An Intrusion Detection System for Healthcare Applications using Machine Learning

Miloud Khaldi[1,*], Nadir Mahammed[1], Mohammed Abdrrahim Lahmar[1] and Fadela Djelloul Daouadji[1]

[1]LabRI-SBA Lab, Ecole Superieure en Informatique Sidi Bel Abbes, Algeria

Abstract

The Internet of Things (IoT) is an extension of the current Internet to all objects that can communicate, directly or indirectly, with electronic equipment that is connected to the Internet. IoT offers services in many areas related to human life such as health, transport, home, smart cities, etc. The security of these components and data transfers is a major issue. In the field of healthcare, a medical staff submits requests to the Internet of Medical Things (IoMT) for tasks execution. Intruders can submit false requests to disrupt the operation of these devices. The detection of these attacks requires the development of a reliable security system capable of detecting any intrusion during all phases of the execution process. In this work we propose a supervised Machine Learning based Intrusion Detection System (IDS) for Internet of Medical Things (IoMT), in which we have adopted a Features Selection approach to improve the proposed system performance. This ML-based IDS has been designed to detect suspicious or malicious activities in IoMT, thereby contributing to preventing privacy breaches and security attacks.

Keywords

Intrusion Detection System (IDS), Machine Learning (ML), Features selection, Internet of Medical Things (IoMT), Healthcare

1. Introduction

The Internet of Medical Things (IoMT) [1] is a technology that connects medical devices to the Internet, enabling them to communicate with each other. It has various applications in healthcare, such as remote patient monitoring, medical device management, and healthcare analytics, and remote surgery. The security of IoMT devices and data transfers is a significant concern. Malicious individuals can exploit vulnerabilities in the system to send false requests, leading to system disruptions and compromising network security.

The problem statement of this project is the security challenges associated with IoT components and data transfers in a Cloud-IoT system specifically designed for healthcare applications. In the realm of healthcare, the Internet of Things enables seamless interaction and communication between various objects, facilitating the delivery of diverse services. However, ensuring the security of these components and data transfers is a critical issue. The system is vulnerable to potential threats, as malicious actors can exploit vulnerabilities by submitting false requests, leading to disruptions in system operations and compromising the overall network security.

The purpose of this project is to address the security concerns associated with the Internet of Medical Things (IoMT) in a healthcare environment. The goal is to develop a reliable Intrusion Detection System (IDS) [2] that can detect any intrusion during all phases of the execution process. This system will use Machine Learning techniques to detect any false requests submitted by intruders that could potentially compromise the security of the network and devices. The ultimate objective is to provide a secure and reliable environment for the users to submit their requests without any threat of intrusion. To reach this goal, we introduced a classification model using a combination of two features selection methods and hyper parameters tuning. These methods include Pearson's Correlation Coefficient (PCC), which is one of the filtering methods, and Backward Elimination (BE), which is one of the wrapper methods. We sought to find the optimal threshold to obtain an efficient classification model with high accuracy and a low false alarm rate. In our experiments, we used the NSL-KDD datasets to train and evaluate our model.

The rest of this paper is structured as follows. Section 2 provides an overview of the related work. Section 3 introduces the proposed model. Section 4 describes the dataset used. The preprocessing of the dataset is presented in Section 5. Section 6 details the features selection method applied. In Section 7, the experimental results of our proposed model are presented and discussed. Finally, Section 8 concludes the paper and gives some research perspectives.

6th International Hybrid Conference On Informatics And Applied Mathematics, December 6-7, 2023 Guelma, Algeria

*Corresponding author.

m.khaldi@esi-sba.dz (M. Khaldi); n.mahammed@esi-sba.dz (N. Mahammed); m.lahmar@esi-sba.dz (M. A. Lahmar); f.djellouldaouadji@esi-sba.dz (F. D. Daouadji)

2. Related Work

Many works are being carried out in the context of Intrusion Detection System using Machine learning to find the best parameters and results in terms of performance in various environments, but only a few researches are done particularly for IoMT and healthcare systems. Pandey and Badal [3] proposed a Machine Learning-based Intrusion Detection System for Denial of Service (DoS) attacks. In this work, the training and test datasets were preprocessed by removing irrelevant attack classes such as Probe, User to Root (U2R), and Remote to Local (R2L). Then, 14 new datasets were generated for each combination of features group. Next, Random Tree was used as a binary classifier to train and test the models with the datasets. The instances were classified as either normal or attack. The experimental results of 15 models were compared based on performance metrics. Then, the "best class model" for features selection was chosen based on superior performance compared to the other models. Finally, Correlation-based Feature Selection (CFS), Information Gain (IG), and Gain Ratio (GR) algorithms were applied to the datasets of the "best class model" to perform features selection. The work done by Saheed et al. [4] presents a Machine Learning-based intrusion detection for detecting internet of things network attacks approach. The proposed approach utilizes a combination of feature reduction techniques and ensemble learning algorithms to effectively identify attacks. The researchers employed Principal Component Analysis (PCA) feature selection method. The proposed model was evaluated on UNSW-NB15 dataset. Several machine learning algorithms are trained on the dataset, such as Extreme Gradient Boosting (XGBoost), Cat Boost, K Nearest Neighbor (KNN), Support Vector Machine (SVM), Quadratic Discriminant Analysis and Naïve Bayes. The experimental results showed that the proposed model outperformed other models in terms of accuracy, precision, recall, and F1 score. The XGBoost gave outstanding accuracy reaching 99.99%, precision, F1 score, and MCC compared to other proposed models. In [5] the authors build an Intrusion Detection System (IDS) model based on optimized Machine Learning algorithms. The machine learning algorithms used in this research are KNN, SVM and RF. To improve these algorithms classification accuracy, some parameters of the algorithms are optimized using Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) optimization techniques, while other parameters are used with default values. The result of this experiment shows that optimized KNN, SVM and RF perform better than these algorithms with their default parameter values. Furthermore, the results of the experiment show that KNN is the most suitable algorithm for network anomaly detection regarding detection of known network attacks and unknown network attacks. NSL-KDD standard dataset is used for the experiments of this research. Ten popular Machine Learning (ML) algorithms were evaluated using the NSL-KDD dataset in [6]. These algorithms were ranked based on their performance on various parameters, including specificity, sensitivity, and accuracy. After analyzing the top four performing algorithms, it was found that they consumed a significant amount of time during model building. As a result, feature selection techniques were applied to reduce the time required for intrusion detection without sacrificing accuracy. The experimental results clearly demonstrated the effectiveness of various algorithms with or without feature selection in achieving high accuracy while minimizing the time taken for model building. A study was carried out by the authors [7] to explore the potential of Machine Learning classification algorithms in safeguarding IoT against DoS attacks. The researchers conduct a thorough examination of classifiers that can enhance the development of anomaly-based Intrusion Detection Systems (IDSs). To evaluate the performance of the classifiers, the study employs prominent metrics and validation methods and utilizes well-known datasets, such as CIDDS-001, UNSW-NB15, and NSL-KDD, for benchmarking. Furthermore, the study proposes a methodology for selecting the best classifier based on specific application requirements. The primary objectives of the research are to inspire IoT security researchers to develop IDSs using ensemble learning and suggest appropriate approaches for statistically assessing the classifier's performance. The performance of single classifiers including CART and MLP, and classifier ensembles namely Random Forest (RF), AdaBoost (AB), Extreme Gradient Boosting (XGB), Gradient Boosted Machine (GBM), and Extremely Randomized Trees (ETC) is measured in terms of prominent metrics, i.e., accuracy, specificity, sensitivity, false positive rate, area under the receiver operating characteristic curve. Hyper-tuning of all the classifiers is done using random search. The significant differences of classifiers are statistically assessed using a well-known statistical test. Random Forest outperforms other classifiers in terms of accuracy (94.94%) and specificity (91.6%). Pande et al. [8] provide novel deep learning framework for the detection of attacks. Also, a comparison of machine learning and deep learning algorithms is provided. Findings the obtained results are more than 99% for the NSL-KDD dataset.

3. Proposed Model

In this section, we will present our IDS architecture, describing in detail the datasets used for this work and giving a comprehensive overview of the entire Machine Learning (ML) process. Starting from the initial step of dataset preprocessing, including feature engineering,

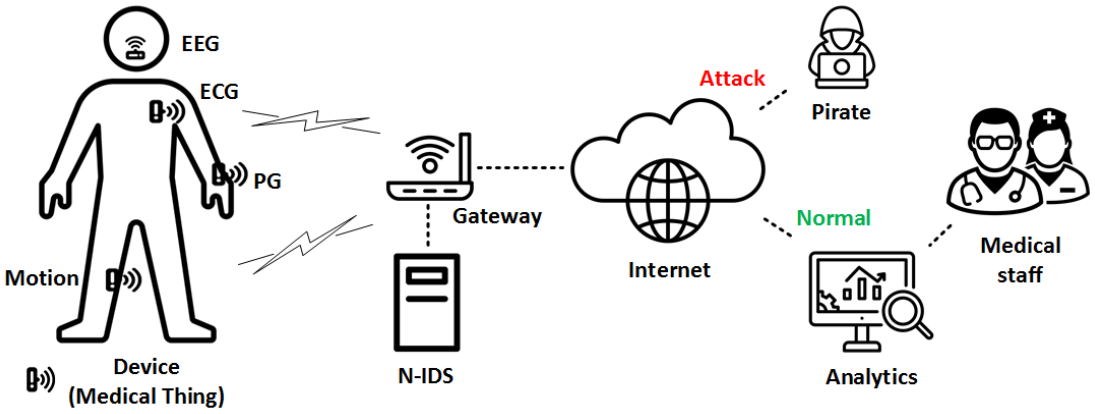

Figure 1: Global architecture.

through to Cross-Validation. Our Intrusion Detection System is of the Network-based IDS (N-IDS) anomaly-based detection type, its role is to detect if there is an attack that wants to damage our devices (medical things) or perturb their operation (see Figure 1).

4. Data-Set

For this study, we opted to use the NSL-KDD dataset [9], which is widely recognized and frequently used in the field of Intrusion Detection Systems (IDS) for networks. This dataset was chosen based on statistics demonstrating its popularity and relevance in the IT security community. NSL-KDD provides comprehensive coverage of attack scenarios and network traffic, making it particularly suitable for our study. NSL-KDD: The NSL-KDD dataset contains real-world network security attacks. it is an improved version of the "KDD cup 99" dataset where all redundant features have been removed NSL-KDD stands for "NSL-KDD Cup 99" where NSL stands for "Network Security Laboratory," indicating its development by researchers from the University of New Brunswick, Canada. The primary purpose of the NSL-KDD dataset is to facilitate the development and evaluation of Intrusion Detection Systems (IDS) and related network security applications. It consists of network traffic data captured from a simulated environment, which emulates various types of attacks and normal activities. The NSL-KDD train dataset consists of 125,973 records and the test dataset contains 22,544 records for 41 features [10]. The NSL-KDD dataset can be utilized in Intrusion Detection Systems (IDS) for the Internet of Medical Things (IoMT) due to several reasons. It serves as a standardized benchmark that enables the evaluation and comparison of different IDS techniques within the IoMT context [6].

4.1. Description of dataset features

The NSL-KDD dataset includes a total of 41 features shown in Table 1. These features are divided into three different types:

- Basic features.
- Content features.
- Traffic features.

4.2. Default classes

The NSL-KDD dataset consists of network traffic data for network intrusion detection. The dataset includes five (05) different classes:

- `Normal class` : Represents normal network connections that are considered legitimate and non-threatening.
- `DoS class` : Denotes Denial-of-Service attacks, where the goal is to disrupt or disable the targeted system or network [11].
- `Probe class` : Represents probing attacks, where an attacker attempts to gather information about the target system or network for potential vulnerabilities [11].
- `R2L class` : Stands for Remote-to-Local attacks, where an unauthorized user tries to gain access to a local system from a remote location [12].
- `U2R class` : Represents User-to-Root attacks, where a local user with limited privileges attempts to escalate their privileges to gain root access [13].

Table 1
The features of the NSL-KDD dataset

№	Feature name
1	duration
2	protocol_type
3	service
4	flag
5	src_bytes
6	dst_bytes
7	land
8	wrong_fragment
9	urgent
10	hot
11	num_failed_logins
12	logged_in
13	num_compromised
14	root_shell
15	su_attempted
16	num_root
17	num_file_creations
18	num_shells
19	num_access_files
20	num_outbound_cmds
21	is_host_login
22	is_guest_login
23	count
24	srv_count
25	serror_rate
26	srv_serror_rate
27	rerror_rate
28	srv_rerror_rate
29	same_srv_rate
30	diff_srv_rate
31	srv_diff_host_rate
32	dst_host_count
33	dst_host_srv_count
34	dst_host_same_srv_rate
35	dst_host_diff_srv_rate
36	dst_host_same_src_port_rate
37	dst_host_srv_diff_host_rate
38	dst_host_serror_rate
39	dst_host_srv_serror_rate
40	dst_host_rerror_rate
41	dst_host_srv_rerror_rate

5. Data Preprocessing

In this section, we will discuss the approach and methods used to preprocess this data.

5.1. Data transformation

The train set and the test includes three categorical features: "`protocol_type`", "`service`" and "`flag`".

To transform these categorical features into numerical features, we used the "`1-N encoding`" method.

Table 2
Dataset structure after data preprocessing

Dataset	Number of records	Normal	Attack
Train	125,973	67,343	58,630
Test	22,544	9,711	12,833

This encoding allows categories to be represented numerically for later use in machine learning algorithms.

5.2. Data normalization

In our study, we opted for "`MinMaxScaler`" normalization, which scale the features values in a range between 0 and 1 for both train and test datasets using the following equation:

$$\frac{x - min(x)}{max(x) - min(x)}. \tag{1}$$

5.3. Binary classification

Binary classification in the context of the NSL-KDD dataset refers to the task of classifying network connections into two distinct categories: "`normal`" and "`attack`", for which we have assigned:

- The value "`0`" to the normal class.
- The value "`1`" to the attack class (DoS, U2R, R2L and Probe).

After the data preprocessing stage, we obtain a new structure of datasets that are shown in Table 2.

6. Features selection

The features selection procedure that we followed is based on the hybridization of filter methods (Pearson's Correlation Coefficient) and wrapper methods (Backward Elimination).

6.1. Filter method: Pearson's Correlation Coefficient (PCC)

As part of the filter method, several criteria can be used to evaluate the relevance of features. The features selection technique used is Pearson's Correlation Coefficient (PCC). After performing multiple tests for correlation, we found that the best result achieved was a correlation coefficient of 0.6. This result was obtained by utilizing a set of 27 features.

The Pearson Correlation Coefficient ρ between two random variables X and Y is given by the following equation:

```
def backward_elimination(XTrain, yTrain, XTest, yTest):
    model = LogisticRegression()
    model.fit(XTrain,yTrain.astype(int))
    initial_acc = model.score(XTest, yTest)
    best_features = XTrain.columns

    while len(best_features) != 20:
        worst_feature = None
        best_acc = initial_acc
        for feature in best_features:
            X_train_subset = XTrain.drop(feature, axis=1)
            X_test_subset = XTest.drop(feature, axis=1)

            model = LogisticRegression()
            model.fit(X_train_subset,yTrain.astype(int))
            acc = model.score(X_test_subset, yTest)

            if acc > best_acc:
                best_acc = acc
                worst_feature = feature

        if worst_feature is not None:
            best_features = best_features.drop(worst_feature)
        else:
            break

    final_features = best_features.tolist()

    return final_features
```

Figure 2: BE Algorithm.

$$\rho = \frac{cov(X,Y)}{\sqrt{\sigma^2(X)\sigma^2(Y)}}. \quad (2)$$

where, cov is the covariance and σ is the variance. The value of ρ lies between -1 and 1, ρ is close to the extreme values -1 and 1 if X and Y are strongly correlated, and ρ = 0 if X and Y are totally uncorrelated. Thus, a feature which is strongly correlated to some other features is a redundant one.

6.2. Wrapper method: Backward Elimination (BE)

After conducting a correlation analysis, we proceeded to employ the Backward Elimination (BE) technique on the initial set of 27 features. Through this iterative process, we refined the feature selection to a final subset of 20 features, resulting in optimal accuracy performance (See Table 3).

Figure 2 shows the pseudo code for the Backward Elimination (BE) method:

These features are considered to be the most important, significant, informative and relevant for the attack prediction model.

Table 3
The list of features selected by PCC and BE

№	Feature name
1	protocol_type
2	service
3	src_bytes
4	dst_bytes
5	land
6	wrong_fragment
7	urgent
8	num_failed_logins
9	num_compromised
10	root_shell
11	num_file_creations
12	num_shells
13	num_access_files
14	is_host_login
15	rerror_rate
16	diff_srv_rate
17	srv_diff_host_rate
18	dst_host_count
19	dst_host_diff_srv_rate
20	dst_host_same_src_port_rate

7. Results and discussion

In this section, we will proceed to evaluate and improve the chosen methods. The experimental results of our dataset were tested for six (06) Machine Learning algorithms: Decision Tree (DT), Random Forest (RF), K-Nearest Neighbor (KNN), Extreme Gradient Boosting (XGBoost), Logistic Regression (LR), Support Vector Machine (SVM), as well as the performance evaluation metrics: Confusion Matrix, Accuracy, Precision, Recall, F1-Score and False Alarm.

7.1. Initial state results

The Table 4 represents the training results of the initial state without normalization and without selection of features for each classifier.

7.2. Improvements

In this section, we will present the performance improvement we've got at each stage for each algorithm using accuracy as performance evaluation metric from the initial state the final state: after normalization, after features selection and after cross validation.

7.2.1. Normalization improvements

After considering normalization, we obtained the improvements shown in Table 5:

Table 4
Initial state learning results

Model	Accuracy	Precision	Recall	F1-Score	False Alarme
DT	76.96	96.32	64.18	77.88	3.23
RF	76.56	96.62	61.14	74.83	2.85
KNN	77.82	97.26	62.84	76.31	2.36
XGBoost	77.10	84.63 80.23	80.14	3.68	
SVM	43.08	81.00	65.48	75.54	18.27
LR	70.28	88.19	55.12	67.82	9.9

Table 5
Normalization improvements

Model	Initial state	After normalization	Improvements
RF	76.56	77.59	1.0335
KNN	76.23	77.31	1.0823
XGBoost	77.10	77.10	0
SVM	43.08	77.31	34.2308
LR	70.28	75.38	5.1011

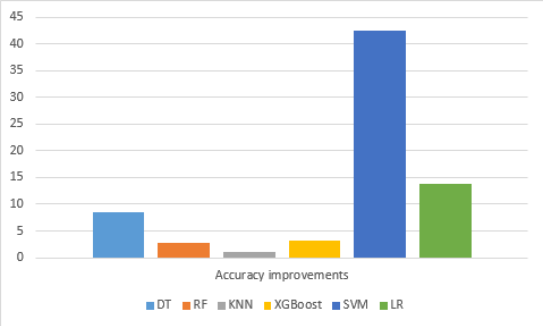

Figure 3: Improved algorithm results between initial state and final state.

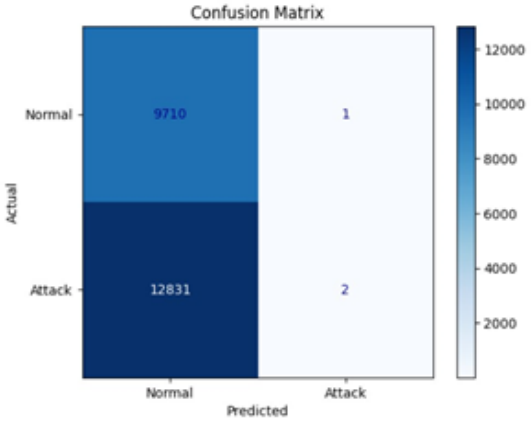

Figure 4: SVM Confusion Matrix for the first state.

7.2.2. Features selection improvements

After selecting best features, we obtained the improvements shown in Table 6:

7.2.3. Cross-Validation improvements

After performing Cross-Validation with 10-fold, we obtained the improvements shown in Table 7:

7.2.4. Total improvements (final state results)

The overall improvement in term of accuracy from the first state to the last state (after Normalization, FS and CV) are shown in Table 8, Figure 3, Figure 4 and Figure 5:

7.2.5. Results discussion

Through the application of learning techniques including Normalization, Features Selection and Cross-Validation, significant improvements in performance (Accuracy, Precision, Recall, F1-Score and False Alarm) were observed comparing to the initial state.

Firstly, data transformation was performed by encoding categorical features using "`1-N encoding`" method, and the "`MinMaxScaler`" function was applied to normalize the data where the Support Vector Machine (SVM) model showcased the most substantial improvement, reflecting a remarkable improvement of 34.2308%.

Next, features selection was carried out using a hybrid approach. The filter method, which involved correlation

Table 6
Features selection improvements

Model	Before FS	After FS (PCC and BE)	Improvements
DT	78.89	80.42	1.5303
RF	77.59	78.94	1.3484
KNN	77.31	76.44	-0.8694
XGBoost	77.10	78.35	1.2508
SVM	77.31	82.85	5.5447
LR	75.38	83.86	8.4767

Table 7
Cross-Validation improvements

Model	Before CV	After CV	Improvements
DT	80.42	85.37	4.9503
RF	78.94	79.44	0.4923
KNN	76.44	77.32	0.8782
XGBoost	78.35	80.24	1.8907
SVM	82.85	85.48	2.6259
LR	83.86	84.00	0.1419

(PCC) analysis, was combined with the wrapper method utilizing Backward Elimination (BE). Through this process, 20 features were selected based on their ability to improve accuracy. The Logistic Regression (LR) model demonstrated notable progress indicating an improvement of 8.4767%.

Finally, Cross-Validation was employed to determine the optimal hyperparameters. This involved dividing the dataset into 10 folds and iteratively training and evaluating the model using different hyperparameter settings. The aim was to identify the hyperparameters that yielded the best performance. The Decision Tree (DT) model exhibited a modest improvement with an increase of 4.9503%.

In terms of overall improvements, the Support Vector Machine (SVM) model showed the most robust improvement, with accuracy rising from 43.08% in the first state to an impressive 85.48% in the last state, representing a remarkable improvement of 42.4015%.

8. Conclusion

In this paper, we explored the use of Machine Learning techniques for intrusion detection in Internet of Medical Things (IoMT). To gain an in-depth understanding of the concepts and mechanisms used in our project, we began by conducting a global study of IoMT, their security issues and the various solutions available in the literature. We have developed an Intrusion Detection System (IDS) intended for IoMT which is based on a learning method based on the features selection using a hybrid approach between a filtering method (PCC) and a wrapper method (BE). During our experiments, we introduced our own features selection technique to the NSL-KDD dataset after an encoding and normalization phase. Our technique has proved extremely effective, and is independent of the classification model used. Using the correlation method (PCC), we identified the features most closely related to the target class. Subsequently, the use of the BE method allowed us to select the most important features among those previously selected by the correlation

Table 8
Total improvements

Model	First state	Final state	Improvements
DT	76.96	85.37	8.4102
RF	76.56	79.44	2.8743
KNN	76.23	77.32	1.0911
XGBoost	77.10	80.24	3.1416
SVM	43.08	85.48	42.4015
LR	70.28	84.00	13.7198

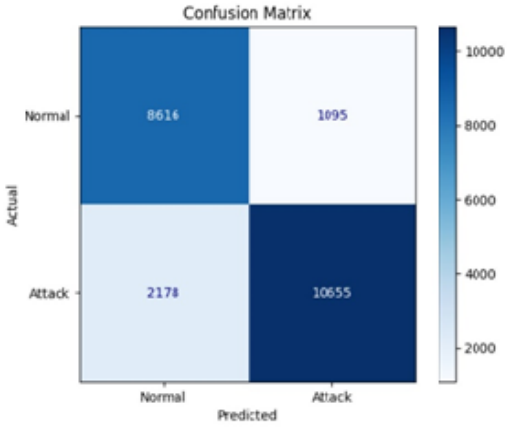

Figure 5: SVM Confusion Matrix for the final state.

method (PCC). This selection of features not only simplified the learning step, but also reduced the complexity of the algorithms and the calculation time required. The results obtained demonstrated the effectiveness of the system we developed in terms of performance, with an accuracy rate of 85.48% for the SVM algorithm. Following on from our work, we propose to further explore trend algorithms, such as Deep Learning, and to broaden the application of features selection techniques by exploring approaches that combine different methods. This would improve the selection of the most relevant features to enhance the performance of the proposed model.

References

[1] B. Lutkevich, A. D. Vecchio, What is the internet of medical things (iomt)?, 2023. URL: https://www.techtarget.com/iotagenda/definition/IoMT-Internet-of-Medical-Things.

[2] F. M. et al, Machine learning for classification analysis of intrusion detection on nsl-kdd dataset, Turkish Journal of Computer and Mathematics Education (TURCOMAT) 12 (2021) 2286–2293.

[3] A. Pandey, N. Badal, Machine learning based intrusion detection system for denial of service attack, Computational Methodologies for Electrical and Electronics Engineers (2021) 29–47.

[4] Y. K. S. et al, A machine learning-based intrusion detection for detecting internet of things network attacks, Alexandria Engineering Journal 61 (2022) 9395–9409.

[5] T. Khorram, Network intrusion detection using optimized machine learning algorithms, European Journal of Science and Technology (2021).

[6] H. Malhotra, P. Sharma, Intrusion detection using machine learning and feature selection, International Journal of Computer Network and Information Security 11 (2019) 43–52.

[7] A.Verma, V. Ranga, Machine learning based intrusion detection systems for iot applications, Wireless Personal Communications 111 (2019) 2287–2310.

[8] S. P. et al, An intrusion detection system for health-care system using machine and deep learning, World Journal of Engineering 19 (2022) 166–174.

[9] unb.ca, Nsl-kdd dataset, 2023. URL: https://www.unb.ca/cic/datasets/nsl.html.

[10] A. K. et al, Survey of intrusion detection systems : techniques, datasets and challenges, Cybersecurity 2 (2019) 1–21.

[11] S. Choudhary, N. Kesswani, Analysis of kdd-cup'99, nsl-kdd and unsw-nb15 datasets using deep learning, iot. Procedia Computer Science 167 (2020) 1561–1573.

[12] S. F. et al, A machine learning-based lightweight intrusion detection system for the internet of things, Rev. d'Intelligence Artif. 33 (2019) 203–21.

[13] S. Paliwal, R. Gupta, Denial of-service, probing remote to user (r2l) attack detection using genetic algorithm, International Journal of Computer Applications 60 (2012) 57–62.