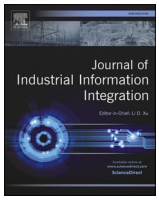
Journal of Industrial Information Integration 33 (2023) 100466

Contents lists available at ScienceDirect

Journal of Industrial Information Integration

journal homepage: www.sciencedirect.com/journal/journal-of-industrial-information-integration



Review article

Anomaly detection in NetFlow network traffic using supervised machine learning algorithms

Igor Fosić [a,*], Drago Žagar [b], Krešimir Grgić [b], Višnja Križanović [b]

[a] HEP-Telekomunikacije d.o.o. PS Osijek, M.Divalta 199, 31000 Osijek, Croatia
[b] Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek, Kneza Trpimira 2B, HR-31000 Osijek, Croatia

ARTICLE INFO

Keywords:
Supervised algorithm
Machine learning
Anomaly classification
NetFlow
Imbalanced dataset

ABSTRACT

Anomaly detection is an important method for monitoring network traffic where is important to successfully distinguish normal traffic from abnormal traffic. For this purpose, one could use the existing classification algorithms as a part of the machine learning (ML) process. In this paper, some of the classification algorithms (Stochastic Gradient Descent (SGD), Support Vector Machines (SVM), K-Nearest Neighbor (K-NN), Gaussian Naive Bayes (GNB), Decision Tree (DT), Random Forest (RF), AdaBoost (AB)) were tested on the public UNSW-NB15 dataset. Different encoding methods and ratios of training and test data resulted in the optimal parameters classifiers. Due to the imbalanced distribution of normal and abnormal network traffic data, both standard performance scores and additional classification performance scores (F2-score, Area Under ROC Curve (AUC)) were used, that better describe the obtained results. The RF Classifier with F2-score = 97.68% and AUC score = 98.47% obtained the best results using a representative subset within the original dataset due to the shorter duration of the computations. Features in the referential dataset were reduced by 82% and selected following the structure of the NetFlow data stream. Concerning similar studies, this paper compares several algorithms for anomaly detection and selects the best one for NetFlow data streams. The F2-score and AUC metric is applied, which achieves very high accuracy compared to classic metrics that do not show realistic accuracy in imbalanced datasets. Less time was spent using Label enoding (LE) with the same accuracy compared to One-hot (OH) encoding used in similar research. The novelty introduced by this paper is in the optimization of the ML process and influence of the ratio of data for learning and testing, different encoding methods of categorical features, and feature reduction on the NetFlow data streams.

1. Introduction

The increase in the number and sophistication of cyberattacks in recent years requires more advanced innovations in defense strategy today. Traditional methods of intrusion detection, anomaly detection, and deep packet inspection are no longer sufficient to meet today's evolving security threat trends. Hardware and software costs are decreasing as computing power increases, and machine learning is emerging as an alternative method or additional defense mechanism to prevent cyberattacks. This paper explores machine learning as a viable security option by investigating the ability to classify malicious network traffic using NetFlow network traffic data.

The motivation and objective of the research are to correctly and as securely as possible classify network traffic anomalies in the labeled dataset using classification as one of the machine learning methods. This paper includes the preliminary research and overview of the classification methods, testing the performance of each algorithm, studying the performance of the algorithm using general parameters related to machine learning such as the ratio of shared data for learning and testing, the method of encoding categorical data, measuring the performance using various metric parameters and their interpretation.

Mentioned classification problem was solved through machine learning using the Python programming language and the Scikit-learn platform. Multiple classification algorithms were used and machine learning parameters were optimized to obtain more successful classifications of network traffic containing anomalies and normal network traffic. Many classification algorithms are an integral part of the Scikit-learn platform and can be easily implemented and adapted depending

* Corresponding author.
E-mail address: igor.fosic@hep.hr (I. Fosić).

https://doi.org/10.1016/j.jii.2023.100466

Available online 20 April 2023

on the machine learning task involving classification or regression [1].

The features or characteristics of the observed problem are input parameters which can be quantified or measured, and the algorithm assigns a label to the output values, i.e. an assumption about which class the output value belongs to after the classification process is finished [2]. The problem of detecting anomalies in network traffic falls under the classification problem, which attempts to predict, correctly mark, and separates normal traffic from anomalies [1].

In this paper, we have proposed a new model for increasing the robustness of the classification of various network traffic mainly focusing on NetFlow streams. The presented method can be applied to various types of classifications and it does not impose a heavy computational load on an existing system. To substantiate our claim, we have implemented our approach on an existing NetFlow collector. The main challenge we confronted was the lack of a specific benchmark for evaluating the proposed model and comparing it with other existing ML models. To deal with this deficiency, we tried to create a benchmark that included the commonly used public IDS dataset UNSW-NB15. The main contributions of this work can be summarized as follows:

- Classifying different types of network anomalies and applying them to the benchmark.
- Creating own benchmark procedure and comparing the obtained results with other NetFlow ML classification models.
- Comparing and evaluating the obtained results by applying the criteria of accuracy, F2-Score and AUC.
- Selecting the best ML classification algorithm and applying it to the proposed model.

The paper is organized as follows: Section 2 gives an overview of the related work on the classification and detection of network traffic anomalies using machine learning. The following Section 3 describes the machine learning categories. An overview of the measures used to classify a given dataset is presented in Section 4, while Section 5 presents the research methodology. This sets out the general parameters for running classification algorithms (encoding method, ratio of training and test data), matching the NetFlow test set to the structure of the network traffic, and the results of comparing the performance of different supervised classification algorithms are presented. Also it presents experimental setup with real dataset and imported anomalies. The discussion is presented in Section 6 and Section 7 presents the conclusion.

2. Related work

The literature review covers topics related to anomaly detection in network traffic based on NetFlow [3–7], their binary [6,8-–9], and multi-class classification [3,6] via machine learning algorithms and comparison of their performance. Several machine learning researchers mentioned in the literature use older NSL-KDD dataset [10] for intrusion detection, which limits their comparison. Therefore, this literature review focuses on papers that use newer UNSW-NB15 [3,9–15] and CICIDS-2017 [11] datasets for machine learning classifiers.

Also, the literature review highlighted some shortcomings in the research such as testing algorithms on a small number of samples [16–18] and using algorithms without encoding or encoding categorical features in only one way [19,20]. Classifier performance evaluation is largely based only on classical accuracy derived from a confusion matrix and is inadequate performance metrics on an imbalanced dataset [12, 15,17,20–22]. In [23,24], it is recommended to use performance measures that are better adapted to imbalanced datasets, such as Fβ-score, AUC, and others. Feature reduction is introduced in [3,6,8,9,10,13,14], some features are not obtained from network traffic but are computed and characterize a particular data flow. Apart from [10,14], and [25], the encoding methods for categorical features used in sets with a large number of features are not well described. The studied literature does not explore the impact of the ratio of learning and test data on the applied models, and there is a need to investigate the optimal ratio for a specific problem and dataset. In many cases of related literature [3,6,11, 10,14,15], the Random Forest Classifier proved to be the best, but other performance measures should be considered due to the extremely imbalanced dataset mentioned earlier.

The literature review shows numerous possibilities for the integration of classification and cyber security with the help of machine learning. The performance of different machine learning algorithms can be evaluated and compared with the help of publicly available datasets, which contributes to a better understanding of the topic and provides an excellent opportunity to apply this technology in IT systems to reduce the risk of known and unknown security attacks. In the comparison of our approach with other approaches in the observed literature shown in Table 1, it can be concluded that our approach is better in all significant scores compared to other results. With a much smaller number of features (8) and well-defined machine learning parameters (train/test ratio, encoding method, number of samples), better results are achieved using the RF classifier. Although the training time (*) of our approach is also the best, it depends very much on the hardware used and on the possibility of using parallel processes, which was used in our approach.

3. Machine learning categories

According to [26–30], machine learning methods can be summarized into several categories depending on the degree of human control/supervision of the learning process: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning.

Supervised learning: the goal of supervised learning is to find a model based on labeled data and predict future data. The term "supervised" refers to a sample dataset where the output labels are already known. The knowledge gained in this way can be used to predict output values for each new sample of input data that was previously unknown or unseen during the training process. The two most common supervised machine learning tasks are classification and regression.

Unsupervised learning: the system seeks to detect patterns in unclassified and unlabeled data. The observed dataset has no labels or targets. One of the common tasks of unsupervised learning is to group similar data, i.e., cluster similar items or reduce data to a small number of important features. Unsupervised learning is more about extracting meaningful information from the data rather than predicting outcomes based on prior supervised training.

Semi-supervised learning is conceptually in between supervised and unsupervised learning and allows the exploitation of large amounts of unlabeled data in combination with smaller sets of labeled data. In general, semi-supervised machine learning algorithms attempt to improve the performance of one of the previously mentioned methods by using interrelated information.

Reinforcement learning: refers to goal-oriented algorithms that learn how to achieve a complex goal or maximize along a particular dimension over many steps. This method allows machines and software agents to automatically determine the ideal behavior in a given context to maximize performance. The agent needs feedback to learn which action is the best to achieve the maximum value.

Machine learning in the detection of network traffic anomalies is very well accepted due to the possibility of predicting unknown network traffic patterns and their classification. In the binary classification, it can be determined which traffic flows are normal traffic and which are anomalies. Supervised machine learning is very suitable for classification problems because the model can be trained very precisely since the learning datasets are already classified or labeled. Once learned, the model can classify unknown traffic with very high precision and thus take certain actions to stop anomalies (unwanted network traffic) and thereby improve the security of the IT system as a whole.

In this paper, supervised machine learning is applied because the

Table 1
Comparison of our approach to other related tasks with the UNSW-NB15 dataset.

Ref.	Year	ML algorithm	Train/Test size	Training time/ sec	Encoding method	Features	No of samples	RF Classification score		
								Accuracy	AUC	F-Score
[3]	2018	NB, RF, RT	90/10	12.91	/	45	2%	90.10%	98.90%	/
[8]	2021	RF, SVM, ANN	/	/	/	37	60%	98.67%	/	/
[11]	2020	LR, NB, K-NN, DT, AB, RF, CNN	70/30	/	/	49	100%	87.70%	96.10%	91.20%
[10]	2021	CNN+LSTM, GRU, BiLSTM, DT, RF	80/20	/	OH	47	10%	92.76%	92.73%	94.44%
[14]	2020	K-NN, SGD, DT, NB, RF, LR	80/20	/	LE	45	7%	95.43%	/	97.00%
[15]	2018	SVM, NB, DT, RF	/	5.69	/	42	10%	97.49%	/	/
our approach		SGD, SVC, K-NN, GNB, DT, RF, AB	60/40	**5.51***	LE	**8**	30%	**99.06%**	**98.47%**	**97.68%**

used dataset has all records uniquely marked. Normal traffic is marked with 0 (class 0) while anomalies or records as network attacks are marked with 1 (class 1).

4. Classification of the imbalanced dataset

Exceptions or anomalies are examples or parts of data that do not fit the majority in some way. Detecting anomalies in data is called deviation or anomaly detection, and the machine learning category that focuses on this problem is called classification. Classification algorithms can be used for binary classification tasks with a highly imbalanced class distribution and can be effective for imbalanced datasets with none or very few examples of minority classes. A typical problem in classification is a class imbalance, where for a given dataset in the learning process, there may be hundreds or thousands of samples in one class and only a dozen samples in another class. This results in poor prediction models, especially for the minority class. In general, the minority class is more important because it represents anomalies. Therefore, the problem is more sensitive to the classification errors of the minority class than those of the majority class. A common error in imbalanced classification represents the usage of the classical method of classification accuracy, which is defined by:

$$\text{accuracy} = \frac{\text{total number of correct predictions}}{\text{total number of predictions}} \tag{1}$$

or for binary classification with confusion matrix as:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

where prediction TP = true positive, TN = true negative, FP = false positive and FN = false negative. The classical method of classification accuracy can be very misleading because if most samples in a dataset belong to one class, the model usually always predicts a high classification accuracy. A better view of performance in imbalanced datasets is achieved by F2-score and Area Under the ROC Curve (AUC) scores. AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is the probability that the model ranks a random positive example more highly than a random negative example [31]. Typical AUC graph is shown in Fig. 1.

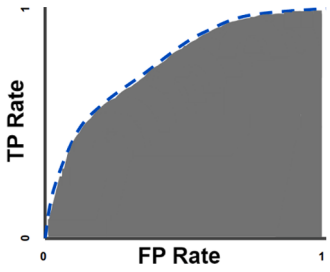


Fig. 1. Graphical representation of AUC.

$$TPR = \frac{TP}{TP + FN};\ FPR = \frac{FP}{FP + TN} \tag{3}$$

$$F\beta = \frac{\left(1 + \beta^2\right) * \left(\frac{TP}{TP+FP}\right) * \left(\frac{TP}{TP+FN}\right)}{\beta^2 * \left(\frac{TP}{TP+FP}\right) + \left(\frac{TP}{TP+FN}\right)} \tag{4}$$

F2-score (β=2): aims to minimize false negative predictions [32].

Different classification algorithms can be used for binary imbalanced classification problems, where the negative case (class 0) is considered "normal" and the positive case (class 1) is considered a deviation or anomaly. Binary classification problems can be considered the most developed branch of machine learning from imbalanced data. They are observed in various domains, such as medicine [30] (sick versus healthy condition), computer security [17,21,33,34] (valid activity versus unauthorised or malicious activity), or computer vision [29] (target object versus background). In such a scenario, the relationship between classes is well defined: one class is the majority while the other is a minority. This problem requires balancing the distribution of classes or emphasizing the minority class when a particular classifier is used [24, 32,35,36]. According to [36], supervised machine learning algorithms can be classified as follows:

- Logic-based algorithms (Decision Tree is the most common example).
- Support vector machine algorithm.
- Statistical algorithm (Naïve Bayes is the most common example).
- Lazy learning algorithm (KNearest Neighbors is the most common example).

5. Methodology

Classification can be performed on an unstructured or structured dataset where the samples of the dataset are classified according to a specific label or category. For new input data, the class or label assigned to it is provided by one of the machine learning algorithms. Common terms in this field include:

- Classifier – an algorithm that relies on the training dataset and then assigns a class to the new data.
- Classification model – attempts to conclude from the input values given for training. It predicts the class labels/categories for the new data.
- Feature – a parameter found in each dataset that can contribute sufficiently to build an accurate predictive model.

The classification model can be built according to the steps shown in Fig. 2:

- Collecting, cleaning, and preprocessing the data (1).

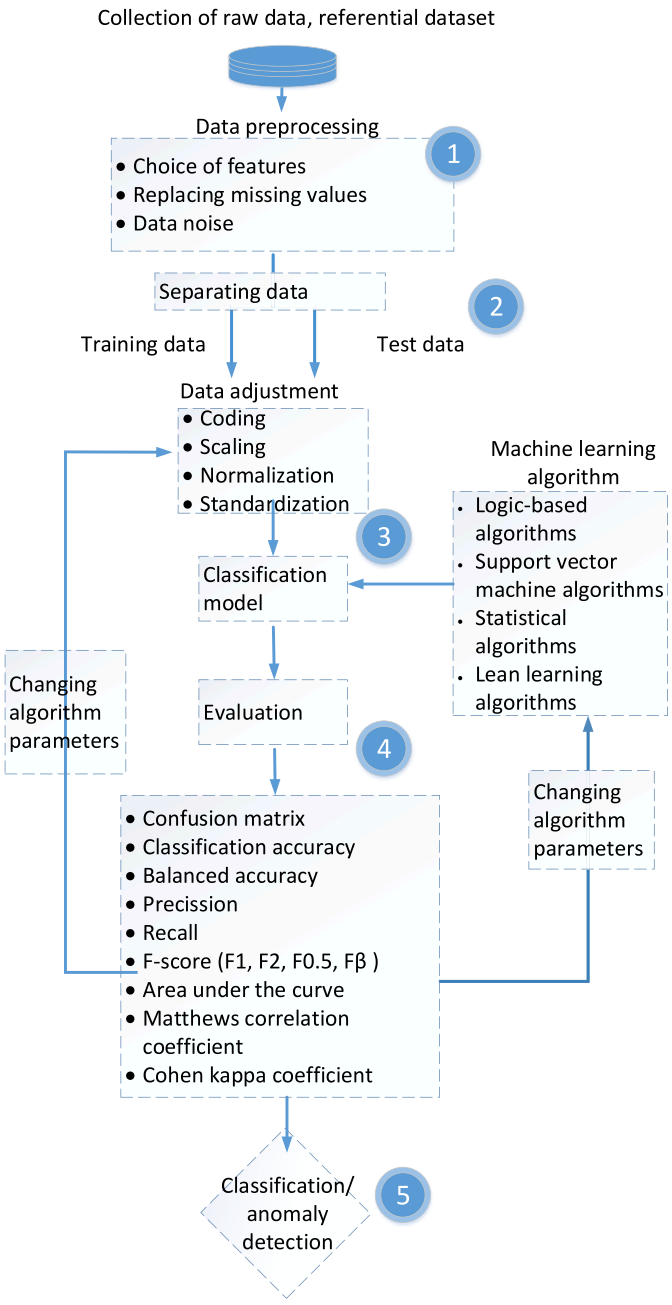


Fig. 2. Proposed model for classification.

- Dividing the data into training and testing data for the classification model (2).
- Preparing the classifier: the scikit-learn platform, which is based on the Python programming language, has built-in methods called *fit_transform* functions that map the input dataset X and the corresponding y label to prepare the classification model (3).
- Evaluation of a classification model on a test dataset (4).
- Label prediction for new observation data, a method that returns a mapped y-label for input instance X (5).

5.1. Dataset selection

The initial stage of the proposed classification approach involves gathering traffic flow data. This data can be gathered either by observing the real-time traffic of recognized network infrastructure or by utilizing existing public datasets that relate to a specific problem. In this particular case, the UNSW-NB15 dataset was selected as the input data. The UNSW-NB15 dataset is more recent than other publicly available datasets used in previous studies presented in the literature. It contains all the features found in the NetFlow data structure, it is very well documented and described and contains a large number of network traffic anomalies. All records are uniquely labeled as normal or abnormal traffic. Other well-known datasets used for this purpose are CTU-13, KDDCUP993, CIC-IDS −2017, etc., which are also used in recent publications but the selected dataset UNSW-NB15 has more common features with NetFlow structure than others. The raw traffic (Pcap files) of the UNSW-NB15 dataset was created using the IXIA PerfectStorm tool in the Cyber Range Lab at UNSW Canberra to generate a hybrid of real modern normal activities and synthetic contemporary attack behaviors. This dataset contains nine types of attacks, namely., Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. A total of 49 features are generated with the class label. The total number of records is 2,540,044 and they are stored in four CSV files [37]. Different types of features are used (categorical, numeric (integer, decimal, and binary), and temporal). The dataset obtained from the NetFlow protocol of network devices is not identical to the original UNSW-NB15 dataset, and therefore it is necessary to reduce its features to those that are identical, i.e. those that will be used. By comparing the NetFlow and UNSW-NB15 datasets, common features of both datasets were observed and the original dataset was significantly reduced to only a few features. The following sections of this paper will be based on these features. The structure of the NetFlow dataset was created according to the recommendations of network equipment manufacturers [38]. The network device was configured as follows:

```
flow record SW_FLOW_RECORD
match ipv4 protocol
match ipv4 source address
match ipv4 destination address
match transport source-port
match transport destination-port
collect counter bytes long
collect counter packets long
collect timestamp absolute first
collect timestamp absolute last
```

It sends data that matches the UNSW-NB15 dataset in the following features: *srcip(1), sport(2), dstip(3), dsport(4), proto(5), state(6), dur(7), sbytes(8), spkts(17), stime(29), ltime(30)*. Features that can be further obtained from the network device and are not included in the UNSW-NB15 dataset are InputInterface, SourceVLAN, InputsourceMACaddress and ExportingSystemID.

The experimental model was developed based on a customized UNSW-NB15 dataset and was divided into two parts: training and test dataset. A customized dataset is adapted to the NetFlow structure of the data flow that can be obtained from network devices, so only features that will be included in the NetFlow dataset are selected. According to the description in [37], the dataset contains normal (class 0) and abnormal (class 1) traffic representing different types of network attacks on the IT system. Fig. 3 depicts the distribution of the two types of data - the full dataset which contains 2540,047 records, and the partial dataset which includes 700,044 records. The figure shows that there is a significant imbalance between the two types, or classes, of data, with both types being represented in almost equal proportions. A comparison, shown in Fig. 4, was made between the full and partial datasets to determine their respective representation and their effect on the computation speed of each classification algorithm. It was observed that the learning time for classification is substantially reduced when using the smaller partial dataset, which took only 4 s compared to 48 s for the full dataset, resulting in a 91% reduction in learning time for the partial dataset. Despite using the smaller dataset, the obtained results can be

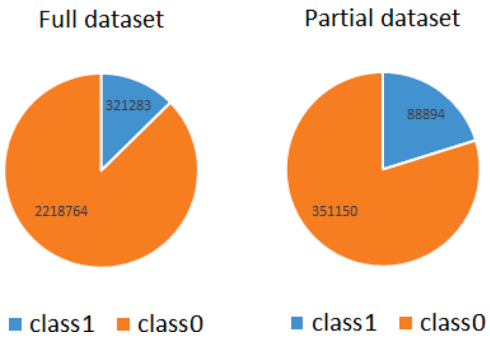


Fig. 3. Comparison and distribution of classes in the full and partial datasets.

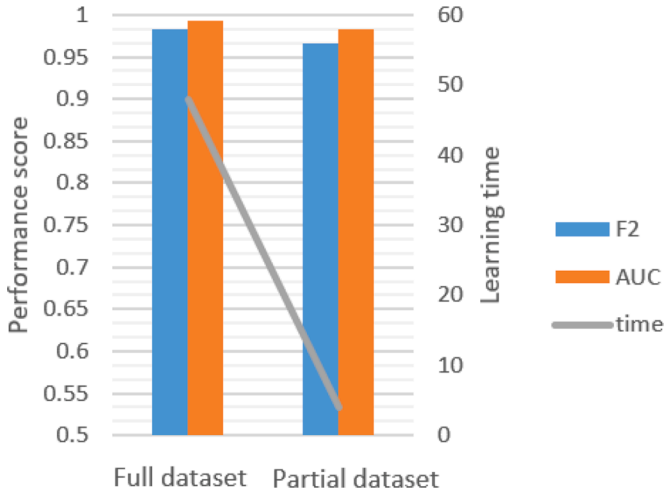


Fig. 4. Comparison of performance scores and time between full and partial dataset.

applied to the entire source dataset.

5.2. Dataset preprocessing

The classifier was tested on a portion of the UNSW-NB15 dataset that exclusively consists of features that align with the NetFlow feature structure. For this purpose, features *sport(2), dsport(4), proto(5), state (6), dur(7), sbytes(8), spkts(17), Label(48)* were selected. The feature *Label* (48) classifies normal samples or anomalies in the dataset and is crucial for evaluating the performance of the machine learning classifier. Features such as source and destination IP addresses (*srcip and dstip*) are not used because they have no significance for defining the traffic class, i.e., potentially anomalous traffic is not determined by the IP address of the device. The same applies to the features *Stime* and *Ltime*, which indicate the start and end times of a data stream and are not included in the dataset as input data for the classifier.

All machine learning algorithms work exclusively with numeric feature values and must be modified if they are defined as categorical. Although some features of the dataset are defined as numeric (*sport, dport*), they contain some sign values that cause machine learning to malfunction. All these values must be deleted or converted to numeric values. In the UNSW-NB15 partial dataset, such cases are few and far away, only 0.0096% (67 records out of 700,001). Normally, the deletion of these records may lead to the loss of potentially important information. But in our case, all these records are labeled with class 0, indicating normal traffic, so deleting these records will not have much impact on the classification/anomaly detection result. There are far more records with the so-called NaN value, i.e., the Not-A-Number value. This feature indicates that the value is not a number, but it is also not a categorical value and cannot be used in the machine learning process.

These features (*ct_flw_http_mthd, is_ftp_login, attack_cat*) are not included in the machine learning process and do not need to be deleted or replaced. After preprocessing is complete, the dataset used for the further classification process has the structure shown in Table 2.

As suggested in the methodology, it is necessary to adapt all features to the structure of the NetFlow record and recode the categorical features into numerical values suitable for the machine learning process. Although certain features are exclusively numeric, for some unknown reason they contain some non-numeric values that interfere with the machine learning process. Therefore, certain records were deleted. Deleting the records did not significantly change the features of the two datasets, as can be seen in Table 3. In the full dataset, 308 records (0.0121%) were deleted, while in the partial dataset, 67 records (0.0096%) were deleted. They were all class 0 in features *sport* (source port) and *dsport* (destination port) with character values "0xcc09", "0 × 20,205,321″, "0xc0a8", "0 × 000b", "0 × 000c", "-" instead of numeric values. One of the options was to replace these values with numeric values, but since the meaning of these values was unknown and all of these records were marked as class 0 (normal traffic), they were deleted since they were not of interest (class 1 or anomalies). After deleting this data, the class distribution did not change significantly, as presented in Table 3.

5.3. Separating the training and testing data

To identify the best set of model features, it is crucial to segregate the training and testing data. The training dataset is employed to construct models utilizing various classifier parameter settings. Subsequently, each classifier is assessed using the test dataset, which comprises samples that have known classification labels. By predicting the classification of the test dataset, an estimation of the model accuracy can be obtained. The *scikit-learn* platform provides the implementation of the process of splitting the original dataset via the built-in function *train_test_split ()*. Based on the metric parameters (classification errors) over the test dataset, the optimal parameter of the classification model is determined. Selecting the appropriate ratio between the size of the training and testing datasets is crucial to ensure that the model can provide accurate outcomes. If the training set is too small, the model may perform poorly, whereas if the testing set is too small, it may not offer adequate information to accurately assess the model's performance. Thus, there is no universally optimal value for this ratio. There are some common train-test size ratios that are often used in the literature. Authors suggest using a 70/30, 80/20 or 90/20 train-test size ratio for most machine learning problems [3,11,10,14,20,39–41]. Different ratios (10/90, 20/80, 30/70, 40/60, 50/50, 60/40, 70/30, 80/20, and 90/10) were used to divide the datasets into the training and testing datasets for the performance assessment of models [42]. In [43] data were split to training and test sets, keeping 50%, 60%, 70%, or 80% of data in the training set. From abowe it can be concluded that:

- 70:30 - is a common ratio used for small to medium-sized datasets.
- 80:20 - is another common ratio that is often used for small to medium-sized datasets.

Table 2
Chosen features of the dataset.

#	Feature	Feature type	Description
2	sport	Integer	Source port
4	dsport	Integer	Destination port
5	proto	Categorical	Protocol type
6	state	Categorical	Protocol status flag, mark (-) if not used
7	dur	Decimal	Duration
8	sbytes	Integer	Bytes from source to destination
17	spkts	Integer	Packets from source to destination
48	Label	Binary	0 for normal and 1 for attack records

Table 3
Absolute and relative values of both classes in datasets after preprocessing.

	Total records	class 0	class 1	class 0	class 1
Full dataset	2,539,739	2,218,456	321,283	87.35%	12.65%
Partial dataset	699,934	558,547	141,387	79.8%	20.2%

- 90:10 - this ratio is used when the dataset is very large, and it may be difficult to train the model on a larger proportion of the data.

60:40 - This ratio is sometimes used when the dataset is relatively small and there is a need for more data to train the model.

The "*test_size*" value of the *train_test_split ()* function is a parameter that determines the ratio of training and test data and has a value between 0 and 1 because it determines the part of the data set that will be included in the split of test and training data. It is very important that the samples of the training and testing datasets are randomly assigned to be a representative sample of the original data set. In this way, the result would be a representative dataset, because the comparison of the performance of machine learning algorithms should be based on representative subsets of the original dataset. This is achieved by setting the seed to the pseudo-random number generator used to share the dataset, or by setting the "random_state" parameter to an integer value [1]. Many classification problems solved by machine learning use an imbalanced dataset, and it is desirable to split the dataset into a training and a test dataset such that the same sample as in the original dataset is maintained in each class. The identical distribution of samples in the training and test datasets as in the original dataset is achieved by setting the argument "*stratify*" to y. Using the function *train_test_split ()* in the form:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=<decimal number>, random_state = 1, stratify = *y*)

The original datasets *X* and *y* are taken as input, and the return is the dataset split into two subsets specifically for training and testing, *X_train* and *X_test*. The dataset *y* is also split according to the same parameters and serves to validate the performance of the classification model. In the specific example of UNSW-NB 15 datasets described in Table 2. in the previous section, the input dataset X contains the features *sport, dsport, proto, state, dur, sbytes, spkts*, while the input set y contains the labels (classes) of the samples or the *Label* feature. The result of the applied *train_test_split ()* function returns subsets of the original dataset for training and testing. When splitting the data, the representativeness of the training and test data was taken into account to obtain an equal ratio of the original set.

The input data *X* and *y* are dataset records split in the sense that *X* is a data matrix of dimension *n*m*,

$$\begin{bmatrix} X_{11} & > & X_{1m} \\ \vdots & \ddots & \vdots \\ X_{n1} & > & X_{nm} \end{bmatrix}$$

n is the number of records, m is the number of features and y is a one-dimensional field [y1…yn] with network data type labels (values 0 or 1), where 0 denotes normal traffic and 1 denotes an anomaly. It is not advisable to split the data into a train-test set ratio of less than 50/50, such as 30/70 or 20/80, as it may result in inadequate data for either training or testing. A small training set may lead to an underfit model that cannot capture the intricacies of the problem, while a small testing set may not provide enough information to accurately evaluate the model's performance. The following values were chosen for the test_size parameters: 0.2, 0.3, 0.33, 0.4, 0.5 for each iteration of the classifier algorithm change. It is important to note that while it is generally not recommended to split the data into a train-test set ratio of less than 50/50 [42], there may be certain situations where this approach could be appropriate. In such cases, techniques like cross-validation or bootstrap sampling can be used to estimate the performance of the model. The *random_state* parameter allows samples to be taken for the same random principle for the training and test part in each iteration, while the *stratify* parameter ensures an equal distribution of normal traffic and anomalies in all parts of the data. This procedure obtained partial datasets *X_train, X_test, y_train, y_test* that will be used as input parameters for each classification algorithm.

5.4. Data adjustment

We have extracted features from the selected data set that need to be converted into numerical values that are suitable for the machine learning process. This step is also an important part of the methodology since NetFlow data in its original form contains character features that need to be encoded into numeric or Boolean values. In this paper, the two most commonly used techniques are considered: LE and OH coding. The choice of encoding type depends on the dataset and the model applied. The general recommendations for choosing these two encoding types depend on whether the categorical features are ordinal and how many unique values categorical features contain.

The reduced and split dataset adapted to the NetFlow structure contains two categorical features that need to be converted to numeric values. The categorical features are proto and state data. They are converted to numeric values by encoding functions. Both of the above-mentioned encoding types were used in the calculation of classification accuracy in all split data ratios, as described in the previous subsection. When the LE method was used, the number of features did not change, while when the OH encoding method was used, there was a multi-fold increase in the number of features, which significantly affects the speed of the classification. Table 4 shows the increase in the number of features after applying each of the encoding methods. The largest increase in the number of features is due to the *state* feature, which has 134 unique values, while the *proto* feature has 16 unique values.

5.5. Results of machine learning algorithms in solving classification problem

Following the methodology defined in Section 5, we compared several machine learning algorithms suitable for solving classification problems with two classes: SGD, SVM, K-NN, GNB Classifier, DT, RF and AB on a portion of the UNSW-NB15 dataset containing both data on normal network traffic and data on various anomalies. Normal network traffic is labeled differently than anomalies and it is possible to determine the success of a particular algorithm using this labeled dataset.

The process of testing classification algorithms includes both methods of data encoding, all ratios between training and test data, and F2 and AUC as measures of the success of a single classifier in all combinations. To compare the performance of the algorithms, we selected the previously described portion of the UNSW-NB15_1.csv dataset, which is a representative set of the original dataset. In evaluating the performance, we observed the value of the F2-score, and AUC metrics corresponding to the dataset previously characterized as highly imbalanced in classes 0 and 1. The idea and used pseudocode in

Algorithm 1 is to determine the optimal ratio of training and test data and the way of encoding the categorical features in the iterative procedure that leads to the selection of the best classifier. The focus of benchmarking was achieving the highest possible performance scores or the lowest possible number of false-negative classifications. This is because negative predictions are related to normal traffic and positive

Table 4
The comparison of the dataset dimensionality after encoding.

	Original dataset	Label encoding	One-hot encoding
Total number of features	8	8	156

predictions are related to anomalies. It is important to keep false-negative predictions to a minimum, as these are anomalies that are classified as normal traffic and can harm the proper functioning of IT's systems. In practice, such cases mean that certain anomalies or network attacks are not detected and are carried out without proper response and prevention. The pseudocode program shows how the success of classification is tested depending on the encoding categorical features and the ratio of training and test data.

- Training and test data ratio: 0.2, 0.3, 0.33, 0.4, 0.5
- Encoding method: LE, OH
- Classification algorithms: SGD, SVM, K-NN, GNB, DT, RF, AB

For the overall evaluation of each performance metric, the mean of the results of all classification algorithms for the observed encoding parameter and split ratio was used.

The comparison of the mean values of F2-score and AUC metric for each encoding method is shown in Fig. 5, which indicates that slightly better results are achieved by algorithms using LE versus OH encoding.

The computation time required, shown in Fig. 6, is also in favor of the LE method. Although the computation time is not crucial, it can be used as one of the parameters for comparison. The diagram shows the mean time required to compute all classifiers in all split training and test ratios.

Since the results of the relevant performance metrics are slightly better with the LE method, this encoding method is used for further consideration of the performance of a particular machine learning classifier. The following performance comparison contains an overview of the relationship between training and testing data. Fig. 7 suggests that the 60/40 ratio between training and test data provides the best classification results.

The value on the y-axis of these diagrams is the average value of the performance evaluation of all classifiers according to the observed ratio.

The following parameters were set for further optimization and selection of classifiers: the LE method and the 60/40 ratio between training and test data. Considering these selected parameters, the individual performance of each of the observed classifiers is shown in Table 5. The selection of the train/test data ratio was based on the F2-score and AUC measures displayed in Fig. 7, which took into consideration the classification success of all classifiers. Meanwhile, the F2 and AUC measures shown in Table 5 were utilized to determine the best classifier using the pre-selected machine learning classification parameters, which involved a chosen train/test ratio of 60/40.

Considering the most important scores F2-score, and AUC, as well as Cohen Kappa and Matthews coefficient with the previously selected method of encoding and ratio of training and test data, the Random Forest Classifier stands out because it achieves the best scores with given parameters.

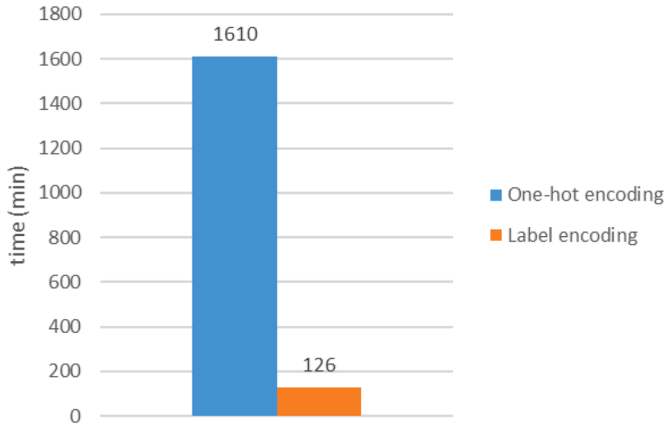


Fig. 5. Encoding time concerning the categorical features encoding method.

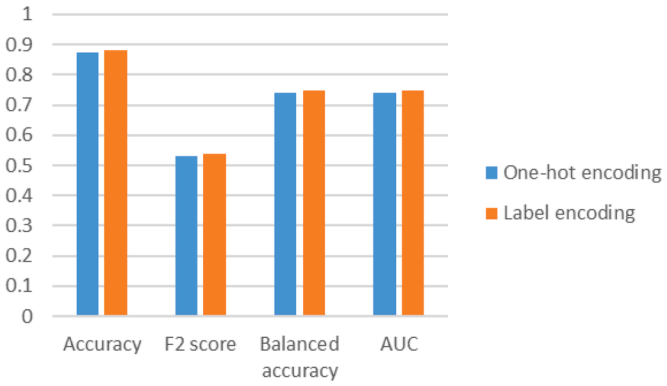


Fig. 6. Classifier performance concerning the categorical encoding method.

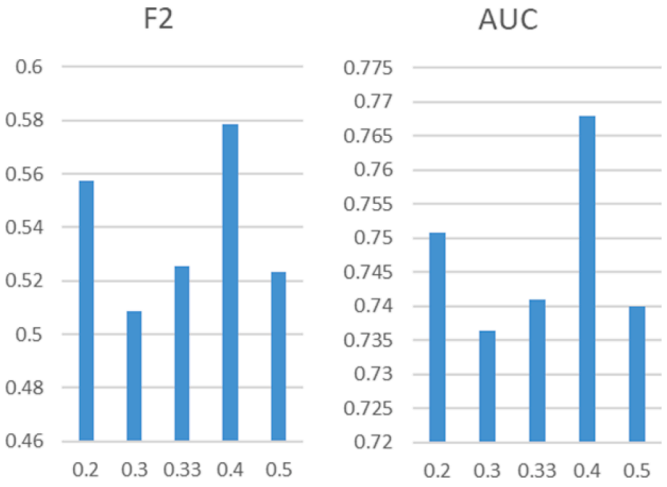


Fig. 7. Performance of the classifier concerning training and test data ratio.

5.6. Experimental setup and results

The research's experimental section comprises a NetFlow dataset gathered from a real isolated network environment, with real users on the network. The data was collected in the form of NetFlow over three weeks on a dedicated network infrastructure with normal user activities and after conversion, it consisted of 1,256,026 records, all marked as normal traffic. The experimental method includes two parts:

- The first part incorporates the NetFlow dataset and anomalies from the reference dataset, ensuring all anomaly records from the reference dataset are present.
- The second part uses a modified set of records that match the number of records in the dataset used in Section 5.2.

In the first part the proposed model was tested with anomalies were extracted from the UNSWB-NB15 dataset and added to the NetFlow dataset, resulting in 321,263 records of anomalies. The ratio of anomalies to normal traffic was 20.37% to 79.63%, respectively, which was in accordance with the research's unbalanced dataset definition.

The experiment's second part utilized a NetFlow dataset that had been reduced, consisting of 699,934 randomly selected records with included anomalies that were inserted from the UNSWB-NB15 reference dataset. This dataset was more comparable in size to the one described in chapter 5.2 and maintained the same proportion of normal traffic records (79,69%) and anomalies (20,31%).

The classification results were slightly weaker than when classifying only the reference dataset as shown in Table 6. The Random Forest classifier's predefined settings were used to classify anomalies, but it can

Table 5
Performance comparison of all classifiers.

	SGD	SVC	K-NN	GNB	DT	RF	AB
Acc	0.768825	0.799282	0.970156	0.768524	0.989109	**0.990654**	0.968958
Prec	0.456624	0.78934	0.966188	0.247369	0.97375	0.978821	0.957634
Rec	0.759829	0.008746	0.883177	0.071404	0.972299	0.97483	0.885511
F2	0.570439	0.017301	0.922819	0.11082	0.973024	**0.976822**	0.920161
AUC	0.765466	0.504078	0.937676	0.508203	0.982832	**0.984745**	0.937797
Cohen	0.42544	0.01293	0.904367	0.022341	0.966201	0.970969	0.900931
Matthews	0.451257	0.069286	0.905769	0.028111	0.966202	0.970972	0.901995

Table 6
Comparison of full and reduced dataset with randomly inserted anomalies.

	Full dataset (all anomalies)	Reduced dataset (randomly sampled normal records and anomalies)
# records	1,577,289	699,934
normal	79.63%	79.69%
anomalies	20.37%	20.31%
AUC	0.9463	0.9353
F2	0.906	0.895

Algorithm 1
: F2, AUC score.

```
Data: UNSW NB15 dataset
Input1: train-test split size
Input2: encoding type
Input3: classification algorithm
Output: F2 score, AUC score
Initialization:
foreach split ∈ Input1 do
    foreach enc ∈ Input2 do
        foreach alg ∈ Input3 do
            get F2, AUC score (alg, enc, split)
        end
```

be more fine-tuned to reduce false negative classifications and significantly reduce false positive classifications by relearning with a modified dataset. This very high accuracies of the AUC and F2-score measures can be considered a good basis and proof-of-concept for implementing the model in a real environment.

6. Discussion

Anomaly detection is an important method for monitoring network traffic. In the case of anomaly detection problems, it is important to successfully distinguish between normal and abnormal traffic. For this purpose, the existing classification algorithms can be well utilized as a part of machine learning and implemented in IT. Adaptations of the input data with a suitable classification algorithm can give very good results in detecting and classifying anomalies in network traffic. The highest value in the table of results has classic accuracy with 99.06%. which is consistent with the values shown in previous related researches. The used reference dataset was very well classified and with a smaller number of selected features achieved respectable results. For the imbalanced dataset used in this study, F2-score and AUC are more appropriate, taking into account the uneven distribution of classes in the dataset. In fact that anomalies are generally less common, this class of data is less represented in the overall dataset. Looking at F2-score and AUC, the highest values are also achieved using the Random Forest algorithm. The dataset was adjusted by encoding the category features by the LE method, which showed better performance according to the overall accuracy results and the time required for the calculation than OH encoding method. Compared to encoding methods in the literature, this method is less represented, but in this paper it has proven to be more appropriate than OH encoding of category features. The separation of data for the learning and testing process was done with a ratio of 60:40. 60% is used in the learning process while the rest of the data was used to test the performance of the classifier. No research has been conducted on the most appropriate separation ratio, and various ratios were used in this paper, of which a coefficient of 0.4 proved to be the most appropriate as shown in Fig. 7. The results confirm the correct choice of Random Forest algorithm as the most appropriate classifier for the model and will be able to detect network traffic anomalies of an IT system based on telemetry NetFlow data.

The significance of the train/test data ratio is less crucial in real-life anomaly detection scenarios compared to the method used for selecting the best classifier to solve the problem. In such scenarios, the anomaly detection model no longer needs to be tested for success as the parameters of the classifier will have already been chosen during model creation to ensure optimal anomaly detection performance. This study examines the parameters that have the most impact on the performance of classifier for detection model when applied in a real-life environment. The amount of data required for effective model learning was not specifically examined in this study, but the classification of a real NetFlow dataset was compared with imported anomalies from the reference UNSWB-NB15 dataset, which provides a reliable representation of real-life scenarios and different types of cyber attacks. The total increase in the number of records (twice as much data) did not result in a significant difference in terms of increasing accuracy. While having an adequate amount of data for training is important for achieving successful detection, it is likely more critical to have accurately labeled benign traffic and as many accurately labeled anomaly cases as possible to minimize the occurrence of false positive or false negative classifications during anomaly detection.

The model suggested in this research can be easily utilized and customized for other classification problems with the use of a distinct dataset and appropriate feature selection. Even with minimal feature adjustments, the model can still yield favorable classification outcomes, making it suitable for implementation in real-time environments because of its high performance.

7. Conclusion

In this paper, some of the classification algorithms were tested on the public UNSW-NB15 dataset used to test the IDS systems. Furthermore, the impact of different methods of encoding categorical features and different ratios of training and testing data on the performance of classification algorithms was researched. Different encoding methods and ratios of training and testing data resulted in the optimal general parameters used for each classifier. Due to the imbalanced data distribution in normal and abnormal network traffic, encoding methods were used in addition to standard performance scores and additional performance scores to better describe the obtained results. Concerning all relevant performance indicators, the best results were obtained using the Random Forest Classifier with an F2-score of 97.68% and 98.47% AUC. The experimental tests were performed with a representative set of the original dataset due to the shorter duration of the computations. The

small time of execution is not so much crucial in the calculation and classification of already collected and processed data as it is important in real-time environments where the speed of classification can be crucial. The Random Forest algorithm can use all available hardware resources such as parallel computing on multiple processors which greatly reduces the time required for computations.

Seven features with the addition of the *Label* feature were selected according to the structure of the NetFlow data stream, discarding features that have no bearing on the definition of anomalies, such as the IP address or the time of occurrence of the data stream. Categorical features were encoded using a simple label encoding method, which showed better results compared to the One-hot encoding method and required less time to compute the classification.

Based on the results, further research should focus on the following:

- Additional parameterization of the Random Forest Classifier to try to achieve an even lower number of false-negative predictions (undetected anomalies).
- Possible reduction of the dimensionality of the input dataset to further improve model performance.
- Using the same input data for unsupervised classifiers and comparing it with the results of this paper.
- Implementing the proposed model in a real network environment and classification of NetFlow streams in real-time.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] I. Fosic, D. Zagar, K. Grgic, Network traffic verification based on a public dataset for IDS systems and machine learning classification algorithms. 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO), IEEE, May 2022, pp. 1037–1041, https://doi.org/10.23919/MIPRO55190.2022.9803674.

[2] F. Pedregosa, et al., Scikit-learn: machine learning in Python, J. Mach. Learn. Res. (2011) 2825–2830.

[3] P. Dahiya, D.K. Srivastava, Network intrusion detection in big dataset using spark, Procedia Comput. Sci. 132 (2018) 253–262, https://doi.org/10.1016/j.procs.2018.05.169.

[4] R. Kumar, R. Singh, Netflow based cyber threat classification using J48 and random forest machine learning algorithms, Int. J. Eng. Adv. Technol. 9 (1) (Oct. 2019) 2973–2979, https://doi.org/10.35940/ijeat.A1326.109119.

[5] S. and M. N. P.M. Sarhan Mohanad, Layeghy, NetFlow datasets for machine learning-based network intrusion detection systems. Big Data Technologies and Applications, Springer International Publishing, Cham, 2021, pp. 117–135. H. and H. R. and R. S. and C. N. Deze Zeng and Huang

[6] M. Awad, S. Fraihat, K. Salameh, A. al Redhaei, Examining the suitability of NetFlow features in detecting IoT network intrusions, Sensors 22 (16) (Aug. 2022) 6164, https://doi.org/10.3390/s22166164.

[7] T. Bakhshi, B. Ghita, On internet traffic classification: a two-phased machine learning approach, J. Comput. Netw. Commun. 2016 (2016) 1–21, https://doi.org/10.1155/2016/2048302.

[8] M. Ahmad, Q. Riaz, M. Zeeshan, H. Tahir, S.A. Haider, M.S. Khan, Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using UNSW-NB15 data-set, EURASIP J. Wirel. Commun. Netw. 2021 (1) (2021), https://doi.org/10.1186/s13638-021-01893-8.

[9] W. Xu, Y. Fan, C. Li, I2DS: interpretable intrusion detection system using autoencoder and additive tree, Secur. Commun. Netw. 2021 (3) (2021), https://doi.org/10.1155/2021/5564354.

[10] M. Ahsan, R. Gomes, Md.M. Chowdhury, K.E. Nygard, Enhancing machine learning prediction in cybersecurity using dynamic feature selector, Journal of Cybersecurity and Privacy 1 (1) (2021) 199–218, https://doi.org/10.3390/jcp1010011.

[11] N. Elmrabit, F. Zhou, F. Li, H. Zhou, Evaluation of machine learning algorithms for anomaly detection, in: 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), IEEE, Jun. 2020, pp. 1–8, https://doi.org/10.1109/CyberSecurity49315.2020.9138871.

[12] M. Nawir, A. Amir, O.B. Lynn, N. Yaakob, R.Badlishah Ahmad, Performances of machine learning algorithms for binary classification of network anomaly detection system, J. Phys. Conf. Ser. 1018 (1) (2018), https://doi.org/10.1088/1742-6596/1018/1/012015.

[13] N. Koroniotis, N. Moustafa, E. Sitnikova, J. Slay, Towards developing network forensic mechanism for botnet activities in the IoT based on machine learning techniques, Springer International Publishing 235 (2018), https://doi.org/10.1007/978-3-319-90775-8_3.

[14] G. Kocher and G. Kumar, "Performance analysis of machine learning classifiers for intrusion detection using UNSW-NB15 dataset," pp. 31–40, 2020, doi: 10.5121/csit.2020.102004.

[15] M. Belouch, S. el Hadaj, M. Idlianmiad, Performance evaluation of intrusion detection based on machine learning using apache spark, Procedia Comput. Sci. 127 (2018) 1–6, https://doi.org/10.1016/j.procs.2018.01.091.

[16] A. Prakash, R. Priyadarshini, An intelligent software defined network controller for preventing distributed denial of service attack, in: 2018 S International Conference on Inventive Communication and Computational Technologies (ICICCT), IEEE, Apr. 2018, pp. 585–589, https://doi.org/10.1109/ICICCT.2018.8473340.

[17] J. Ye, X. Cheng, J. Zhu, L. Feng, L. Song, A DDoS attack detection method based on SVM in software defined network, Secur. Commun. Netw. 2018 (2018) 1–8, https://doi.org/10.1155/2018/9804061.

[18] F.A. Khan, A. Gumaei, A. Derhab, A. Hussain, TSDL: a two-stage deep learning model for efficient network intrusion detection, IEEE Access 7 (March 2020) (2019) 30373–30385, https://doi.org/10.1109/ACCESS.2019.2899721.

[19] Md.A.M. Hasan, M. Nasser, B. Pal, S. Ahmad, Support vector machine and random forest modeling for intrusion detection system (IDS), J. Intell. Learn. Syst. Appl. 06 (01) (2014) 45–52, https://doi.org/10.4236/jilsa.2014.61005.

[20] M.A. Umar and C. Zhanfang, "Effects of feature selection and normalization on network intrusion detection," pp. 1–25, 2020, doi: 10.36227/techrxiv.12480425.

[21] D. Li, C. Yu, Q. Zhou, J. Yu, Using SVM to Detect DDoS Attack in SDN Network, IOP Conf. Ser. Mater. Sci. Eng. 466 (1) (Dec. 2018), 012003, https://doi.org/10.1088/1757-899X/466/1/012003.

[22] C. Khammassi, S. Krichen, A GA-LR wrapper approach for feature selection in network intrusion detection, Comput. Secur. 70 (June) (2017) 255–277, https://doi.org/10.1016/j.cose.2017.06.005.

[23] H. M, S.M. N, A review on evaluation metrics for data classification evaluations, Int. J. Data Mining Knowl. Manage. Proc. 5 (2) (2015) 01–11, https://doi.org/10.5121/ijdkp.2015.5201.

[24] J. Brownlee, Imbalanced classification with Python better metrics, balance skewed classes, in: Cost-Sensitive Learning, V1.3, Machine Learning Mastery, 2021 [Online]. Available, https://books.google.hr/books?id=jaXJDwAAQBAJ&printsec=copyright#v=onepage&q&f=false.

[25] D.K. Bhattacharyya, J.K. Kalita, Network anomaly detection, Netw. Anomaly Detect. (3035344211) (2020), https://doi.org/10.1201/b15088.

[26] J. Nabi, "Machine learning —fundamentals." https://towardsdatascience.com/machine-learning-basics-part-1-a36d38c7916.

[27] H. Singh, *Practical machine learning with AWS*. 2021. doi: 10.1007/978-1-4842-6222-1.

[28] J.E. van Engelen, H.H. Hoos, A survey on semi-supervised learning, Mach. Learn. 109 (2) (2020) 373–440, https://doi.org/10.1007/s10994-019-05855-6.

[29] A. Amirkhani, A.Hossein Barshooi, A. Ebrahimi, Enhancing the robustness of visual object tracking via style transfer, Comput. Mater. Continua 70 (1) (2022) 981–997, https://doi.org/10.32604/cmc.2022.019001.

[30] A.H. Barshooi, A. Amirkhani, A novel data augmentation based on Gabor filter and convolutional deep learning for improving the classification of COVID-19 chest X-Ray images, Biomed. Signal. Process. Control 72 (Feb. 2022), 103326, https://doi.org/10.1016/j.bspc.2021.103326.

[31] "Classification: ROC Curve and AUC." https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc (accessed Oct. 05, 2022).

[32] A. Fernández, S. García, M. Galar, R.C. Prati, Learning from Imbalanced Data Sets, springer, 2018.

[33] J. Smith-Perrone, J. Sims, Securing cloud, SDN and large data network environments from emerging DDoS attacks, in: Proceedings of the 7th International Conference Confluence 2017 on Cloud Computing, Data Science and Engineering, 2017, pp. 466–469, https://doi.org/10.1109/CONFLUENCE.2017.7943196.

[34] V. Deepa, K. Muthamil Sudar, P. Deepalakshmi, Detection of DDoS attack on SDN control plane using hybrid machine learning techniques, in: *Proceedings of the International Conference on Smart Systems and Inventive Technology, ICSSIT 2018*, no. Icssit, 2018, pp. 299–303, https://doi.org/10.1109/ICSSIT.2018.8748836.

[35] B. Krawczyk, Learning from imbalanced data: open challenges and future directions, Progr. Artific. Intell. 5 (4) (2016) 221–232, https://doi.org/10.1007/s13748-016-0094-0.

[36] P.C. Sen, M. Hajra, M. Ghosh, Supervised Classification Algorithms in Machine Learning: A Survey and Review, 937, Springer, Singapore, 2020, https://doi.org/10.1007/978-981-13-7403-6_11.

[37] N. Moustafa, J. Slay, UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set, in: 2015 Military Communications and Information Systems Conference, MilCIS 2015 - Proceedings, 2015, https://doi.org/10.1109/MilCIS.2015.7348942.

[38] "Network Management Configuration Guide, Cisco IOS XE Everest 16.6.x (Catalyst 9300 Switches)." https://www.cisco.com/c/en/us/td/docs/switches/lan/cata

lyst9300/software/release/16-6/configuration_guide/nmgmt/b_166_nmgmt_9300_cg/b_166_nmgmt_9300_cg_chapter_0111.html (accessed Jun. 28, 2022).
[39] S. Raschka, V. Mirjalili, Python Machine Learning: Machine Learning and Deep Learning With Python, scikit-learn, and TensorFlow 2, 3rd Edition, 3rd edition, Packt Publishing, 2019.
[40] K.K. Dobbin, R.M. Simon, Optimally splitting cases for training and testing high dimensional classifiers, BMC Med. Genomics 4 (1) (2011) 31, https://doi.org/10.1186/1755-8794-4-31.
[41] J. Brownlee, "Train-test split for evaluating machine learning algorithms." https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/.
[42] Q.H. Nguyen, et al., Influence of data splitting on performance of machine learning models in prediction of shear strength of soil, Math. Probl. Eng. 2021 (Feb. 2021) 1–15, https://doi.org/10.1155/2021/4832864.
[43] A. Rácz, D. Bajusz, K. Héberger, Effect of dataset size and train/test split ratios in QSAR/QSPR multiclass classification, Molecules 26 (4) (Feb. 2021) 1111, https://doi.org/10.3390/molecules26041111.