class (intrusive traffic). This imbalance can result in a biased classifier that consistently predicts normal traffic, even in the presence of intrusive traffic samples. This can lead to a high rate of false negatives, which, in the context of cybersecurity, can be severe, as intrusions can stay undetected for long periods and result in data theft, system compromise, and other security threats.

In terms of AI, a class imbalance issue arises when the classifier is trained on a dataset where the majority class dominates, resulting in a biased classifier that often predicts the majority class. In binary classification, the classifier is trained to distinguish between two classes, typically normal and abnormal. In this scenario, the class imbalance can be addressed by oversampling the minority class, undersampling the majority class, or combining both. However, in multiclass classification, where the classifier is trained to distinguish between multiple classes, the class imbalance problem becomes more complicated. If the majority class is oversampled or the minority class is undersampled, it could result in a distorted distribution across all classes. Additionally, the imbalanced class distribution would affect the model's performance and accuracy in recognizing minority classes. Therefore, it is crucial to address the class imbalance issue in multiclass intrusion detection, as it can significantly impact the classifier's performance on detecting intrusions correctly.

Considering the aforementioned issues, this paper proposes an intelligent system cyberattack classification. It consists of a two-phase ensemble classification model using a variety of machine learning techniques able to classify intrusions among preset categories [19–22]. Since the distribution of

different types of network traffic is uneven and the number of samples for some traffic is much greater than for the rest (class imbalance), this solution is chosen. The implementation of a two-phase model provides two main benefits: reducing the imbalance problem [23] and making intrusion detection easier [24–26]. Thus, a first level of classification distinguishes only between legitimate and attack traffic. A second classification process is responsible for classifying the type of attack detected, following an ensemble scheme. The models that make up the final model are built from a balanced one-vs-rest (OvR) strategy. Specifically, a separate binary classifier is trained for each class, with the class being considered as the positive class and all the other classes combined as a single negative class. To carry out the training of the proposed system, the labeled dataset UNSW-NB15 is used. This dataset is widely known in the scientific literature [27–29] and includes 10 different categories of network traffic. This paper demonstrates the effectiveness of this strategy which has not been implemented in modern cybersecurity solutions for cyberattack detection and classification.

Thus, the novelty of this study lies in the construction of a two-stage cyberattack classification ensemble model addressing class imbalance following an OvR approach. This is a new direction among current approaches to network traffic analysis since, to the best of our knowledge, there are no previous works that have considered this approach. This contribution seeks to mitigate the imbalanced learning in network traffic analysis and classification, which is a significant issue in this research field. Obtained results demonstrate the effectiveness of the proposal yielding the in following improvements in binary classification: (+0.7%) accuracy, (+2.92%) recall, (+0.75%) F1 and (−2.12%) FPR. In multiclass classification following improvements are achieved: (+0.22%) accuracy, (+7.04%) recall, (+3.54 %) F1.

The main contributions of the proposed work are as follows:

- A balanced binary dataset generation using an OvR scheme using Synthetic Minority Oversampling Technique (SMOTE) and Random Undersampling algorithms.
- An implementation of a two-phase ensemble model for intrusion detection and classification is proposed by combining outputs from multiple binary models.
- An evaluation of the proposed system is conducted using a modern dataset (UNSW-NB15). Our proposal improves other state-of-art contributions and demonstrates its efficiency in terms of model generation.

The remainder of the article is organized as follows: Section 2 describes related works in the field of intrusion detection and datasets with greater impact in the scientific field. Section 3 describes UNSW-NB15, hardware and software resources, machine and deep learning algorithms and a set of evaluation metrics for estimating the performance of models. Section 4 details the methodology applied for the construction of the model, and Section 5 shows and discusses the results obtained. Finally, the conclusions of the research are presented together with possible lines of future research to improve the performance of our solution.

2 Related Works

In recent years, the number of studies related to intrusion detection systems has grown considerably. One of the main reasons for this increase in publications is directly related to the increase of smart devices. Likewise, the variety of computer attacks is growing, and as a result, there is a need to build effective intrusion detection systems. The scientific literature contains a wide variety of systems according to the nature of the system to be protected.

Some authors have proposed systems based on classical classifiers to distinguish normal behavior and attack behavior [30]. Other works perform multiclass classification, where samples that are not considered legitimate behavior are classified as a specific attack type [31]. Khammasi et al. [32], after feature selection based on genetic algorithms, applies decision trees for intrusion detection. On the other hand, some research, such as Bagui et al. [33] focuses on applying deep learning models based on artificial neural networks. A convolutional neural network is used by Al-Turaiki et al. [34] for the implementation of both a binary and a multiclass intrusion classifier.

Other works are based on the construction of ensemble models. Specifically, in [35] a combined voting model is used where the class prediction is obtained from the probabilities of each classifier. Similarly, the paper by Maniriho et al. [36] applies a combined model. However, the ensemble strategy applied in this research consists of randomly generated classifiers. In the results of their contribution, the binary classifier presents high accuracy due to the selection of a limited dataset, using 34% of the training set to test the system. Both studies show that, from the predictions obtained by the individual models, a higher attack detection capability is generally achieved.

Other approaches perform two-stage classification, where, first, a binary classifier determines whether a sample is an attack or not, and then a second process tries to identify the type of attack using classical algorithms [37]. However, the results of this study do not consider all the categories present in the original dataset when evaluating their proposal. Following a two-stage model, Khan et al. [38] proposes a system applying deep learning at the second level. In this work, an oversampling of only legitimate traffic is carried out assuming a considerable increase in the number of samples in this category and influencing the results obtained.

Regarding the availability of datasets that allow easy comparison of the accuracy and precision of the proposed algorithms, datasets that collect the behavior of simulated environments under different attacks are used in the scientific literature, e.g., KDD99 [39], NSL-KDD [40] or UNSW-NB15 [19]. The datasets prior to UNSW-NB15 had several limitations that affected the effectiveness of an IDS in a real environment, and they were not designed with an IOT perspective in mind. The KDD99 dataset, developed in the late 1990s, lacks many of the attacks that are present today [41]. Another example of a deficient set is NSL-KDD. It was released in 2009, and its main drawback is the presence of multiple repeated packets that affect the performance of the algorithms [42].

The classification of attacks collected in the described datasets is handled differently depending on the dataset used. For example, for KDD99, feature selection based on multidimensional feature fusion together with stacking ensemble learning is applied in [43]. Alternatively, a logistic regression-based model is built from a dataset with feature selection based on genetic algorithms, as in [27]. Another research employs NSL-KDD [36]. In [44], the main contribution is a novel correlation analysis of the dataset features for the selection of the most important features. Regarding the most recent ensemble, UNSW-NB15, AI techniques applied are very varied, as demonstrated by Ponmalar's study in [45] where an SVM-based ensemble is combined with the Chaos Game Optimization algorithm.

Regarding the UNSW-NB15 dataset, it is worth mentioning that the authors have published a reduced version of the original available in [46] and it is widely used in research [34,47], as it tries to improve the balance between traffic categories, reducing the number of legitimate traffic samples.

As preliminary conclusions from the review of the state of the art, the UNSW-NB15 dataset can be considered the most complete. Due to the wide variety of attacks collected, this dataset is highly suitable. For the construction of an IDS capable of anticipating cyber threats in IS.

3 Materials

This paper proposes a system to identify and classify network traffic from the UNSW-NB15 dataset combining outputs from binary models trained with several well-known classification algorithms. This section presents a description of the dataset, including classes distribution, classification algorithms used in the experiments, and a set of evaluation metrics that enable to evaluating the proposed system in the Experimental Design section.

3.1 Dataset

The UNSW-NB15 dataset was created by the Cyber Range Lab at the University of New South Wales, Canberra (UNSW) with the goal of simulate a heterogeneous environment of legitimate and real attack traffic [19]. The original dataset consists of 2540044 samples and exhibits a high-class imbalance. One of the categories, Normal, represents more than 87% of the total samples. However, as previously indicated, the authors have published a new dataset with fewer rows and columns while maintaining the complexity [46]. This latter dataset is already split into training and testing sets. The experiments in this research have been performed with the updated dataset. This version of the UNSW-NB15 contains a total of 44 characteristics, 40 being numeric and 4 being categorical (proto, service, state, and attack cat). Table 1 shows the distribution of classes for the training and test sets.

Table 1: Train and test sets distribution in UNSW-NB15

	Train	Train (%)	Test	Test (%)
Analysis	2000	1.14	677	0.82
Backdoor	1746	1.00	583	0.71
DoS	12264	6.99	4089	4.97
Exploits	33393	19.04	11132	13.52
Fuzzers	18184	10.37	6062	7.36
Generic	40000	22.81	18871	22.92
Normal	56000	31.94	37000	44.94
Reconnaissance	10491	5.98	3496	4.25
Shellcode	1133	0.65	378	0.46
Worms	130	0.07	44	0.05
TOTAL	**175341**	**100%**	**82332**	**100%**

3.2 Algorithms

For the construction of the ensemble classifier proposed in this paper, machine learning and deep learning algorithms that obtain the best results both in our previous experiments and in the scientific literature are combined [48,49].

- K Nearest Neighbors Classifier (KNN). KNN algorithm is an instance-based learning algorithm. This means that the model classification process is based on knowledge gained during the training phase [50]. The classification of a sample consists of applying an algorithm that calculates the distance of that sample from the rest. The class to which it belongs is decided by applying a majority voting algorithm between the nearest k observations.

- Support Vector Machine (SVM). The fundamental idea of the SVM algorithm is to find the hyperplane that best divides the dataset into a given number of classes or categories [51].
- Decision Trees (DT). DT-based models are classifiers capable of predicting the category of a sample by applying simple decision rules learned in the training phase [52].
- Multilayer Perceptron (MLP). MLP is a simple type of artificial neural network composed of an input layer, an output layer, and multiple intermediate layers, all fully interconnected.

The experiments have been carried out with Python, version 3.8. Pandas, Numpy, Scikit-Learn and Imbalanced-learn libraries were used; these libraries are frequently used in this field of research [33,53]. The selection of the best value for the hyperparameters of each algorithm was carried out using grid search and cross-validation techniques [28,54]. This hyperparameter tuning approach allows testing multiple combinations of values after having previously defined a search space for each of them. Regarding the hardware, a computer with an Intel Core i7-9750H @4.50 GHz with 16 GB of RAM was used.

3.3 Evaluation Metrics

In this study, several metrics are chosen to evaluate the performance of intrusion detection models. The metrics considered are accuracy, precision, recall/detection rate (DR), F1-score and false alarm rate/false positive rate (FAR/FPR), which are calculated as a function of true positives (TPs), true negatives (TNs), false positives (FPs) and false negatives (FNs). In addition, the confusion matrix is also used. For each metric, Table 2 shows a concise definition and how it is calculated.

Table 2: Evaluation metrics

Metric	Definition	Calculation
Accuracy	It is the ratio between the classification hits and the total predictions.	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	The proportion between the attacks detected by the classifier that are attacks.	$\frac{TP}{TP+FP}$
Recall (or detection rate)	The ratio of attack detection to all tested attacks.	$\frac{TP}{TP+FN}$
F1-score	Estimation of precision and recall using harmonic mean.	$2 * \frac{Recall * Precision}{Recall + Precision}$
False positive rate	Measure of the proportion of false positives among all negative cases	$FP/(FP+TN)$

4 Experimental Design

Figs. 1, 2, and Algorithm 1 show the experimental design followed to define the proposed detection and classification intelligent system. The design of the proposal consists of three stages, which are described below.

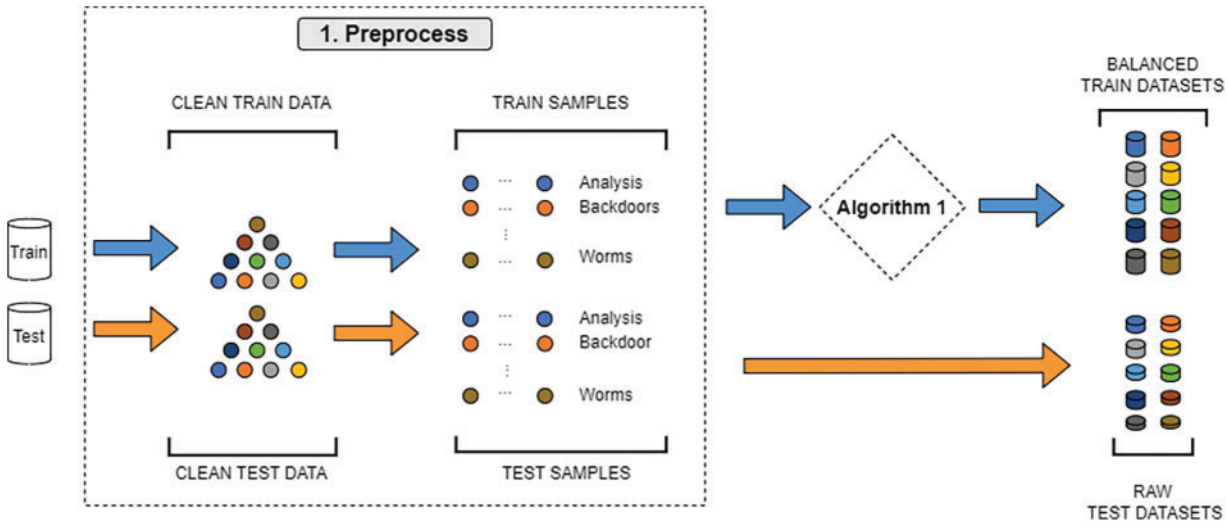

Figure 1: Data flow for balanced binary datasets generation

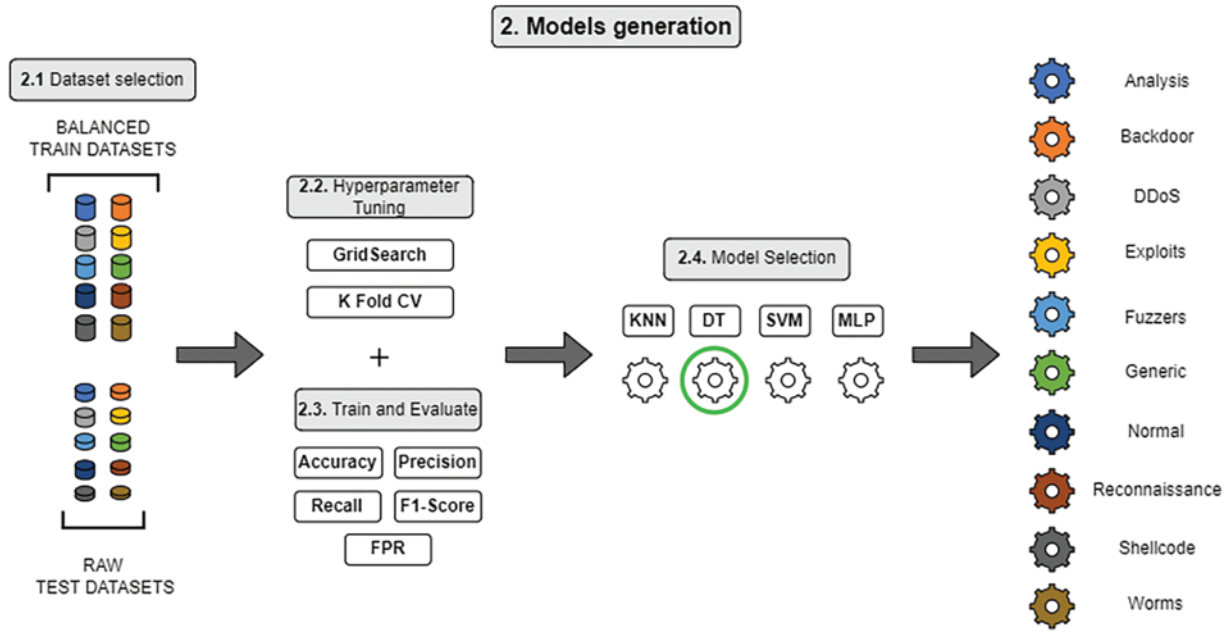

Figure 2: Binary models generation

4.1 Phase 1: Cleaning, Preprocessing and Normalization

As usual, the data must be cleaned and normalized prior to the application of classification algorithms. To this end, the numerical features have been normalized by calculating the standard score so that each feature follows a Gaussian distribution (mean zero and unit variance). On the other hand, categorical features have been encoded using numerical labels. As indicated in [20] there are no redundant tuples.

4.2 Phase 2: Classifier Models Generation

The next step is a generation of ten binary specialized classification models because there are ten different categories in the dataset, which are listed in Table 1. Therefore, each of the generated models can distinguish between samples corresponding to one type of traffic and the rest of the samples.

Obtaining each classification model requires the creation of a specific dataset. This model generation process is organized into several subphases to be carried out for each type of traffic collected in UNSW-NB15 as shown in Fig. 2:

- Phase 2.1. Dataset selection for training and testing binary models.
- Phase 2.2. Hyperparameter configuration using grid search together with cross-validation for the algorithms mentioned in the Algorithms subsection (KNN, SVM, DT and MLP).
- Phase 2.3. Model training and evaluation for each combination of hyperparameters.
- Phase 2.4. Best model selection based on the evaluation metrics.

In Phase 2.1, training and test sets are selected for each binary classification model. The training sets used for the generation of binary models are generated in a balanced way, so half of each dataset consists of samples belonging to one category, and the other half contains traffic for the rest of the categories. The number of samples in the second half remains the same for each type of traffic. For example, according to Table 1, the Analysis category consists of 2000 samples, so the resulting training set consists of 3998 tuples: 2000 of type Analysis and 222 of the other 9 existing traffic types. Algorithm 1 generates balanced training sets following this approach. The result of this process for Analysis traffic is shown in Fig. 3. The same procedure is applied to the remaining categories.

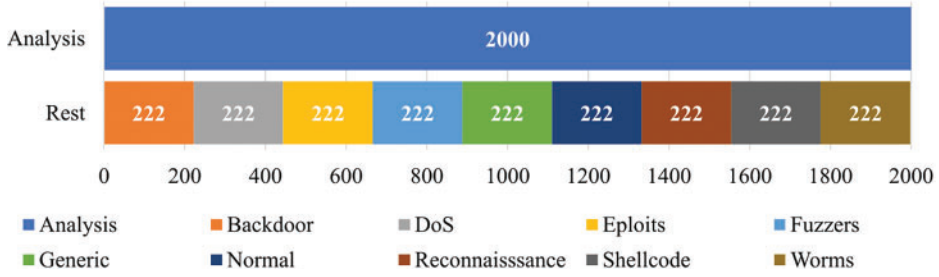

Figure 3: Detailed train dataset distribution for analysis model generation

In Algorithm 1, resampling techniques are applied when the number of samples in a category does not reach or exceed a certain threshold (n_samples). This threshold varies according to the model to be generated and allows training models using balanced datasets. On the one hand, SMOTE is an oversampling algorithm that allows the generation of synthetic samples (without repetition) from a dataset [55] and, thus, to reach the calculated number of samples. On the other hand, random undersampling is used to select the number of samples to be removed from a class that exceeds the threshold. Due to the random nature of the latter method, the evaluation of the binary models is calculated as the average of the evaluation metrics obtained in 10 experiments to increase the variability of the undersampled classes.

In Phase 2.2, for hyperparameter tuning, grid search and cross-validation parameters are configured. Table 3 shows the hyperparameter search space for each of the applied algorithms. Next, in Phase 2.3, a model is trained with each combination of parameters. Once the hyperparameters have been tuned, each generated binary model is evaluated with a specific test set. It consists of samples of the target traffic (positive class) and the rest (negative class) from the original test set listed in Table 1.

Note that the full original test set is later used to evaluate the proposed system. Both Phase 2.2. and Phase 2.3. should be applied for each of the four algorithms studied.

Table 3: Hyperparameter search space per algorithm

Metric	Hyperparameters
KNN	n neighbors = {1 ...30}
	weights = {uniform, distance}
	algorithm = {auto, ball tree, kd tree}
SVM	C = {1 ...1000}
	gamma = {1x10−5 ...1}
DT	criterion = {gini, entropy}
	max depth = {1...35}
MLP	activation = {relu, tanh}
	learning rate init = {1x10−5 ...1}
	max iter = {20, 40, 60, 80, 100, 120, 140, 160, 180, 200}
	batch size = {32, 64, 128, 256, auto}

Once the four models have been generated (one for each algorithm) for each type of traffic collected, in Phase 2.4, a criterion is established to select the binary model with the best performance, considering the evaluation metrics obtained in the previous phase. In network traffic classification, a model that simply predicts the majority class for all traffic samples will have a high accuracy, but it may not be a good model if the minority class is of interest. To address this issue, F1-score is selected. The rationale behind choosing this metric is that it balances the trade-off between precision and recall. The selected model is used to build the final proposal.

Algorithm 1: Balanced train set generation for binary models

Input Preprocessed DATASET
Output Train sets for each category

categories ← *list of categories from DATASET*
n_categories ← *number of categories from DATASET*
rest_categories ← ∅
for *category in categories* **do**
    // threshold of samples for the rest of categories
    n_samples ← *int (number of records from category / n_categories − 1)*
    rest_categories ← *categories* ∩ *category*
    for *other_category* **in** *rest_categories* **do**
        n_samples_c ← *number samples from another category*
        if *n_samples_c* ≤ *n_samples* **then**
            OVERSAMPLING
        else
            UNDERSAMPLING
    end for
end for

4.3 Phase 3: Construction of the Final Model

In Phase 3, with the binary models already generated, and after evaluating the individual performance through the evaluation metrics, two alternatives are proposed for the construction of the final model.

The first approach consists of calculating the weighted average of the metrics calculated from the binary models selected after Phase 2. This calculation causes the model performance to be influenced by the initial distribution of the classes. Each weighted metric is calculated according to Eq. (1).

$$metric_{weighted} = \frac{\sum metric * n\_test\_samples_{category}}{n\_test\_samples_{total}} \tag{1}$$

The results obtained with this approach should be interpreted as the average ability of the model to determine whether a sample belongs to a class or not. However, in a real environment, the same sample can be classified into different categories by different classifiers. For this reason, in the experimental design, it is necessary to implement an alternative that allows measuring the degree of certainty of each classifier.

In this sense, the second approach is based on the construction of a two-stage classification model, organized into two phases, as shown in Fig. 4. The first one is responsible for identifying the behavior as Normal or as an Attack, using the binary model trained for this purpose. In a second level of detection, the possible threat is classified as one of the nine possible attacks collected in UNSW-NB15.

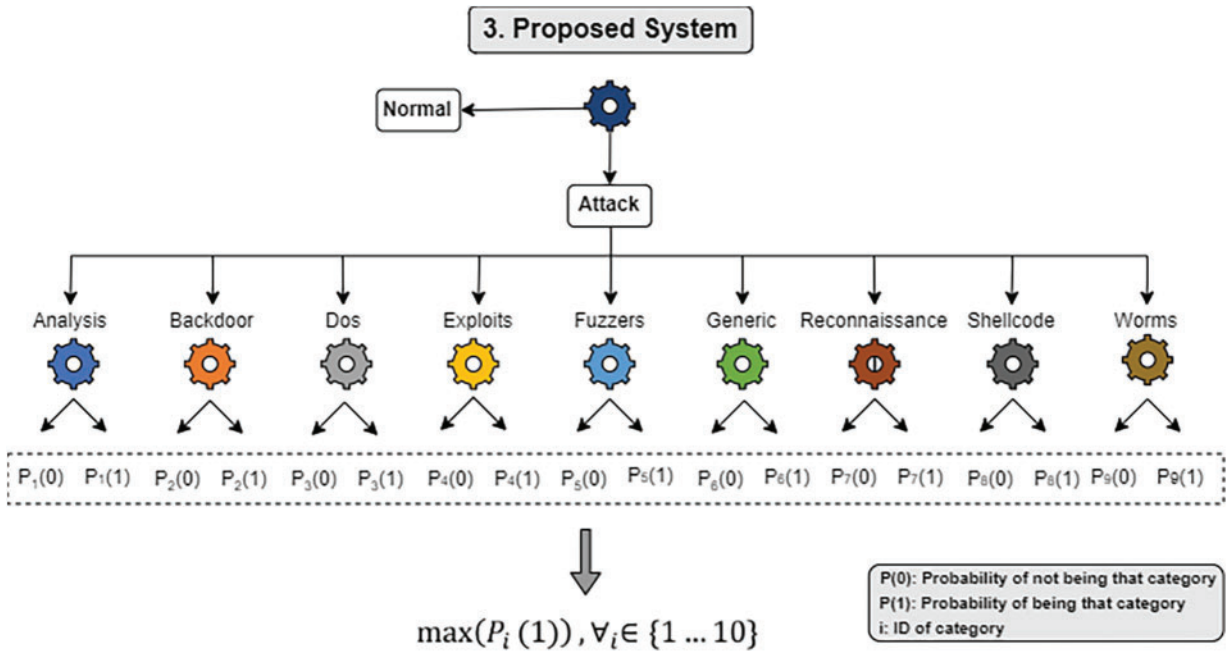

Figure 4: Two-stage classification ensemble

This classification task is performed using an ensemble model composed of the binary classifiers of the nine attack types. Each classifier outputs the probability of belonging to its class and the rest. Taking into consideration the former probability, the class with the highest value is taken as the final prediction. Finally, a calculation of the evaluation metrics is made from the resulting confusion matrix.

5 Results and Discussion

This section presents the results obtained after the experiments performed for the traffic classification into different categories (normal or 9 different types of attacks) using the system based on machine learning and deep learning proposed in this contribution. First, the evaluations of the first set of experiments, based on the construction of multiple binary classifiers, are presented. After selecting the best model for each category, configurations are provided. Second, the performance of the final system for both approaches described in Phase 3 subsection is discussed and compared with some relevant research works in this field. Finally, a study in terms of computation time is conducted. This is an important aspect because it can help greatly in deciding which model to use depending on the circumstances and usage needs.

Table 4 lists the accuracy, precision, recall and F1 metrics for the expert classifiers obtained at the end of Phase 2. The traffic that is best detected from the rest is Generic, reaching a value higher than 0.98 for all the calculated metrics. This remarkable performance may be influenced by the high number of samples used to generate the model compared to the rest. The detection (or recall) capability for Reconnaissance and Shellcode attacks exceeds 0.9. In the case of attack detection, regardless of the category, the proposed classifier reaches a value slightly above 0.91. The attacks with the lowest accuracy/F1 are DoS, Exploits and Fuzzers, with values of approximately 0.8. Nevertheless, the results obtained by the system for traffic detection and classification have a good accuracy, above 0.85 in most of the categories, reaching 0.9 and achieving very good values in some cases.

Table 4: Evaluation of binary models grouped by traffic and algorithm

Category	Algorithm	Accuracy	Precision	Recall	F1
Analysis	KNN	0.7918	0.7986	0.7918	0.7901
	SVM	0.8017	0.8501	0.8017	0.7934
	DT	**0.8312**	**0.8388**	**0.8312**	**0.8298**
	MLP	0.8244	0.8330	0.8244	0.8228
Backdoor	KNN	0.7925	0.8092	0.7925	0.7891
	SVM	0.8047	0.8154	0.8047	0.8026
	DT	**0.8553**	**0.8747**	**0.8553**	**0.8530**
	MLP	0.7908	0.7927	0.7908	0.7902
DoS	KNN	0.7015	0.7011	0.7015	0.7007
	SVM	0.7740	0.7761	0.7740	0.7723
	DT	0.7507	0.7602	0.7507	0.7457
	MLP	**0.7907**	**0.7936**	**0.7907**	**0.7909**
Exploits	KNN	0.7398	0.7464	0.7398	0.7414
	SVM	0.7697	0.7702	0.7697	0.7699
	DT	**0.8041**	**0.8170**	**0.8041**	**0.7962**
	MLP	0.7983	0.7975	0.7983	0.7963
Fuzzers	KNN	0.8000	0.8063	0.8000	0.8004
	SVM	**0.8127**	**0.8247**	**0.8127**	**0.8128**
	DT	0.7970	0.8133	0.7970	0.7967
	MLP	0.8005	0.8002	0.8005	0.8002

(Continued)

Table 4: Continued

Category	Algorithm	Accuracy	Precision	Recall	F1
Generic	KNN	0.9780	0.9789	0.9780	0.9781
	SVM	0.9791	0.9798	0.9791	0.9792
	DT	**0.9835**	**0.9839**	**0.9835**	**0.9835**
	MLP	0.9812	0.9817	0.9812	0.9813
Normal	KNN	0.7964	0.8396	0.7964	0.7999
	SVM	0.8276	0.8469	0.8276	0.8304
	DT	**0.9112**	**0.9112**	**0.9112**	**0.9112**
	MLP	0.8696	0.8825	0.8696	0.8714
Reconnaissance	KNN	0.8378	0.8377	0.8378	0.8377
	SVM	0.8844	0.8855	0.8844	0.8845
	DT	**0.9183**	**0.9255**	**0.9183**	**0.9183**
	MLP	0.9136	0.9206	0.9136	0.9136
Shellcode	KNN	0.8955	0.9054	0.8955	0.8949
	SVM	0.9299	0.9355	0.9299	0.9297
	DT	**0.9643**	**0.9644**	**0.9643**	**0.9643**
	MLP	0.9325	0.9355	0.9325	0.9324
Worms	KNN	0.8764	0.8766	0.8764	0.8764
	SVM	0.8764	0.8766	0.8764	0.8764
	DT	**0.8876**	**0.8885**	**0.8876**	**0.8876**
	MLP	0.8876	0.8884	0.8876	0.8876

Another aspect to consider, looking closely at Table 4, is that the algorithm that has shown the best performance is the decision tree, in addition to always being the most efficient in terms of computation time, as shown in Table 5.

Table 5: Selected model for each traffic category

Category	Acc.	Prec.	Recall	F1	FPR	Train Time	Parameters
Analysis	0.8312	0.8388	0.8312	0.8298	0.1723	105 ms	criterion = 'entropy' max depth = 8
Backdoor	0.8553	0.8747	0.8553	0.8530	0.1502	98 ms	criterion = 'entropy' max depth = 9
DoS	0.7907	0.7936	0.7907	0.7909	0.2076	10.6 s	hidden layer sizes = (12) max iter = 190 activation = 'relu' solver = 'adam'
Exploits	0.7983	0.7975	0.7983	0.7963	0.2157	692 ms	criterion = 'gini' max depth = 10

(Continued)

Table 5: Continued

Category	Acc.	Prec.	Recall	F1	FPR	Train Time	Parameters
Fuzzers	0.8127	0.8247	0.8127	0.8128	0.1823	47.4 s	C = 20 kernel = 'rbf' degree = 3 probability = True
Generic	0.9835	0.9839	0.9835	0.9835	0.0144	2 s	criterion = 'gini' max depth = 10
Normal	0.9112	0.9112	0.9112	0.9112	0.0952	2.4 s	criterion = 'entropy' max depth = 4
Reconn.	0.9183	0.9255	0.9183	0.9183	0.0785	170 ms	criterion = 'gini' max depth = 7
Shellcode	0.9643	0.9644	0.9643	0.9643	0.0357	88 ms	criterion = 'entropy' max depth = 14
Worms	0.8876	0.8884	0.8876	0.8876	0.1126	58 m	criterion = 'gini' max depth = 12

To ensure that any researcher can examine the values selected after the optimization of hyper-parameters using grid search and cross-validation, Table 5 shows the best results obtained for each classifier. The Scikit-Learn implementation of the decision tree and multilayer perceptron algorithms includes a parameter, random state, which allows configuring the seed for the generation of pseudo-random values and obtaining the same results in each execution. In all experiments, this parameter has taken the value 42, a frequent value for the initialization of AI algorithms. In addition, the FPR and the training time of each of them have been calculated since they are metrics of special interest in the field of intrusion detection systems. It can be seen that the expert classifier aimed at detecting any anomalous traffic (the row with the Normal category value) obtains an FPR close to 0.09.

Table 6 contains the confusion matrix obtained from the two-phase model of the binary classifiers. Considering the main diagonal of the matrix, the model performs well for the categories Exploits, Generic, Normal and Reconnaissance, with a high number of correctly classified samples. In an intrusion detection system, locating an attack is fundamental. Therefore, it is important to highlight the number of samples correctly classified as Normal, 34381, out of a total of 37000. Regarding the second level of classification devoted to classifying threats, the matrix shows how a large part of the traffic coming from attacks, such as DoS or Exploits, is classified as Analysis or Backdoor. This may be because the environment where this traffic has been generated has experienced few changes during the course of the attacks, and therefore, the samples used to generate the Analysis or Backdoor models lack useful information to distinguish these attacks from the rest.

From the results previously discussed in Table 5, it is possible to calculate how proposed solution behaves when detecting attacks, regardless of the category to which they belong. Table 7 shows the results of this classifier under the proposal called Normal Classifier. After generating the binary models in Phase 2, two approaches have been discussed to estimate the performance of the final proposed system. The first consists of calculating the average capability of the model to decide whether a sample belongs to a certain class and is shown in Table 7 as Weighted proposal. Since there are no works in the

literature that address the classification problem with this novel approach, the results are not amenable to comparison with other proposals. Finally, the Ensemble row of Table 7 shows the results of the two-step solution. These metrics can be compared with other studies, as they have been calculated from the test set described in the Method section.

Table 6: Confusion matrix for ensemble approach

	Analy.	Back.	DoS	Exploits	Fuzzers	Generic	Normal	Reconn.	Shell.	Worms
Analysis	348	232	25	36	4	0	32	0	0	0
Backdoor	353	182	7	2	3	0	31	0	3	2
DoS	1531	1074	326	540	75	16	270	42	123	92
Exploits	2175	1275	301	5237	218	53	793	188	420	472
Fuzzers	854	438	29	191	1114	20	2694	4	659	59
Generic	82	26	25	224	61	18248	79	2	107	17
Normal	133	41	223	477	828	7	34381	8	809	93
Reconn.	234	121	33	43	10	0	96	2391	358	210
Shellcode	5	0	0	1	6	0	16	0	350	0
Worms	2	0	0	11	0	2	1	1	3	24

Table 7: Overall results for proposed methods

Approach	Classification	Accuracy	Precision	Recall (DR)	F1-score	FPR (FAR)
Expert	Binary	0.9292	0.8955	0.9292	0.9120	0.0885
Weighted	Multiclass	0.8966	0.8983	0.8966	0.8963	0.1070
Ensemble	Multiclass	0.7605	0.8271	0.7604	0.7754	0.0456

Finally, it is interesting to compare the performance of the proposed system with other alternatives proposed in the scientific literature, some of them previously discussed in the Related works section. Table 8 shows the performance of the proposal compared with the other approaches, both the Normal binary model, which distinguishes legitimate and attack traffic, and the proposed 10-class classification model. In addition, it has also been verified that the exposed works use the same reduced dataset in their experiments and a full set of features, which is a key aspect to be able to show a comparison of results.

Comparing the results of the binary classifier with other related works, it can be seen that the results obtained are slightly better in accuracy, detection, F1 and FPR. Al-Turaiki's work employs a two-stage approach for feature selection and applies convolutional neural networks for attack classification. Their work improves in terms of accuracy but performs worse on the rest of the calculated metrics. In addition, convolutional neural networks take considerable time to train, with high energy and resource consumption. In the study addressed by Murovic, he seeks the creation of deep learning models minimizing the use of FPGA resources, obtaining good results. In the experiments carried out by Kasongo, the construction of a simple neural network is proposed in combination with a wrapper-based feature extraction approach. Tian used the SVM algorithm and

performed a hyperparameter adjustment with an algorithm based on the swarm of bees, obtaining a low FPR, close to the FPR proposed in this work.

Table 8: Comparison of performance with related works

Works	Classification	Year	Accuracy	Precision	Recall	F1	FPR
Proposed (Normal)	Binary	2021	0.9292	0.8955	0.9292	0.9120	0.0885
Al-Turaiki et al. [34]	Binary	2021	0.9025	0.9100	0.9000	0.9045	–
Murovic et al. [56]	Binary	2021	0.9218	–	–	–	0.1097
Kasongo et al. [31]	Binary	2020	0.8710	–	–	–	–
Tian et al. [57]	Binary	2019	0.8963	–	–	–	0.0916
Proposed (Ensemble)	Multiclass	2021	0.7605	0.8271	0.7604	0.7754	0.0456
Al-Turaiki et al. [34]	Multiclass	2021	0.6946	0.8400	0.6900	0.7400	–
Kasongo et al. [31]	Multiclass	2020	0.7583	–	–	–	–

Regarding the time complexity of the model, classification and training times have been considered. The average computation time required to classify a sample by our proposal is under 0.02 seconds. Fig. 5 graphically displays the time taken to generate/train each model. Since all the experiments were run on the same machine, the times are able to be compared. However, it should be noted that the time taken to train and test a model is influenced by a) the number of samples used to train and test the model and b) the algorithm used and its configuration. It can be seen that the Fuzzers class classifier requires a large portion of the total time since the SVM algorithm with a computational complexity of O(n3) is used [58]. The second most time-consuming algorithm to generate is the one for detecting DoS attacks, and although it is generated with a contained number of samples, the best results are obtained with a neural network. For the rest of the categories, the decision tree is the algorithm that returns the best results with a time complexity of O(mnlog2n), where m is the dimensionality of the data and n is the number of samples [59].

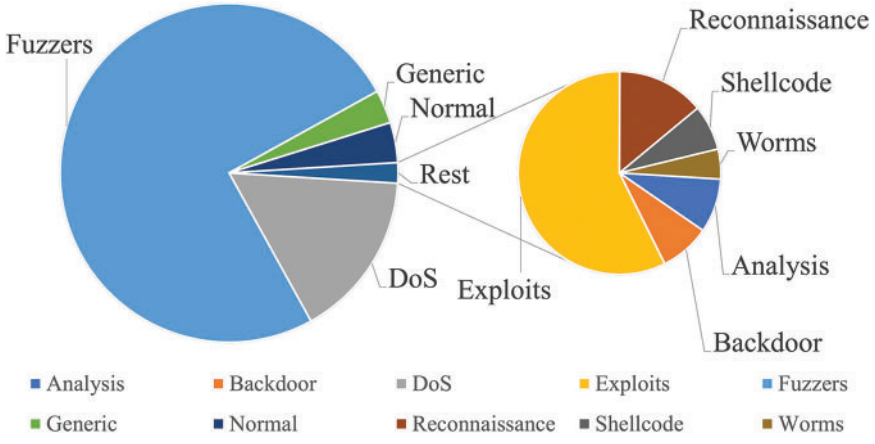

Figure 5: Model generation time per traffic

From the times obtained, it is observed that the binary model (Normal) is efficient. This aspect is of special interest in a real environment where the model should be trained periodically to improve

the overall detection of the system. Regarding training times, it is observed that the Fuzzers category requires time-consuming training in comparison to the other binary classifiers. This problem could be addressed by choosing a more efficient but less accurate model. Thus, the overall system update time (retraining of the models) would be considerably reduced when dealing with models of lower complexity.

6 Conclusions and Future Works

This paper proposes a novel method of constructing an attack detection model on the UNSW-NB15 dataset. The strength of this research lies in the development of a model for the cyber-protection of the connected environments. Our approach, based on an ensemble of binary expert models, can differentiate several types of network traffic. This approach has not been suggested by any previous study.

In binary classification (distinction between attack and legitimate traffic), our proposal obtains an accuracy, DR and F1 greater than 0.91 and an FPR of 0.085. This result exceeds several of the proposals analyzed in the state of the art. With regard to multiclass classification, our proposal easily classifies the Generic category, reaching a value above 0.98 in terms of the accuracy and F1 score, and Reconnaissance and Shellcode both exceed 0.9 for accuracy and DR. Overall, the ensemble model achieves 0.7605, 0.7604, and 0.7754 for Accuracy, DR and F1, respectively. Notably, a low FPR rate is reached with this approach, achieving a value of 0.0456.

As a result, based on our model it is possible to determine the set of attacks that an organization should be aware of in their information technology infrastructure. Analysis, Backdoor, DDoS and Worms are identified as the most difficult attacks to identify. In particular, organizations should pay attention to mitigating the effects of the first two types, as these are difficult to identify.

Finally, the proposed model generates good results even in unbalanced datasets, improving other state-of-art contributions and demonstrating its efficiency in terms of model generation.

In future work, in-depth research will be conducted to improve the proposed model's performance. One limitation of the proposed system is that it requires training a separate classifier for each class, which can be computationally expensive and time-consuming during hyperparameter tuning step, especially for datasets with many categories. To address this issue, the identification of the most difficult categories should be made prior to model generation. Another solution would be to identify what features define these attacks. To accomplish this goal, an optimal selection of features for dataset generation could be carried out. Another limitation found in the proposed two-stage model when running experiments is that if the first step is not accurate, the entire model may perform poorly. For this reason, the construction of a robust first-stage model is crucial. To accomplish this, the implementation of a deep learning model is proposed in the first step.

Funding Statement: This work was supported by the Junta de Extremadura (European Regional Development Fund), Consejería de Economía, Ciencia y Agenda Digital, under Project GR21099.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. Kemp, "Creative agency-we are social usa," 2022. [Online]. Available: https://wearesocial.com/cn/wpcontent/uploads/sites/8/2022/01DataReportal-GDR002-20220126-Digital-2022-Global-Overview-Report-Essentials-vpdf

[2] S. N. Kew and Z. Tasir, "Developing a learning analytics intervention in e-learning to enhance students' learning performance: A case study," *Education and Information Technologies*, vol. 27, no. 5, pp. 7099–7134, 2022.

[3] X. Lei, U. H. Mohamad, A. Sarlan, M. Shutaywi, Y. I. Daradkeh *et al.,* "Development of an intelligent information system for financial analysis depend on supervised machine learning algorithms," *Information Processing and Management*, vol. 59, no. 5, pp. 103036, 2022.

[4] M. Avila, M. Durán, D. Caballero, T. Antequera, T. Palacios-Pérez *et al.,* "Magnetic resonance imaging, texture analysis and regression techniques to non-destructively predict the quality characteristics of meat pieces," *Engineering Applications of Artificial Intelligence*, vol. 82, no. 1, pp. 110–125, 2019.

[5] U. Chelladurai and S. Pandian, "A novel blockchain based electronic health record automation system for healthcare," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 1, pp. 693–703, 2021.

[6] A. M. Roy, R. Bose and J. Bhaduri, "A fast accurate fine-grain object detection model based on YOLOv4 deep neural network," *Neural Computing and Applications*, vol. 34, no. 5, pp. 3895–3921, 2022.

[7] B. Xue, M. Warkentin, L. A. Mutchler and P. Balozian, "Self-efficacy in information security: A replication study," *Journal of Computer Information Systems*, vol. 63, no. 1, pp. 1–10, 2023.

[8] P. McCorduck and C. Cfe, *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence*. USA: CRC Press, 2004.

[9] E. B. Pancar and M. V. Akpınar, "Information systems and artificial intelligence technology applied in concrete road design," *Intelligent Computing and Applications*, vol. 343, pp. 559–568, 2015.

[10] V. Tolubko, V. Vyshnivskyi, V. Mukhin, H. Haidur, N. Dovzhenko *et al.,* "Method for determination of cyber threats based on machine learning for real-time information system," *International Journal of Intelligent Systems and Applications*, vol. 10, no. 8, pp. 11–18, 2018.

[11] J. C. Sancho Núñez, A. Caro Lindo, M. Ávila and A. Bravo, "New approach for threat classification and security risk estimations based on security event management," *Future Generation Computer Systems*, vol. 113, pp. 488–505, 2020.

[12] T. Dbouk, A. Mourad, H. Otrok, H. Tout and C. Talhi, "A novel ad-hoc mobile edge cloud offering security services through intelligent resource-aware offloading," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1665–1680, 2019.

[13] P. Horchulhack, E. K. Viegas and A. O. Santin, "Toward feasible machine learning model updates in network-based intrusion detection," *Computer Networks*, vol. 202, no. 4, pp. 108618, 2022.

[14] M. Bagaa, T. Taleb, J. B. Bernabe and A. Skarmeta, "A machine learning security framework for IoT systems," *IEEE Access*, vol. 8, pp. 114066–114077, 2020.

[15] D. Li, L. Deng, M. Lee and H. Wang, "IoT data feature extraction and intrusion detection system for smart cities based on deep migration learning," *International Journal of Information Management*, vol. 49, no. 6, pp. 533–545, 2019.

[16] S. Jeon and H. K. Kim, "AutoVAS: An automated vulnerability analysis system with a deep learning approach," *Computers and Security*, vol. 106, no. 4, pp. 102308, 2021.

[17] O. A. Wahab, A. Mourad, H. Otrok and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1342–1397, 2021.

[18] B. Jiang, S. Chen, B. Wang and B. Luo, "MGLNN: Semi-supervised learning via multiple graph cooperative learning neural networks," *Neural Networks*, vol. 153, no. 4, pp. 204–214, 2022.

[19] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Military Communications and Information Systems Conf., MilCIS2015*, Canberra, Australia, 2015.

[20] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Information Security Journal*, vol. 25, no. 1–3, pp. 18–31, 2016.
[21] M. AL-Hawawreh, N. Moustafa and E. Sitnikova, "Identification of malicious activities in industrial internet of things based on deep learning models," *Journal of Information Security and Applications*, vol. 41, no. 5, pp. 1–11, 2018.
[22] N. Moustafa, B. Turnbull and K. K. R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4815–4830, 2019.
[23] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2012.
[24] B. A. Tama, M. Comuzzi and K. H. Rhee, "TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system," *IEEE Access*, vol. 7, pp. 94497–94507, 2019.
[25] F. A. Khan, A. Gumaei, A. Derhab and A. Hussain, "TSDL: A two-stage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30373–30385, 2019.
[26] C. A. de Souza, C. B. Westphall, R. B. Machado, J. B. M. Sobral and G. dos Santos Vieira, "Hybrid approach to intrusion detection in fog-based IoT environments," *Computer Networks*, vol. 180, no. 7, pp. 107417, 2020.
[27] H. Zhang, L. Huang, C. Q. Wu and Z. Li, "An effective convolutional neural network based on SMOTE and Gaussian mixture model for intrusion detection in imbalanced dataset," *Computer Networks*, vol. 177, no. April, pp. 107315, 2020.
[28] M. M. Baig, M. M. Awais and E. S. M. El-Alfy, "A multiclass cascade of artificial neural network for network intrusion detection," *Journal of Intelligent and Fuzzy Systems*, vol. 32, no. 4, pp. 2875–2883, 2017.
[29] A. Verma and V. Ranga, "Machine learning based intrusion detection systems for IoT applications," *Wireless Personal Communications*, vol. 111, no. 4, pp. 2287–2310, 2020.
[30] M. N. Aziz and T. Ahmad, "Clustering under-sampling data for improving the performance of intrusion detection system," *Journal of Engineering Science and Technology*, vol. 16, no. 2, pp. 1342–1355, 2021.
[31] S. M. Kasongo and Y. Sun, "A deep learning method with wrapper based feature extraction for wireless intrusion detection system," *Computers and Security*, vol. 92, no. 1, pp. 101752, 2020.
[32] C. Khammassi and S. Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," *Computers and Security*, vol. 70, no. 2, pp. 255–277, 2017.
[33] S. Bagui and K. Li, "Resampling imbalanced data for network intrusion detection datasets," *Journal of Big Data*, vol. 8, no. 1, pp. 238, 2021.
[34] Al-Turaiki and N. Altwaijry, "A convolutional neural network for improved anomaly-based network intrusion detection," *Big Data*, vol. 9, no. 3, pp. 233–252, 2021.
[35] A. V. Elijah, A. Abdullah, N. Z. JhanJhi, M. Supramaniam and O. Balogun Abdullateef, "Ensemble and deep-learning methods for two-class and multiattack anomaly intrusion detection: An empirical study," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 9, pp. 520–528, 2019.
[36] P. Maniriho, L. J. Mahoro, E. Niyigaba, Z. Bizimana and T. Ahmad, "Detecting intrusions in computer network traffic with machine learning approaches," *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 3, pp. 433–445, 2020.
[37] S. Meftah, T. Rachidi and N. Assem, "Network based intrusion detection using the UNSW-NB15 dataset," *International Journal of Computing and Digital Systems*, vol. 8, no. 5, pp. 477–487, 2019.
[38] F. A. Khan, A. Gumaei, A. Derhab and A. Hussain, "TSDL: A two-stage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30373–30385, 2019.
[39] Defense Advanced Research Projects Agency (DARPA). Kdd cup 1999 (1999). [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[40] S. Choudhary and N. Kesswani, "Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT," *Procedia Computer Science*, vol. 167, pp. 1561–1573, 2019.

[41] A. Khraisat, I. Gondal, P. Vamplew and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 384, 2019.
[42] B. Khraisat and A. Alazab, "A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges," *Cybersecurity*, vol. 4, no. 1, pp. 384, 2021.
[43] H. Zhang, J. L. Li, X. M. Liu and C. Dong, "Multi-dimensional feature fusion and stacking ensemble mechanism for network intrusion detection," *Future Generation Computer Systems*, vol. 122, no. Feburary, pp. 130–143, 2021.
[44] F. Gottwalt, E. Chang and T. Dillon, "CorrCorr: A feature selection method for multivariate correlation network anomaly detection techniques," *Computers and Security*, vol. 83, no. 10, pp. 234–245, 2019.
[45] A. Ponmalar and V. Dhanakoti, "An intrusion detection approach using ensemble support vector machine based chaos game optimization algorithm in big data platform," *Applied Soft Computing*, vol. 116, no. 7, pp. 108,295, 2021.
[46] N. Moustafa, "The UNSW-NB15 Dataset (Reduced)," 2015. [Online]. Available: https://cloudstor.aarnet.edu.au/plus/index.php/s/2DhnLGDdEECo4ys
[47] Z. Halim, M. N. Yousaf, M. Waqas, M. Suleman, G. Abbas *et al.,* "An effective genetic algorithm-based feature selection method for intrusion detection systems," *Computers and Security*, vol. 110, no. 34, pp. 102448, 2021.
[48] A. Thakkar and R. Lohiya, "A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges," *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 3211–3243, 2020.
[49] D. Azar, R. Moussa and G. Jreij, "A comparative study of nine machine learning techniques used for the prediction of diseases," *International Journal of Artificial Intelligence*, vol. 16, pp. 25–40, 2018.
[50] G. Guo, H. Wang, D. Bell, Y. Bi and K. Greer, "KNN model-based approach in classification," *Lecture Notes in Computer Science*, vol. 2888, pp. 986–996, 2003.
[51] M. A. Hearst, "SVMs – "A practical consequence of learning theory," *IEEE Intelligent Systems and Their Applications*, vol. 13, no. 4, pp. 18–21, 1998.
[52] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.
[53] Y. Yang, K. Zheng, C. Wu and Y. Yang, "Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network," *Sensors*, vol. 19, no. 11, pp. 2528, 2019.
[54] X. Larriva-Novo, C. Sánchez-Zas, V. A. Villagra, M. Vega-Barbas and D. Rivera, "An approach for the application of a dynamic multi-class classifier for network intrusion detection systems," *Electronics*, vol. 9, no. 11, pp. 1759, 2020.
[55] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer *et al.,* "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
[56] T. Murovic and A. Trost, "Genetically optimized massively parallel binary neural networks for intrusion detection systems," *Computer Communications*, vol. 179, no. 11, pp. 1–10, 2021.
[57] Q. Tian, J. Li and H. Liu, "A method for guaranteeing wireless communication based on a combination of deep and shallow learning," *IEEE Access*, vol. 7, pp. 38688–38695, 2019.
[58] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.
[59] M. Sani, C. Lei and D. Neagu, "Computational complexity analysis of decision tree algorithms," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11311 LNAI, pp. 191–197, 2018.