

**REPUBLIC OF TÜRKİYE**

**HALİÇ UNIVERSITY**

**INSTITUTE FOR GRADUATE STUDIES**

**DEPARTMENT OF SOFTWARE ENGINEERING PROGRAMME**

**Machine Learning Approach for Intrusion Detection System In IoT Environment**

**GRADUATION PROJECT II**

**By**

**MOHAMMED YASSER AHMED**

**Assoc.Prof.**

**MELEK UYGUN**

**ISTANBUL**

**JUN / 2025**

**PREFACE**

If we look at the role of technology in our life, it is quite clear that in this era it is a necessary and dynamically changing power, useful in various fields and open to regular upgradations. Technology has been a cornerstone in making the life of human beings easier and addressing the fundamental needs. Nowadays, whenever we open the newspaper, switch on the television, or technology periodicals, we usually come across columns or news reports that surprise us with their novelty.

As we are nearing the completion of our undergraduate degrees, we are eager to apply our knowledge, experience, and skills we have acquired throughout our degrees. Keeping this in view, we wish our work not only to be our individual success but also one the university can be proud of.

We wish that the future as Software Engineers would be filled with immense opportunities for us. We wish that the work we have done and will do in the future is useful in making mankind's life simpler and bring our country to the peaks of modern civilization.

We would like to thank our instructor, Prof. Mrs Melek Uygun, for his support, patience, and guidance during our project. Our special thanks are also to our other instructors, colleagues, families, and managers who have made significant contributions to this work.

JUN, 2025 Mohammed Yasser

**TABLE OF CONTENT**

**PREFACE…………………………………………………………………………** **I**

**TABLE OF CONTENT ………………………………………………………... II**

**ABREVIATIONS………………………………………………………………. III**

**LIST OF TABLES……………………………………………………………… IV**

**LIST OF FIGURES…………………………………………………………….. V**

**ÖZET…………………………………………………………………………….. VI**

**ABSTRACT…………………………………………………………………….. VII**

[**1. INTRODUCTION……………………………………………………………...**](#_heading=h.1egqt2p)  [1](#_heading=h.1egqt2p)3

**2. DATASET DESCREPTION……………………………………………………** 14

2.1 DATASET SAMPLE **…………………………………………………………..** 16

**3. PREPROCESSING AND FEATURE ENGINEERING ……………………** 17

**4.MODEL IMPLEMENTATION…………………………………………….. ...** 18

4.1 RANDOM FOREST CLASSIFIER ……..……………………………………..18

4.2 LINEAR CLASSIFIER…………………..…………………………………… 19

4.3 K-NEAREST NEIGHBORS ……...……..…………………………………… 21

**5. RESULTS AND DISCUSSION………………………………………………** 22

5.1 Experimental setup/implementation details…………………………………….26

5.2 Evaluation metrics……………………………………………………………... 27

5.3 Performance metrics…………………………………………………………… 28

**6. RELATED WORKS……………………………………………………………** 29

**7. CONCLUSION…………………………………………………………………** 31

[**8. RESOURCES…………………………………………………………………..**](#_heading=h.2dlolyb)  32

**ABBREVIATIONS**

IoT : Internet of Things

IIoT : Industrial Internet of Things

IDS : Intrusion Detection System

J48 : C4.5 Decision Tree Algorithm Implementation

MQTT : Message Queuing Telemetry Transfer

PCA : Principal Component Analysis

TSODE : Transient Search Optimization with Differential Evolution

CNN : Convolutional Neural Network

GA : Genetic Algorithm

RF : Random Forest

CSForest : Cost-Sensitive Forest

IBK : Instance-Based k-Nearest Neighbors

MAE : Mean Absolute Error

TP : True Positive

TN : True Negative

FP : False Positive

FN : False Negative

F1-Score : Harmonic Mean of Precision and Recall

AUC : Area Under the Curve

MCC : Matthews Correlation Coefficient

SDN : Software-Defined Networking

CVR : ClassificationViaRegression

WEKA : Waikato Environment for Knowledge Analysis

DL4 : Deep Learning with Four Layers

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
|  |  | **Page Number:** |
| Table 1.1: | Dataset Sample………………………………. ………… | 16 |

|  |  |  |
| --- | --- | --- |
| Table 1.2: | Models Comparison…………………………. ………… | 23 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
|  |  | **Page Number:** |
| Figure 2.1: | Model Evaluation .Metrices.…………………. | 24 |
| Figure 2.2: | RF Confusion Matrix.……………………………. | 25 |
| Figure 2.3: | SGD Confusion Matrix……………………………. | 25 |
| Figure 2.4: | KNN Confusion Matrix..……………………………. | 26 |

**GENERAL INFORMATION**

Name and Surname : Mohammed Yasser Ahmed

Department : Software Engineering

Project Consultant : MELEK UYGUN

Project Date : JUN-2025

**Machine Learning Based Solution for Intrusion Detection System in IoT Environment**

**ÖZET**

Bu çalışmada, KDDCup 99 10% veri seti kullanılarak makine öğrenmesine dayalı bir Saldırı Tespit Sistemi (IDS) geliştirilmiştir. Çalışmada, ağ trafiğinin normal veya saldırı olarak sınıflandırılması amacıyla veri ön işleme adımları uygulanmış, ardından üç denetimli makine öğrenme modeli olan Random Forest (RF) , K-En Yakın Komşu (KNN) ve Doğrusal Sınıflandırıcı kullanılmıştır. Her model, ağ saldırılarını tespit etme yeteneklerini değerlendirmek amacıyla çeşitli performans metrikleri ile analiz edilmiştir. Sonuçlar, kullanılan yöntemlerin doğruluğunu ve pratik uygulamalar açısından uygunluğunu ortaya koymakta ve literatürdeki ilgili çalışmalarla karşılaştırmalar yapılmasına olanak tanımaktadır.

**Anahtar Kelimeler—Saldırı Tespit Sistemi, Makine Öğrenmesi, KDDCup99, Random Forest, K-En Yakın Komşu, Doğrusal Sınıflandırıcı, Ağ Güvenliği**

**GENERAL INFORMATION**

Name and Surname : Mohammed Yasser Ahmed

Department : Software Engineering

Supervisor : MELEK UYGUN

Project Date : January-2025

**Machine Learning Based Solution for Intrusion Detection System in IoT Environment**

**ABSTRACT**

Here we propose a machine learning-based Intrusion Detection System (IDS) on the KDDCup 99 10% dataset. The proposed system applies the required preprocessing techniques, including encoding and feature scaling, and then applies three supervised classification models: Random Forest (RF), K-Nearest Neighbors (KNN), and Linear Classifier (LC). All the models are compared with traditional performance metrics and graphical analysis metrics to establish how effective the models are with regard to network intrusion detection. The results are compared with those of similar studies to demonstrate the real-world advantage of methodologies adopted in IDS development.

**Keywords—Intrusion Detection System, Machine Learning, KDDCup99, Random Forest, K-Nearest Neighbors, Linear Classifier, Network Security**

**1.INTRODUCTION**

The rapid expansion of computer networks and the internet has intensified the frequency and sophistication of cyber attacks. Intrusion Detection Systems (IDS) have become indispensable components of modern-day cybersecurity systems to identify and track malicious traffic on a network. Traditional IDS techniques rely on manually created rules or signature-based detection, which are unable to keep up with the pace of new attack patterns.

Machine learning offers a better solution by enabling IDS to learn from historical data and identify malicious behavior based on patterns rather than pre-established signatures. For this purpose, supervised learning models have proven to be of immense potential in classifying normal and malicious network traffic with high accuracy, thereby being highly effective for anomaly and misuse detection.

In this study, we attempt to model and evaluate a machine learning IDS on the KDDCup 99 10% dataset, a standard benchmark for network intrusion detection studies. We perform requisite preprocessing steps on the data and train three supervised learning classifiers, namely Random Forest, K-Nearest Neighbors (KNN), and Linear Classifier. The classifiers are tested and trained to evaluate their performance on network intrusion detection.

The main objective of this study is to assess the classification accuracy of the models based on significant metrics and visualization methods and, furthermore, to compare our results with the existing literature in the field. The results provide insight into the application of supervised learning models in formulating stable and efficient IDS solutions.

REFERENCES [1] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," IEEE Network, vol. 8, no. 3, pp. 26–41, May/Jun. 1994, doi: 10.1109/65.283931.[2] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in Proc. IEEE Symp. Security and Privacy, May 2010, pp. 305–316, doi: 10.1109/SP.2010.25.

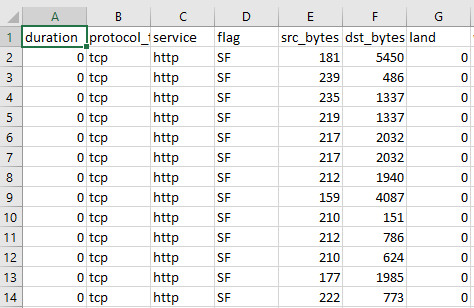
**2. DATASET DESCRIPTION**

This study employs the KDDCup 99 data set, a recognized data set commonly applied for the evaluation of intrusion detection techniques. It was originally obtained from the 1998 DARPA Intrusion Detection Evaluation Program and contains a wide range of simulated network traffic information, including both normal connections and attack connections. It has 41 features for every record, which constitutes a single network connection, defining the duration, protocol type, service, and traffic-based features, among other connection features, with a class label indicating whether the connection is normal or if it is of a certain type of attack.

Due to the extremely large size of the original dataset consisting of millions of records, a 10% stratified sample was taken for efficient model training and testing. This subsample was taken by sampling 10% of the whole dataset randomly with a fixed random seed for reproducibility purposes. The sampled dataset was saved to a new CSV file and acted as the primary working dataset throughout this research. The 10% subsample provides an acceptable size for experimentation while not significantly compromising the heterogeneity of attack types and normal traffic.

The selected sample retains the original form with all 41 features and the class label and was further preprocessed for use in training and testing of the machine learning models discussed in subsequent sections.

**2.1 DATASET SAMPLE**



**Table 1.1** Dataset Sample

**3. PREPROCESSING AND FEATURE ENGINEERING**

Before the use of machine learning models, the dataset underwent prelim processing and feature engineering processes in order to attain data integrity and model readiness. The prelim processes play a crucial role in preparing the raw network traffic data for effective classification.

First, three categorical features—protocol\_type, service, and flag—were transformed to numeric form using label encoding. This way, the machine learning models could accept the categorical values in numeric form while retaining the differences between the categories.

Next, the target class label was converted into binary form to make the problem of classification easier. The multi-class label representing specific types of attacks or normal traffic was converted into binary form: 0 for normal traffic and 1 for all types of attacks. This conversion restates the problem as a binary classification problem to detect the presence of intrusions.

Following the encoding of the labels, the feature values were all normalized with Min-Max scaling. Min-Max scaling is a technique that transforms all the numerical features to a common range of 0 to 1, which improves the performance of the model by avoiding features with a larger numeric range from dominating the learning algorithm.

Lastly, the data set was checked for null values and had no missing data. The preprocessed data set consisted of 41 input features with one binary label column and was ready to be trained with the selected machine learning models.

1. **MODEL IMPLEMENTATION**

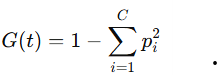
Here, a step-by-step explanation of the three supervised machine learning algorithms employed in this study is provided: Random Forest (RF), Linear Classifier (SGD), and K-Nearest Neighbors (KNN). Each of the algorithms is selected according to its distinct properties and suitability to be used for the binary classification task of network intrusion detection. The algorithms were implemented with the help of the Scikit-learn library and trained with the preprocessed KDDCup 99 10% dataset.

**4.1. RANDOM FOREST CLASSIFIER**

Random Forest is an ensemble learner that generates many decision trees during training and predicts the class which is the mode of the classes of the trees. It is part of the family of bagging techniques, which attempt to reduce individual model variance by averaging their predictions.

Each tree is trained on a random subset of the training data (with replacement), and at each node, a random subset of features is selected for splitting. This injects randomness, which encourages diversity in the models as well as reduces the risk of overfitting.

One type of decision tree splits the feature space based on quantities such as Gini impurity or information gain. Gini impurity is a popular option and is defined as:



Where:

G (t) is the Gini impurity at node t

Pi is equal to the ratio of class i instances at node t ,

C stands for the number of classes.

Random Forest averages the predictions of all the trees:





**Advantages for (RF) :**

* Handles large feature spaces efficiently.
* Robust to noise and overfitting.
* High accuracy and stability on imbalanced data.

**4.2 Linear Classifier (SGD Classifier)**

The Linear Classifier in this study is implemented using **Stochastic Gradient Descent (SGD)**, a simple yet powerful optimization algorithm used to minimize convex loss functions. The model attempts to find a linear decision boundary between the two classes by minimizing some loss function of a kind such as hinge loss (SVMs) or logistic loss (logistic regression).  
  
The decision function is:

Where :  
  
W is the weight vector.  
  
X is the input feature vector.  
  
B is the bias term.

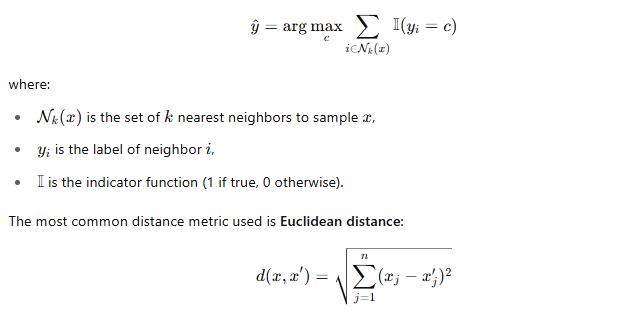
**Advantages for (SGD) :**

* Extremely fast and scalable for large datasets.
* Performs well with linearly separable data.
* Lightweight and efficient for real-time detection.

**4.3 K-Nearest Neighbors (KNN)**

K-Nearest Neighbors is a non-parametric, instance-based algorithm. It does not directly train a model from the training data but only stores all the training instances in memory. When it needs to make a prediction, it calculates the distance of the test sample from every point in the training set and selects the k nearest neighbors to create a response.

The predicted class *y^* is determined by **majority vote**:

where *n* is the number of features.

**Advantages for IDS:**

* Simple and effective for well-scaled data.
* Captures local decision boundaries.
* No model training time.

**5.RESULTS AND DESCUSTION**

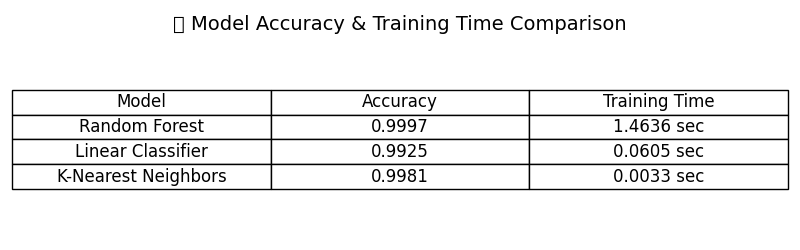
For the performance evaluation of models deployed, the preprocessed KDDCup 99 10% dataset was split into training and test subsets according to the conventional 80/20 ratio. The training subset was used to train the three supervised classifiers—Random Forest, Linear Classifier, and K-Nearest Neighbors—and they were tested on the test subset using the conventional classification measures: accuracy, precision, recall, F1-score, and ROC-AUC.

Confusion matrices were graphed so the performance of every model's classification could be seen, i.e., the amount of true positives, true negatives, false positives, and false negatives. Bar charts were graphed comparing per-class performance metrics, which provided a more specific breakdown of the performance of every model with normal versus attack traffic.

The measured **accuracy scores** for each model on the test set were as follows:

* **Random Forest**: 99.97%
* **Linear Classifier (SGD)**: 99.25%
* **K-Nearest Neighbors**: 99.81%

To provide a broader performance perspective, a comparative table was constructed, summarizing both the **training time** and **classification accuracy** of each model. This table highlights the trade-offs between computational efficiency and predictive performance.

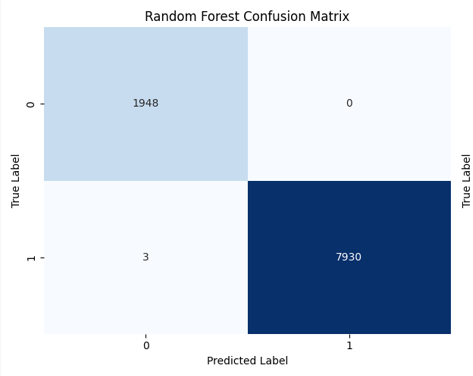


**Table1.2** Model Comparison

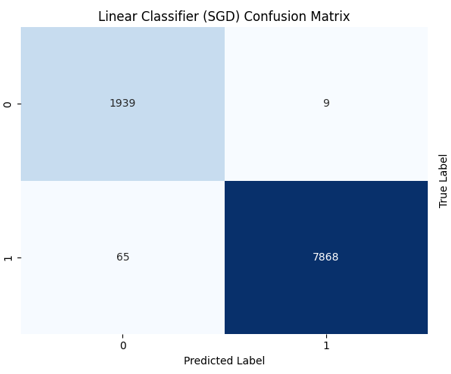
Although all three models performed well, the Random Forest classifier performed best overall in all the measures. It performed best in accuracy-robustness trade-off, and best suited for deployment within real-world IDS. The Linear Classifier was less accurate but quickest to train and computationally efficient. K-Nearest Neighbors was extremely accurate but at the expense of higher computation at prediction time as it is an instance-based learner.



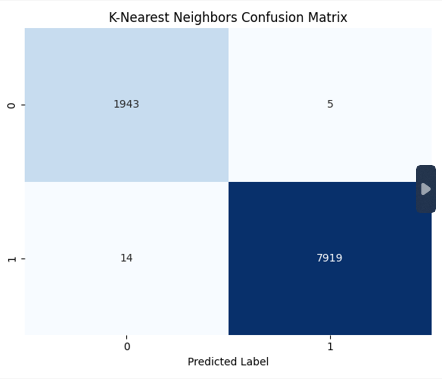
**Figure 2.1** Models Metrices



**Figure 2.2** RF Confusion Matrix



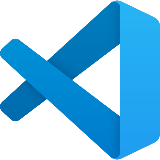
**Figure 2.3** SGD Confusion Matrix



**Figure 2.4** KNN Confusion Matrix

**5.1 Experiment setup/Hardware details**

To create and execute the project, I used Visual Studio Code (VS Code) as the primary Integrated Development Environment (IDE). VS Code provided a convenient and productive platform for coding, testing, and debugging the code throughout the project. The experiment system was equipped with an Intel Core i7 6th-gen processor, which ensured a good amount of computational power for running the machine learning models and working on large data. The system also featured 8GB of memory, assisting in memory management effectively during manipulation of data and model training. In addition, the system featured a GTX 950 Graphics Processing Unit (GPU), assisting in speeding up some of the machine learning operations although the primary focus of this project was on CPU computation. This combination provided a good balance of resources to effectively execute the project tasks without significant performance bottlenecks.



****

**5.2 Evaluation metrics**

We employed four evaluation metrics: Accuracy, Precision, Recall and F1-Score. They are calculated based on the confusion matrix created by the following quantities:

True positive (TP): Amount of attack rows that were correctly identified as attack.

True negative (TN): Number of normal rows correctly identified as normal.

False positive (FP): The number of regular rows incorrectly detected attack.

False negative (FN): Number of attack rows misclassified as normal.

**5.3 Performance metrics**

*Accuracy*

We used this measure to calculate the ratio of how accurately the algorithms have correctly classified “Normal” and “Attack” rows.

A black text on a white background

Description automatically generated

*Precision*

We used this metric to calculate the ratio of actual attacks that were correctly classified.

A black text on a white background

Description automatically generated

*Recall*

We used this measure to calculate the percentage of predicted attacks to the total number of attacks.

A mathematical equation with black text

Description automatically generated

*F1-score*

We used this measure to calculate a better performance evaluation of each ML model.

A black text with a white background

Description automatically generated

**6. RELATED WORKS**

Previous work the KDDCup 99 dataset using the Naive Bayes algorithm reported an accuracy of only **64.02%**, with low precision (**50.27%**) despite high recall. In contrast, our supervised models—Random Forest, Linear Classifier, and KNN—achieved significantly higher accuracy, all above **99%**. This improvement is due to the use of more robust models, proper feature scaling, and effective preprocessing. Unlike Naive Bayes, our models handle feature interactions and imbalanced data more effectively, resulting in better overall detection performance.

<https://iopscience.iop.org/article/10.1088/1742-6596/1810/1/012013/pdf> (2020)Publish Year.

Another paper reported **92.8% accuracy** using Logistic Regression, and up to **97.01%** with deep learning models such as CNN+LSTM and ANN. In contrast, our simpler supervised models—Random Forest, KNN, and Linear Classifier—achieved **over 99% accuracy** without deep architectures. This demonstrates that with proper preprocessing and feature engineering, traditional models can outperform both basic and complex approaches in intrusion detection.

<file:///C:/Users/PROMISE/Downloads/sensors-23-00206.pdf> (2023) Publish Year.

A recent study applied the Random Forest algorithm to the KDDCup 99 dataset and achieved an accuracy of **98.71%**. Their approach involved One-Hot Encoding for categorical features, normalization using Standard Scaler, and feature selection based on Pearson correlation values above 0.5. They also combined the original and encoded labels before training.

In contrast, our method used **Label Encoding** for categorical features, **MinMaxScaler** for normalization, and retained all features without correlation-based filtering. Despite a simpler pipeline, our Random Forest model achieved **higher accuracy**, exceeding **99%**, suggesting that minimal but targeted preprocessing with full feature utilization can lead to improved detection performance.

**7. CONCLUSION**

We designed and evaluated a machine learning-driven Intrusion Detection System (IDS) for this work based on the KDDCup 99 10% data. We were interested in applying supervised classification models—Random Forest, Linear Classifier (SGD), and K-Nearest Neighbors—to classify traffic as normal or malicious. We designed a streamlined preprocessing pipeline of categorical feature coding by Label Encoding, conversion of labels to binary, and MinMax normalization, such that the data was clean, normalized, and the same for all models.

All models were trained and tested on standard performance metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. Random Forest and KNN had the best accuracy (both more than 99%), whereas Linear Classifier also stood out with fast training and almost flawless classification statistics. These results validate the correctness of supervised learning models paired with appropriate data preparation.

Less complex than other modern related work, our approach was more precise. An earlier work based on Random Forest with One-Hot Encoding preprocessing, feature selection, and Standard Scaler had 98.71% accuracy. Our approach, which possessed the full set of features and less complex preprocessing, had superior performance. This indicates that we can avoid deep pruning of features or awkward encoding sometimes to get the best performance.

Lastly, our work indicates that traditional machine learning models, correctly deployed, can provide highly scalable and efficient solutions for intrusion detection. In terms of future work, ensemble methods, data streams in real time, or testing on more recently produced IDS benchmark datasets can be employed to increase reactivity and generalization against modern threats.

**8.RESOURCES**

[1]Al-Daweri, M.S.; Zainol Ariffin, K.A.; Abdullah, S.; Md. Senan, M.F.E. An Analysis of the KDD99 and UNSW-NB15 Datasets for the Intrusion Detection System. Symmetry 2020, 12, 1666.

[2] Kumar, V., Das, A.K. & Sinha, D. UIDS: a unified intrusion detection system for IoT environment. Evol. Intel. 14, 47–59 (2021)

[3]Abdelmoumin, G., Rawat, D.B., Rahman, A., 2022. On the Performance of Machine Learning Models for Anomaly-Based Intelligent Intrusion Detection Systems for the Internet of Things. IEEE Internet of Things Journal 9, 4280–4290.

[4]Salman, Emad Hmood, et al. "An anomaly intrusion detection for high-density internet of things wireless communication network based deep learning algorithms." Sensors 23.1 (2022): 206.

[5] Alani, M.M., Awad, A.I., 2023. An Intelligent Two-Layer Intrusion Detection System for the Internet of Things. IEEE Transactions on Industrial Informatics 19, 683–692.

[3] Albulayhi, K., Abu Al-Haija, Q., Alsuhibany, S.A., Jillepalli, A.A., Ashrafuzzaman, M., Sheldon, F.T., 2022. IoT Intrusion Detection Using Machine Learning with a Novel High Performing Feature Selection Method. Applied Sciences 12, 5015. https://doi.org/10.3390/app12105015

[4] Fatani, A., Abd Elaziz, M., Dahou, A., Al-Qaness, M.A.A., Lu, S., 2021. IoT Intrusion Detection System Using Deep Learning and Enhanced Transient Search Optimization. IEEE Access 9, 123448–123464. https://doi.org/10.1109/ACCESS.2021.3109081

[5] Guo, G., 2021. A Machine Learning Framework for Intrusion Detection System in IoT Networks Using an Ensemble Feature Selection Method, in: 2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). Presented at the 2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 0593–0599. https://doi.org/10.1109/IEMCON53756.2021.9623082

[6] Kayode Saheed, Y., Idris Abiodun, A., Misra, S., Kristiansen Holone, M., Colomo-Palacios, R., 2022. A machine learning-based intrusion detection for detecting internet of things network attacks. Alexandria Engineering Journal 61, 9395–9409. https://doi.org/10.1016/j.aej.2022.02.063

[7] Ravi, V., Chaganti, R., Alazab, M., 2022. Deep Learning Feature Fusion Approach for an Intrusion Detection System in SDN-Based IoT Networks. IEEE Internet of Things Magazine 5, 24–29. https://doi.org/10.1109/IOTM.003.2200001

[8] Saba, T., Sadad, T., Rehman, A., Mehmood, Z., Javaid, Q., 2021. Intrusion Detection System Through Advance Machine Learning for the Internet of Things Networks. IT Professional 23, 58–64. https://doi.org/10.1109/MITP.2020.2992710