






Real-Time Intelligent Anomaly Detection and Prevention System

Remzi Gürfidan¹ , Şerafettin Atmaca² , Tuncay Yiğit³ 

¹ Isparta University of Applied Science, Yalvac Technical Sciences Vocational School, Isparta, Türkiye

² Isparta University of Applied Sciences, Isparta, Türkiye

³ Süleyman Demirel University, Computer Engineering Department, Isparta, Türkiye



Corresponding author:

Remzi Gürfidan, Isparta University of Applied Science, Yalvac Technical Sciences Vocational School, Isparta, Türkiye

E-mail address:

remzigurfidan@isparta.edu.tr

Submitted: 12 May 2023

Revision Requested: 06 September 2023

Last Revision Received: 21 September 2023

Accepted: 24 September 2023

Published Online: 28 September 2023

Citation: Gürfidan R. Atmaca Ş. Yiğit T.(2023). Real-Time Intelligent Anomaly Detection and Prevention System. *Sakarya University Journal of Computer and Information Sciences*. 6 (3) <https://doi.org/10.35377/saucis...1296210>

ABSTRACT

Real-time anomaly detection in network traffic is a method that detects unexpected and anomalous behavior by identifying normal behavior and statistical patterns in network traffic data. This method is used to detect potential attacks or other anomalous conditions in network traffic. Real-time anomaly detection uses different algorithms to detect abnormal activities in network traffic. These include statistical methods, machine learning, and deep learning techniques. By learning the normal behavior of network traffic, these methods can detect unexpected and anomalous situations. Attackers use various techniques to mimic normal patterns in network traffic, making it difficult to detect. Real-time anomaly detection allows network administrators to detect attacks faster and respond more effectively. Real-time anomaly detection can improve network performance by detecting abnormal conditions in network traffic. Abnormal traffic can overuse the network's resources and cause the network to slow down. Real-time anomaly detection detects abnormal traffic conditions, allowing network resources to be used more effectively. In this study, blockchain technology and machine learning algorithms are combined to propose a real-time prevention model that can detect anomalies in network traffic.

Keywords: Anomaly behavior detection, intrusion detection, machine learning, blockchain

1. Introduction

The detection of an outlier that is outside the normal value that may occur in a business process is called anomaly detection. In anomaly cases, unusual or unique patterns may occur in the dataset that deviate from the expected values of the predicted behavior. Anomaly detection is a serious problem in many different fields, including cybersecurity, manufacturing problem detection, and fraud detection in the financial sector. Statistical-based methods and machine learning-based methods are the two main detection techniques for anomaly detection. While statistical methods use variables such as mean and standard deviation, machine learning-based approaches use supervised or unsupervised learning methods to identify spam. Spam refers to electronic messages sent via electronic mail or cell phone messages, sometimes individually and sometimes collectively, without the consent of the users, harassing them. Spam messages can harass users in many different categories. Figure 1 shows an image in which spam messages are classified.

In order to distinguish spam e-mails from others, it is useful to know some tips. These clues can be very useful in the preliminary diagnostic process to help users differentiate between spam e-mails and real ones. Figure 2 shows some of the clues that can be used to identify spam e-mails.

Spam emails and messages put users in a very difficult situation because they slow down routine workflow, bloat the inbox, and pose a security risk by exposing them to phishing scams or malicious links. Anomalies and spam attempts are all caused



by cyber-attacks. Although their strength and effectiveness vary depending on the nature, the main purpose of cyber-attacks is to compromise user security and exploit security vulnerabilities.

SPAM MESSAGES	Advertising messages promoting products or services
	Phishing scams for revealing sensitive information
	Malicious messages with links to viruses or other malware
	Chain letters or other types of pyramid schemes
	Messages of political or religious content that you did not request

Figure 1 Spam message categories

TIPS USED IN SUSPECTING	A misleading or irrelevant subject line
	Sender you don't know
	An unfamiliar email address
	A general greeting such as "Dear friend" or "Hello"
	Poor grammar or spelling mistakes
	Personal or sensitive information requests

Figure 2 Tips used in Suspecting spam

Cyber-attacks on web pages are carried out to gain unauthorized access to the pages, to obtain users' sensitive information, or to disrupt the normal functioning of the web page. Some cyber-attack actions targeting web pages are shown in Figure 3.

CYBER ATTACK METHODS	SQL Injection: A method involving injecting malicious code by accessing the database of a web application containing a database.
	Cross-Site Scripting (XSS): by injecting malicious code into a web application, it is executed in the victim user's browser by navigating to the compromised site.
	Distributed Denial of Service (DDoS): aims to keep a heavy-traffic web application busy, making it inaccessible to users.
	Phishing: aims to create a fake website or email that appears to come from a trusted source to trick users into revealing sensitive information such as passwords or credit card numbers.
	Malware: aims to infect a web application with malware such as viruses or Trojan horses that can compromise the security of the site and the computers of its visitors.

Figure 3 Cyber-attack methods

In cases where cyber-attacks on websites are successful, there are serious consequences such as theft of sensitive information of users, disruption of commercial activities of companies, and damage to the reputation of organizations. To stay safe from such attacks, it is vital to regularly update and maintain your website's security software and implement robust security measures such as intrusion detection systems and encryption.

The motivation and salient features of this work are listed below.

- An artificial intelligence intelligence-based model is proposed to detect real-time network anomalies.
- Six different machine learning models are trained for the proposed model and the training results are presented with different metrics and the most successful one is selected.
- The situations that cause anomalies are collected in a secure and transparent blacklist structure thanks to the blockchain structure.
- A smart contract is prepared to manage the registration process to the blockchain structure.
- The performances of all transactions are meticulously measured and tested for real-time operation.

2. Related Works

Walling and Lodh developed a univariate selection-based IDS model that can be applied with machine learning algorithms such as decision trees, kNN, SVM, and logistic regression. The developed IDS model was applied on the NSL-KDD dataset and performance improvements were demonstrated [1]. Sreenivasula and Sathya presented a NIDS model based on machine

learning methods that can detect and prevent various types of attacks. The NSL-KDD dataset was used to measure the classification performance of various ML classifiers based on different attributes. It was shown that the developed NIDS model achieved better results than existing single ML methods [2]. Aktar and Nur presented a new model for deep learning learning-based intrusion detection focusing on DoS and DDoS attacks. The performance of the proposed model is evaluated using three different datasets (CIC-DDoS2019, CIC-IDS2017, and NSL-KDD). The developed model has shown that it can achieve up to 97.58% accuracy in anomaly detection in the system [3]. In their study, Özalp and Albayrak, unlike other studies in the literature, examined the effect of the weights of the attributes in the dataset on the detection of cyber attacks on computer networks using the NSL-KDD dataset [4]. Fernandes et al. conducted a comprehensive research study on related techniques, systems, and analysis for the detection of network anomalies. They analyzed anomaly detection under five categories: categories of intrusion detection systems, network traffic anomalies, detection methods and systems, network data types, and open issues [5]. In their study, Dutta et al. used Deep Neural Network (DNN) and Long Short Term Memory (LSTM) deep models in combination with a meta-classifier (logistic regression) following the principle of mass generalization. The proposed approach is twofold. In the first step, a DSAE is used for data preprocessing and feature engineering. In the second step, a stacking ensemble learning approach is used for classification. The effectiveness of the method is evaluated on various datasets including IoT-23, LITNET-2020, and NetML-2020 collected in an IoT environment [6]. The methodology presented by Hawawreh and Rawashdeh proposes an approach to detect the presence of anomalies in the hypervisor layer. This approach is designed to deter Distributed Denial of Service (DDoS) activities between virtual machines. The proposed method for the detection and classification of traffic between virtual machines is executed through an evolutionary neural network. This network seamlessly combines particle swarm optimization with neural network to achieve its goal. The approach to detect and categorize DDoS attacks in a cloud environment detects and classifies DDoS attacks with a high success rate [7]. Hoque et al. proposed a real-time approach to detect DDoS attacks using an innovative correlation metric. The effectiveness of the technique is evaluated using three different network datasets, namely CAIDA DDoS 2007, MIT DARPA, and TUIDS. In addition, the proposed technique is executed on FPGA to evaluate its effectiveness. The detection accuracy of this method is extremely high and the FPGA implementation of this process can identify the attack in less than a microsecond [8]. Gurina and Eliseev investigate the detection of network attacks targeting web servers. The study focuses on two common types of attacks, "denial of service" and "code injection". Multiple techniques for detecting attacks are evaluated. A novel approach based on the recognition of the dynamic response of the web server during request processing is proposed to detect attacks as anomalies. After implementing the detection algorithm, its effectiveness is measured and the advantages and disadvantages of the proposed methodology are evaluated [9]. Alsamiri and Alsubhi aimed to contribute to the existing literature by evaluating various machine learning algorithms that can quickly and efficiently identify network attacks targeting IoT devices. Various detection algorithms were evaluated using a newly created dataset called Bot-IoT. In the implementation phase, seven different machine learning algorithms were used, most of which exhibited high performance. After the implementation of the Bot-IoT dataset, new features were derived and compared with previous research studies. The comparison revealed better results showing the superiority of the new features [10].

3. Method

The main purpose of this study is to detect attacks such as Probe, DoS, R2L, and U2R and to create a decentralized blacklist blockchain structure. For this purpose, machine learning infrastructure is prepared as a decision-making mechanism. A decentralized blacklist and request validator blockchain infrastructure that executes actions with the outputs of the decision-making mechanism has been prepared. These two infrastructures work together to create a real-time, reliable, and objective security structure. These infrastructures working together realize a secure network operation by detecting whether the request in the network is an anomaly or not and taking precautions.

3.1 Dataset Description

KDD'99 dataset by Salvatore J. Stolfo et al. [11] has been one of the most widely used datasets for evaluating anomaly detection since 1999. The KDD training dataset consists of about 4,000,000 single-link vectors [12]. Each vector has 42 attributes. His 42nd attribute in the record is the class attribute, which indicates whether the link is an attack or a normal link. Class attributes are divided into five classes, one normal class and four attacks (probe, DoS, R2L, and U2R). The categories in which attacks occur are listed below [13].

- DoS attack: An attacker can cause a computer or memory resource to become sufficiently busy or full that it cannot process legitimate requests or deny access to the computer by legitimate users[14].
- User to Root Attack (U2R): Attackers attempt to gain root access to a system by accessing regular user accounts on the system and exploiting vulnerabilities (through password sniffing, dictionary attacks, or social engineering) [15].
- Remote-to-local attacks (R2L): An attacker could send packets over the network to a computer that does not have an account. However, by exploiting some vulnerability he gains his access locally as a user on this computer. An R2L attack is

unauthorized access from a remote computer. As R2L attack types he can specify Imap, Ftp Write, Phf, and Warezmaster [16].

- Probing Attack: It is a type of attack against computer networks or systems that aims to gather information about the network or systems. A probing attack assumes that the attacker can access individual components of a device, such as CPU/GPU/ASIC, RAM, non-volatile storage, or data paths, but cannot perform the invasive attacks necessary to access the internals of the device.

The first 41 attributes in the dataset can be categorized into four groups according to their characteristics: Basic (T), Content (C), Traffic (TT), and Host (H) attributes. The attributes of individual TCP connections refer to Basic attributes, attributes within a connection refer to Content attributes, attributes calculated using a two-second time window refer to Traffic attributes, and attributes designed to evaluate attacks lasting longer than two seconds refer to Host attributes.

3.2 Machine Learning Module

The Machine Learning Module applies a machine learning approach to the NSL-KDD dataset to determine whether requests that occur in the network are anomalies. We evaluated machine learning algorithms that model with the highest accuracy in anomaly detection [17]. There are many machine learning algorithms as supervised and unsupervised learning. We examined the advantages and disadvantages of these algorithms because of the literature survey. We defined some rules for selecting the machine learning algorithms used in this study. These rules are listed below:

1. Providing algorithm diversity
2. Use of algorithms in current studies
3. Algorithms have the potential for anomaly detection

3.3 Classifier Selection

Machine learning algorithms have been described in detail in many survey studies. Therefore, we have chosen to briefly describe the machine learning algorithms used in this study. Figure 4 shows the selection of the best-performing algorithm.

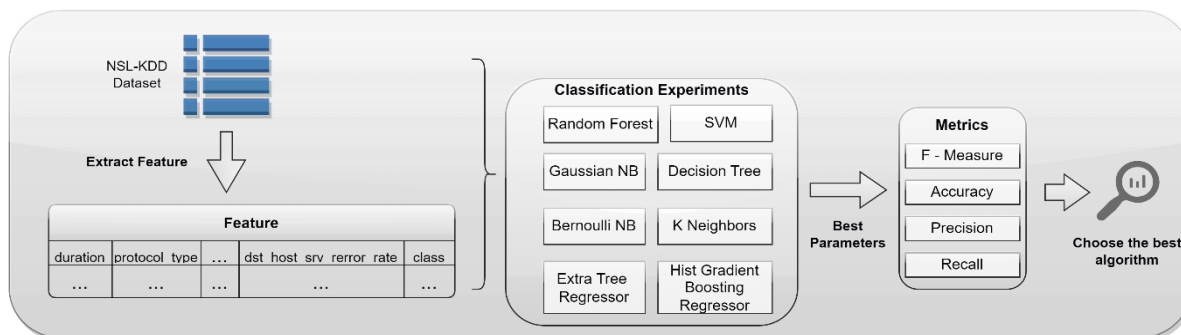


Figure 4 Machine learning algorithm selection

3.4 Random Forest Algorithm

The Random Forest Algorithm (RF) [18], first introduced by Leo Breiman, is a popular tool for ensemble learning. Trees in a forest learn to use a subset of feature variables. While RF works efficiently with large data sets, the generated forests or trees can be stored for later use [19]. It can handle data sets with outliers and noisy data while providing insight into the influence of variables in classification [20]. Tree-based ensemble learning algorithms are used in many industries and services such as healthcare [21], agriculture [22], transportation [23], and energy [24].

3.5 Support Vector Machine Algorithm

Support Vector Machines, developed by Vapnik et al. in the 1900s as one of the supervised learning methods, are used to classify linear or nonlinear data [25]. SVM is a popular machine learning algorithm that creates hyperplanes for the separation of data consisting of multiple classes in the data set [26]. With the developing technology, the amount of data obtained today is increasing. While this may seem beneficial, more data means more possibilities to identify meaningful data. This can create memory and time complexity for SVM training [27]. SVM has significant advantages in classification as it reduces the error

during training by using structural risk minimization [28]. As a result of its success in classification, SVM has been applied in many different fields such as human action recognition [29], text classification [30], and financial application [31].

3.6 Decision Tree Algorithm

The decision tree classifier is one of the most popular machine learning techniques. Decision trees built based on knowledge acquisition are used to classify test data [32]. A decision tree is a structure containing decision nodes and leaf nodes. Decision nodes are associated with a test X on a particular attribute of the input data and have branches that process the results of the X tests. Each leaf node represents a class with a decision outcome of the situation [33].

3.7 KNN Algorithm

The nearest neighbor algorithm (KNN) is a nonparametric supervised classification algorithm that produces efficient results with simple but effective performance [34]. The KNN classifier finds and analyses the nearest neighbors of sample x and classifies x into the class that has the most representatives among the neighbors. KNN calculates all distances for each state in the training set. This may not be practical for large datasets, as the growth of the dataset may incur time costs in calculating distances [35]. It has been successfully applied in many fields such as text classification [36], health [37], and economics [38].

3.8 Gaussian NB & Bernoulli Algorithm

Bayesian method is a statistical method used to calculate the probability of an event occurring based on its observed effects. Naive Bayes is a simple probabilistic classification technique based on the Bayes theorem with strong independence assumptions [39]. Gaussian NB assumes that when the attributes are continuous, the values associated with the classes are sampled according to a Gaussian distribution, i.e., a normal distribution. Bernoulli NB assumes that each of the multiple features is a binary-valued (present-absent, normal-attack) variable [30].

3.9 Extra Trees Regressor

A refinement of the Random Forest algorithm, the Extremely Random Tree (or Extra Tree) algorithm is a relatively new machine learning method that is less prone to overfit a dataset [40]. Similar to random forests, extra trees (ET) train each base predictor using a random selection of features. But to divide the nodes, it chooses at random the best characteristics and related values. Each regression tree is trained by ET using the whole training dataset. To train the model, RF employs bootstrap copies [41].

3.10 Gradient Boosting Regressor

Another sort of ensemble model is a gradient boosting regressor (GBR), which is an iterative collection of sequentially ordered tree models that allows the following model to learn from the errors of the preceding model. By "boosting" an ensemble of weak predictive models (often decision trees) to produce a more reliable model, this machine learning model delivers predictions [42].

3.11 Discussion and Analysis

Data were classified using machine learning for anomaly detection. The NSL-KDD dataset was trained and classified with machine learning algorithms. There are four attack types in the dataset: Probe, DoS, R2L and U2R. There are 67342 DoS, 11656 Probe, 995 R2L, and 52 U2R attacks in the dataset. Information on the number of attacks and normal cases in the dataset is given in Figure 5. The dataset was randomly mixed as 80% training data and 20% test data to obtain test and training datasets.

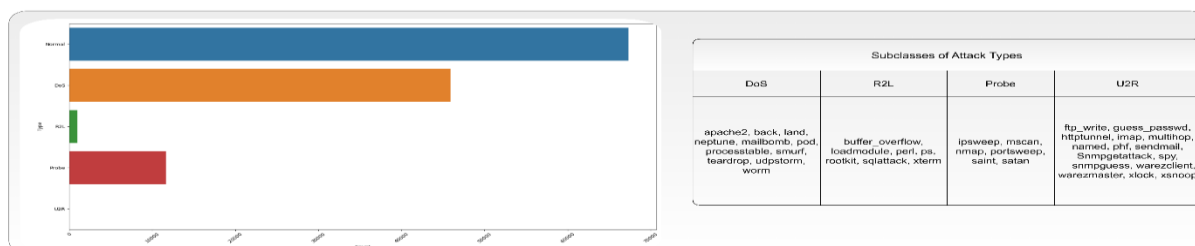


Figure 5 Number and subclasses of attack types in the dataset

Random Forest, SVM, Decision Tree, K Neighbours, Extra Trees Regressor, Gradient Boosting Regressor, Gaussian Naive Bayes, and Bernoulli Naive Bayes algorithms were used to classify the data. The parameters used in classification and classification results are given in Table 1.

Table 1. NSL-KDD Data set properties

Algorithms	Classification Parameters	Accuracy Percentage	Classification Result				
Random Forest	n_estimators:10, max_features:sqrt, criterion: entropy	0.99857	precision	recall	f1-score	support	
			DoS	1.00	1.00	1.00	9302
			Normal	1.00	1.00	1.00	13396
			Probe	1.00	0.99	1.00	2285
			R2L	0.99	0.96	0.97	203
			U2R	0.71	0.56	0.63	9
SVM	kernel='rbf', gamma=0.001, C=1000	0.99571	precision	recall	f1-score	support	
			DoS	1.00	1.00	1.00	9147
			Normal	0.99	1.00	1.00	13463
			Probe	0.99	0.98	0.99	2358
			R2L	0.98	0.93	0.95	218
			U2R	0.40	0.22	0.29	9
Desicion Tree	criterion: entropy, splitter: best, max_depth: None	0.99781	precision	recall	f1-score	support	
			DoS	1.00	1.00	1.00	9302
			Normal	1.00	1.00	1.00	13396
			Probe	0.99	0.99	0.99	2285
			R2L	0.96	0.99	0.97	203
			U2R	0.78	0.78	0.78	9
K-Neighbors	weights: distance, algorithm: auto	0.99293	precision	recall	f1-score	support	
			DoS	0.99	1.00	0.99	9302
			Normal	1.00	1.00	1.00	13396
			Probe	0.97	0.97	0.97	2285
			R2L	0.93	0.97	0.95	203
			U2R	0.75	0.67	0.71	9
Bernualli NB	alpha: 0.5, fit_prior: True	0.81282	precision	recall	f1-score	support	
			DoS	0.96	0.76	0.85	9302
			Normal	0.91	0.91	0.91	13396
			Probe	0.28	0.51	0.36	2285
			R2L	0.29	0.42	0.35	203
			U2R	0.15	0.67	0.25	9
Gaussian NB	var_smoothing: 1.0	0.53169	precision	recall	f1-score	support	
			DoS	0.00	0.00	0.00	9302
			Normal	0.53	1.00	0.69	13396
			Probe	0.50	0.00	0.00	2285
			R2L	0.00	0.00	0.00	203
			U2R	0.00	0.00	0.00	9
Extra Trees Regressor		0.97872	precision	recall	f1-score	support	
			DoS	1.00	1.00	1.00	9231
			Normal	1.00	0.99	0.99	2344
			Probe	0.35	0.53	0.42	15
			R2L	0.27	0.98	0.43	195
			U2R	1.00	0.96	0.98	13410
HistGradient Boositng Regressor		0.57963	precision	recall	f1-score	support	
			DoS	0.89	0.92	0.90	9169
			Normal	0.99	0.56	0.71	2397
			Probe	0.06	0.57	0.11	7
			R2L	0.02	0.84	0.04	198
			U2R	1.00	0.34	0.51	13424

Random Forest, SVM, Decision Tree, and K-Neighbors classifiers achieved approximately 99 percent classification success, while Bernoulli Naive Bayes achieved 81 percent and Gaussian Naive Bayes achieved 53 percent classification success. The error matrices of the classifications are given in Figure 6.

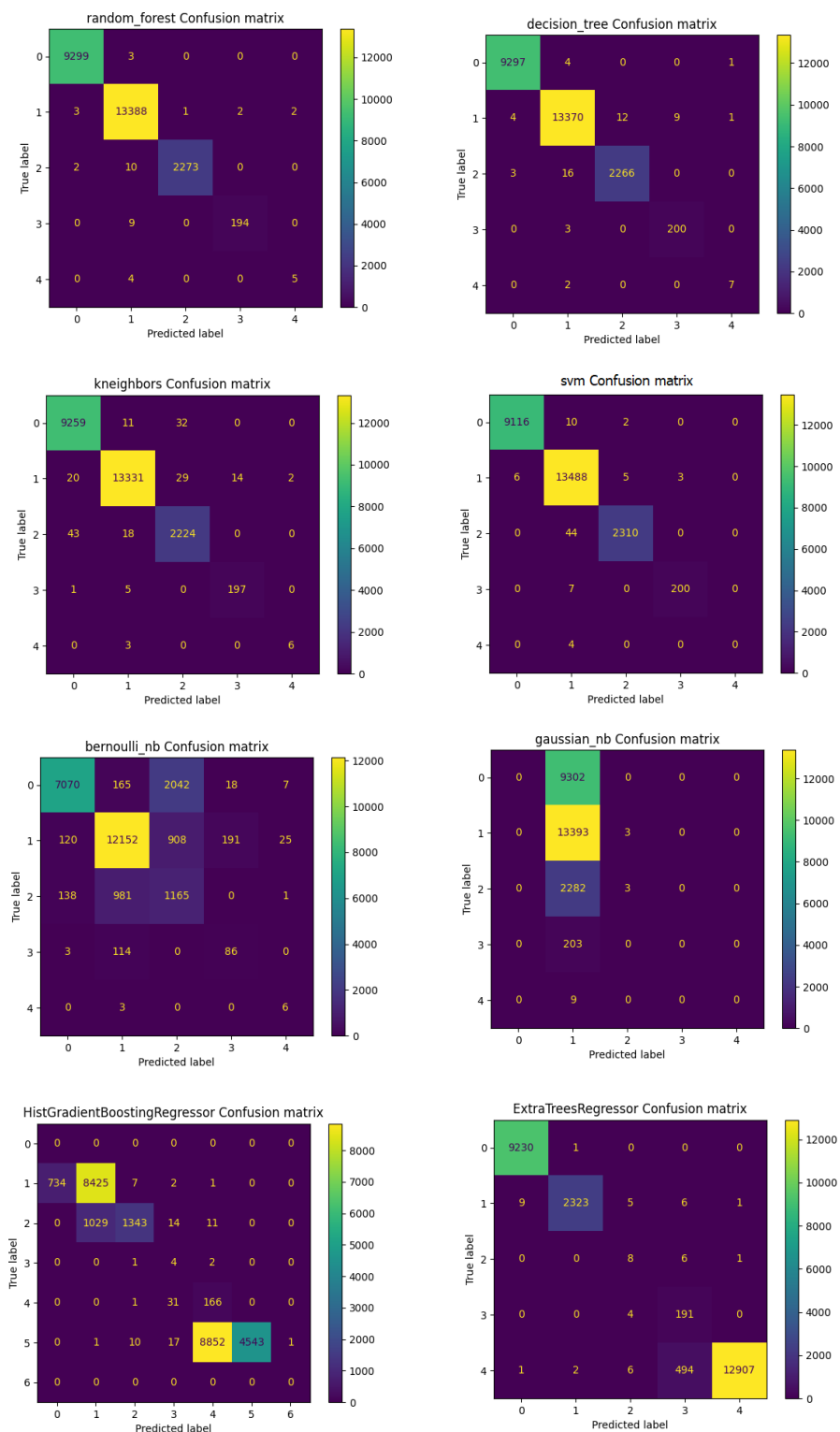


Figure 6. Confusion Matrix for Machine Learning algorithms

In detecting attacks, factors such as machine learning algorithms and data pre-processing change the success rates. Table 2 summarizes the results obtained by different researchers.

Table 2 Comparison of the study with similar studies

Ref.	Year	Research Paper Title	Algorithm used in preprocessing / Model core	Accuracy %
[43]	2015	Research on NSL-KDD data set of intrusion detection system based on classification algorithm	CFS J48 SVM Naive Bayes	Varies between 70.1 and 99.8 for different attack types and algorithms
[44]	2016	Anomaly-based intrusion detection system through feature selection analysis and construction of hybrid efficient models	SMOTE CANN	98.99
[45]	2016	A hybrid data mining approach for intrusion detection in the imbalanced NSL-KDD dataset.	Hybrid comprising of J48 Random Tree Naïve Bayes	99.81
[46]	2022	A Hybrid Machine-Learning Ensemble for Anomaly Detection in Real-Time Industry 4.0 Systems	Hybrid SVM Model	89.7
[47]	2023	Hybrid Statistical-Machine Learning for Real-Time Anomaly Detection in Industrial Cyber-Physical Systems	Hybrid LSTM Model	95
This Work	2023	Real-Time Intelligent Anomaly Detection and Prevention System	Random Forest	99.85

Algorithm 1 Blacklist Smart Contract Pseudo Code

```

1  func Initialize ()
2      configure DistrubutedLedgerRules ()
3      configure DistrubutedLedgerStandarts ()
4  func CreateBlackListAsset (ctx, params) ←Ip, Mac, Timestamp, Request Address
5      if exist (ctx in BlackList) == true then
6          return rejection.
7      else
8          add StandartLogList (id ← params)
9          return (obj ⊃ [params])
10
11 func GetBlackList (ctx)
12     if exist (ctx in BlackList) == true then
13         while! result. done
14             var res = GetAllList (ctx, id) then
15                 return result
16     else
17         add StandartLogList (id ← params)
18         return (obj ⊃ [params])
19
20 func GetAllLogList (ctx, id)
21     do
22         static allLogListResult = []
23         while! result. done then.
24             allLogListResult.Push → Key: result.val.key, Record: params
25             result ← await. iterator. next ()
26         return allResult end
27
28

```

In the prepared smart contract, the initial rules and settings of the distributed ledger structure are executed with the Initialize method. The classification information from machine learning is integrated into our smart contract as an asset. This asset is

represented as "ctx" in the Pseudo code. To complete the security process, the smart contract checks the machine-learning results of the request in the blockchain and executes forked transactions according to the process. When creating the blacklist ledger, some information about the requesting request is requested and its status in the list is checked. Depending on the returned result, the blacklist process is managed. When a positive response is received, a new object is created, and a new record is returned at the end of the process. The GetBlackList or GetAllList method is executed to read the records. After checking the necessary permissions, the data saved in the ledger can be read and listed with the help of an iterator. Algorithm 1 shows the pseudo-code of the generated smart contract.

4. Findings

Each network created in blockchain systems has a limit of transactions per second that it can process. This limit is referred to as TPS (Transfer Per Second). TPS is an acronym that stands for how many transactions per second blockchain networks can confirm and validate. In Figure 7, the error rates at different TPS values are measured with 10 different threads performing simultaneous tasks for a fixed duration of 20 seconds. In these measurements, the TPS value varies between 20-1000. In light of the findings obtained, it is seen that the blockchain successfully manages the requests received from 10 different threads with small-valued error rates up to 800 TPS values. As the TPS value increases above 800, it is seen that the error values start to increase linearly. In this sense, it can be said that the upper limit of the performance of the proposed blockchain system is the TPS value of 800.

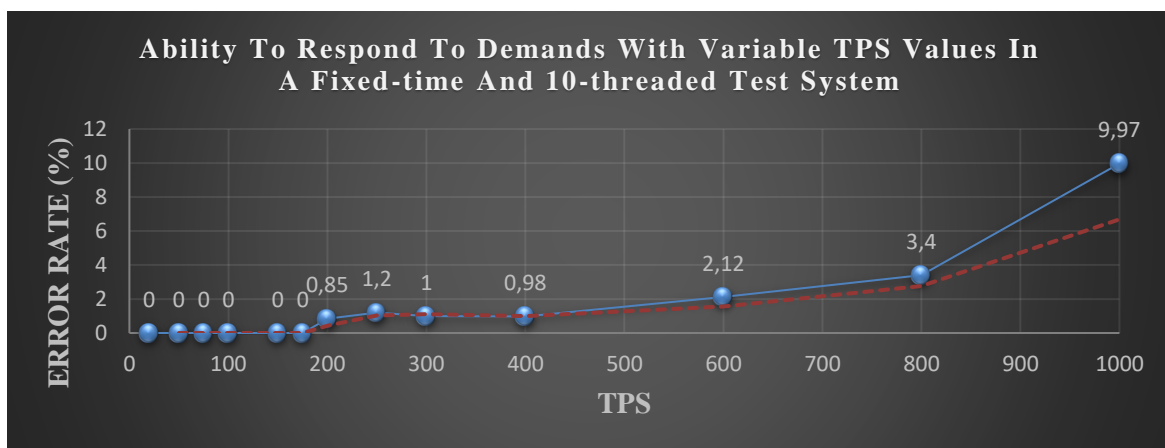


Figure 7 Ability to respond to demands with variable TPS values in a fixed-time and 10-threaded test system.

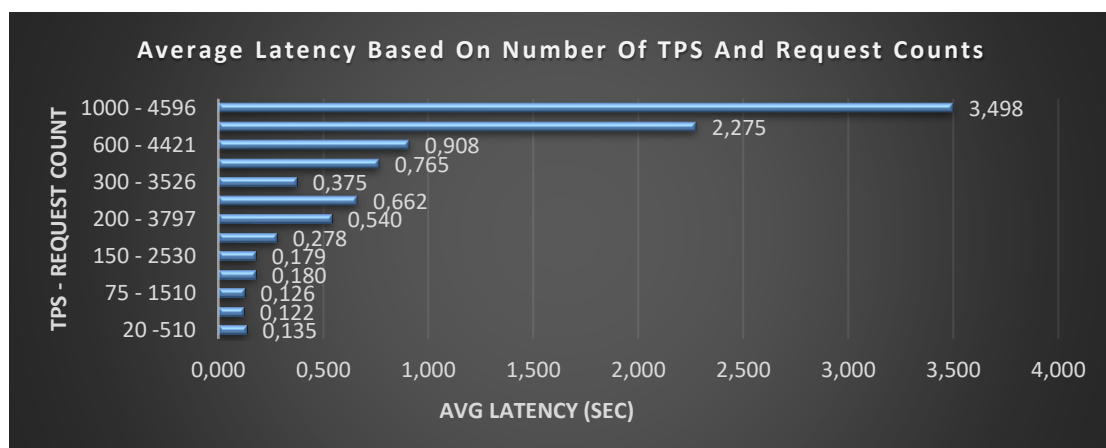


Figure 8 Average latency based on the number of TPS and Request Counts

It was envisaged that the temporal performance of the system should be evaluated by measuring the error rates during its active operation. For this reason, in Figure 8, the number of requests generated according to TPS values ranging from 20-1000 were combined and the average completion time of the blockchain process was measured. With the findings obtained, the average delay time experienced in the blockchain system until the TPS value reaches 800 varies between 0.135 seconds and 0.908 seconds. Although this latency is considered acceptable for a blockchain system with strong verification and

logging processes, it is observed that the average latency suddenly reaches 2.227 and gradually increases as the TPS value exceeds 800.

Considering the findings obtained from Figure 7 and Figure 8, it can be said that the upper limit of the performance of the prepared blockchain structure has a value of 800 TPS. The fact that the results obtained in these two graphs confirm each other shows the objectivity of the measurement processes performed.

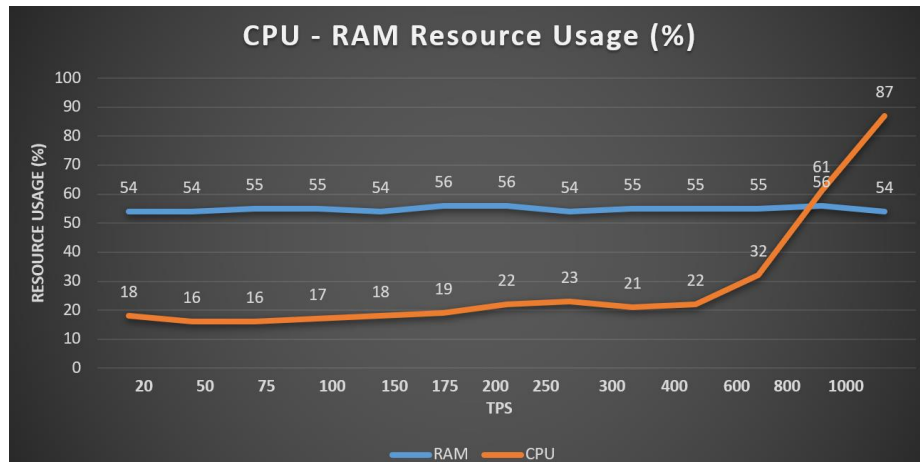


Figure 9 CPU - RAM Resource Usage (%)

To complete the performance evaluation of the proposed blockchain structure, it was deemed necessary to determine how it utilizes computer resources during its operation. The measurements were performed on a computer equipped with an Intel(R) Xeon(R) E-2236 CPU @ 3.40GHz 3.41 GHz and 32 GB of RAM. For this reason, in Figure 9, the CPU and RAM resource utilization rates of the computer during the execution of blockchain transactions are measured and graphed. In the measurement, was aimed to measure the upper limits of the system by increasing the amount of work demand per unit time (TPS value). In the findings obtained, it was interpreted that the RAM capacity did not show a significant change and therefore remained constant. It was found that the processor power remained at similar values until 600 TPS and increased linearly after 700 TPS. These findings show us that the optimal upper limit of the system in terms of hardware is 800 TPS, just like in Figure 6 and Figure 7.

5. Conclusion

In this study, blockchain technology and machine learning algorithms are combined to propose a real-time prevention model that can detect anomalies that may occur in network traffic. The classification criteria, success values, and classification results of the algorithms used in the training are explained and demonstrated in detail. According to the results obtained, the Decision Tree algorithm has the most successful classification results among the tested algorithms. In the prepared blockchain structure, anomalies detected with the help of smart contracts are transferred to the blacklist chain. Standard requests continue their processes in the usual flow of network traffic. The performance measurements of these transactions have been meticulously measured and resource utilization has been measured and shown in the study. As the TPS value exceeded 500, an increase in error conditions, response delay times, and resource utilization was observed. When the security and decentralization contributions provided by the system are evaluated, it can be said that the results obtained are satisfactory. In future studies, we plan to improve the resource utilization and time performance of this system. We aim to minimize error rates by including optimization methods in our proposed model.

References

- [1] S. Walling and S. Lodh, "Performance Evaluation of Supervised Machine Learning Based Intrusion Detection with Univariate Feature Selection on NSL KDD Dataset," Feb. 2023, doi: 10.21203/RS.3.RS-2537820/V1.
- [2] T. S. Reddy and R. Sathya, "Ensemble Machine Learning Techniques for Attack Prediction in NIDS Environment," Iraqi Journal For Computer Science and Mathematics, vol. 3, no. 2, pp. 78–82, Mar. 2022, doi: 10.52866/IJCSM.2022.02.01.008.
- [3] S. Aktar and A. Yasin Nur, "Towards DDoS attack detection using deep learning approach," Comput Secur, vol. 129, p. 103251, Jun. 2023, doi: 10.1016/J.COSE.2023.103251.
- [4] A. N. Özalp and Z. Albayrak, "Detecting Cyber Attacks with High-Frequency Features using Machine Learning Algorithms," Acta Polytechnica Hungarica, vol. 19, no. 7, pp. 213–233, 2022, doi: 10.12700/APH.19.7.2022.7.12.
- [5] G. Fernandes, J. J. P. C. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, "A comprehensive survey

- on network anomaly detection,” *Telecommunication Systems* 2018 70:3, vol. 70, no. 3, pp. 447–489, Jul. 2018, doi: 10.1007/S11235-018-0475-8.
- [6] V. Dutta, M. Choraś, M. Pawlicki, and R. Kozik, “A Deep Learning Ensemble for Network Anomaly and Cyber-Attack Detection,” *Sensors* 2020, Vol. 20, Page 4583, vol. 20, no. 16, p. 4583, Aug. 2020, doi: 10.3390/S20164583.
- [7] A. Rawashdeh, M. Alkasassbeh, and M. Al-Hawawreh, “An anomaly-based approach for DDoS attack detection in cloud environment,” *International Journal of Computer Applications in Technology*, vol. 57, no. 4, pp. 312–324, 2018, doi: 10.1504/IJCAT.2018.093533.
- [8] N. Hoque, H. Kashyap, and D. K. Bhattacharyya, “Real-time DDoS attack detection using FPGA,” *Comput Commun*, vol. 110, pp. 48–58, Sep. 2017, doi: 10.1016/J.COMCOM.2017.05.015.
- [9] A. Gurina and V. Eliseev, “Anomaly-Based Method for Detecting Multiple Classes of Network Attacks,” *Information* 2019, Vol. 10, Page 84, vol. 10, no. 3, p. 84, Feb. 2019, doi: 10.3390/INFO10030084.
- [10] J. Alsamiri and K. Alsubhi, “Internet of Things Cyber Attacks Detection using Machine Learning,” *IJACSA International Journal of Advanced Computer Science and Applications*, vol. 10, no. 12, 2019, Accessed: May 10, 2023. [Online]. Available: www.ijacsa.thesai.org
- [11] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, “Cost-based modeling for fraud and intrusion detection: Results from the JAM project,” *Proceedings - DARPA Information Survivability Conference and Exposition, DISCEX 2000*, vol. 2, pp. 130–144, 2000, doi: 10.1109/DISCEX.2000.821515.
- [12] “UCI Machine Learning Repository: KDD Cup 1999 Data Data Set.” <https://archive.ics.uci.edu/ml/datasets/kdd+cup+1999+data> (accessed Mar. 29, 2023).
- [13] M. Tavallae, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” *IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009*, Dec. 2009, doi: 10.1109/CISDA.2009.5356528.
- [14] R. Vishwakarma and A. K. Jain, “A survey of DDoS attacking techniques and defence mechanisms in the IoT network,” *Telecommun Syst*, vol. 73, no. 1, pp. 3–25, Jan. 2020, doi: 10.1007/S11235-019-00599-Z/TABLES/5.
- [15] D. Sklavounos, “Statistical Process Control Method for Cyber Intrusion Detection (DDoS, U2R, R2L, Probe),” *International Journal of Cyber-Security and Digital Forensics*, vol. 8, no. 1, pp. 82–88, 2019, doi: 10.17781/P002560.
- [16] M. Amini, R. Jalili, and H. R. Shahriari, “RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks,” *Comput Secur*, vol. 25, no. 6, pp. 459–468, Sep. 2006, doi: 10.1016/J.COSE.2006.05.003.
- [17] M. Ahsan, K. E. Nygard, R. Gomes, M. M. Chowdhury, N. Rifat, and J. F. Connolly, “Cybersecurity Threats and Their Mitigation Approaches Using Machine Learning—A Review,” *Journal of Cybersecurity and Privacy* 2022, Vol. 2, Pages 527-555, vol. 2, no. 3, pp. 527–555, Jul. 2022, doi: 10.3390/JCP2030027.
- [18] L. Breiman, “Random forests,” *Mach Learn*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324/METRICS.
- [19] K. Shah, H. Patel, D. Sanghvi, and M. Shah, “A Comparative Analysis of Logistic Regression, Random Forest and KNN Models for the Text Classification,” *Augmented Human Research* 2020 5:1, vol. 5, no. 1, pp. 1–16, Mar. 2020, doi: 10.1007/S41133-020-00032-0.
- [20] V. F. Rodriguez-Galiano, B. Ghimire, J. Rogan, M. Chica-Olmo, and J. P. Rigol-Sanchez, “An assessment of the effectiveness of a random forest classifier for land-cover classification,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 67, no. 1, pp. 93–104, Jan. 2012, doi: 10.1016/J.ISPRSJPRS.2011.11.002.
- [21] C. Iwendi et al., “COVID-19 patient health prediction using boosted random forest algorithm,” *Front Public Health*, vol. 8, p. 357, Jul. 2020, doi: 10.3389/FPUBH.2020.00357/BIBTEX.
- [22] J. Magidi, L. Nhamo, S. Mpandeli, and T. Mabhaudhi, “Application of the Random Forest Classifier to Map Irrigated Areas Using Google Earth Engine,” *Remote Sensing* 2021, Vol. 13, Page 876, vol. 13, no. 5, p. 876, Feb. 2021, doi: 10.3390/RS13050876.
- [23] X. Cheng and B. Huang, “A center-based secure and stable clustering algorithm for VANETs on highways,” *Wirel Commun Mob Comput*, vol. 2019, 2019, doi: 10.1155/2019/8415234.
- [24] D. Liu and K. Sun, “Random forest solar power forecast based on classification optimization,” *Energy*, vol. 187, p. 115940, Nov. 2019, doi: 10.1016/J.ENERGY.2019.115940.
- [25] M. A. Chandra and S. S. Bedi, “Survey on SVM and their application in image classification,” *International Journal of Information Technology (Singapore)*, vol. 13, no. 5, pp. 1–11, Oct. 2021, doi: 10.1007/S41870-017-0080-1/TABLES/1.
- [26] S. Dong, “Multi class SVM algorithm with active learning for network traffic classification,” *Expert Syst Appl*, vol. 176, p. 114885, Aug. 2021, doi: 10.1016/J.ESWA.2021.114885.
- [27] J. Nalepa and M. Kawulok, “Selecting training sets for support vector machines: a review,” *Artificial Intelligence Review* 2018 52:2, vol. 52, no. 2, pp. 857–900, Jan. 2018, doi: 10.1007/S10462-017-9611-1.
- [28] M. Tanveer, T. Rajani, R. Rastogi, Y. H. Shao, and M. A. Ganaie, “Comprehensive review on twin support vector machines,” *Ann Oper Res*, pp. 1–46, Mar. 2022, doi: 10.1007/S10479-022-04575-W/TABLES/8.
- [29] S. Agarwal, D. Tomar, and Siddhant, “Prediction of software defects using twin support vector machine,” *Proceedings of the 2014 International Conference on Information Systems and Computer Networks, ISCON 2014*, pp. 128–132, Nov. 2014, doi: 10.1109/ICISCON.2014.6965232.

- [30] N. Rezaeian and G. Novikova, "Persian Text Classification using naive Bayes algorithms and Support Vector Machine algorithm," *Indonesian Journal of Electrical Engineering and Informatics (IJEI)*, vol. 8, no. 1, pp. 178–188, Mar. 2020, doi: 10.52549/IJEI.V8I1.1696.
- [31] F. E. H. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," *Omega (Westport)*, vol. 29, no. 4, pp. 309–317, Aug. 2001, doi: 10.1016/S0305-0483(01)00026-3.
- [32] I. D. Mienye, Y. Sun, and Z. Wang, "Prediction performance of improved decision tree-based algorithms: a review," *Procedia Manuf*, vol. 35, pp. 698–703, Jan. 2019, doi: 10.1016/J.PROMFG.2019.06.011.
- [33] G. Stein, B. Chen, A. S. Wu, and K. A. Hua, "Decision tree classifier for network intrusion detection with GA-based feature selection," *Proceedings of the Annual Southeast Conference*, vol. 2, pp. 2136–2141, 2005, doi: 10.1145/1167253.1167288.
- [34] S. Hota, S. P.-Int. J. Eng. Technol, and undefined 2018, "KNN classifier based approach for multi-class sentiment analysis of twitter data," *scholar.archive.org*, vol. 7, no. 3, pp. 1372–1375, 2018, doi: 10.14419/ijet.v7i3.12656.
- [35] F. Moreno-Seco, L. Micó, and J. Oncina, "A modification of the LAESA algorithm for approximated k-NN classification," *Pattern Recognit Lett*, vol. 24, no. 1–3, pp. 47–53, Jan. 2003, doi: 10.1016/S0167-8655(02)00187-3.
- [36] S. Tan, "An effective refinement strategy for KNN text classifier," *Expert Syst Appl*, vol. 30, no. 2, pp. 290–298, Feb. 2006, doi: 10.1016/J.ESWA.2005.07.019.
- [37] A. Murugan, S. A. H. Nair, and K. P. S. Kumar, "Detection of Skin Cancer Using SVM, Random Forest and kNN Classifiers," *J Med Syst*, vol. 43, no. 8, pp. 1–9, Aug. 2019, doi: 10.1007/S10916-019-1400-8/FIGURES/6.
- [38] Imandoust SB and Bolandraftar M. *Int. Journal of Engineering Research and Applications*. Vol. 3, Issue 5, Sep-Oct 2013, pp.605-610
- [39] J. Bains, K. Kaki, K. S.-I. J. of Computer, and undefined 2013, "Intrusion detection system with multi layer using Bayesian networks," *Citeseer*, vol. 67, no. 5, pp. 975–8887, 2013, Accessed: Mar. 29, 2023
- [40] Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, 63, 3-42.
- [41] John, V., Liu, Z., Guo, C., Mita, S., & Kidono, K. (2016). Real-time lane estimation using deep features and extra trees regression. In *Image and Video Technology: 7th Pacific-Rim Symposium, PSIVT 2015, Auckland, New Zealand, November 25-27, 2015, Revised Selected Papers 7* (pp. 721-733). Springer International Publishing.
- [42] Otchere, D. A., Ganat, T. O. A., Ojero, J. O., Tackie-Otoo, B. N., & Taki, M. Y. (2022). Application of gradient boosting regression model for the evaluation of feature selection techniques in improving reservoir characterisation predictions. *Journal of Petroleum Science and Engineering*, 208, 109244.
- [43] D. H. Deshmukh, T. Ghorpade, and P. Padiya, "Improving classification using preprocessing and machine learning algorithms on NSL-KDD dataset," in *Proceedings - 2015 IEEE International Conference on Communication, Information and Computing Technology, ICCICT 2015*, 2015.
- [44] K. Rai, M. S. Devi, and A. Guleria, "Decision Tree Based Algorithm for Intrusion Detection," vol. 2834, pp. 2828–2834, 2016.
- [45] S. Aljawarneh, M. Aldwairi, and M. B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," *J. Comput. Sci.*, vol. 25, pp. 152–160, 2016.
- [46] D. Velásquez et al., "A Hybrid Machine-Learning Ensemble for Anomaly Detection in Real-Time Industry 4.0 Systems," in *IEEE Access*, vol. 10, pp. 72024–72036, 2022, doi: 10.1109/ACCESS.2022.3188102.
- [47] W. Hao, T. Yang and Q. Yang, "Hybrid Statistical-Machine Learning for Real-Time Anomaly Detection in Industrial Cyber-Physical Systems," in *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 1, pp. 32–46, Jan. 2023, doi: 10.1109/TASE.2021.3073396.

Author(s) Contributions

All three authors contributed equally to the study.

Conflict of Interest Notice

The authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical Approval and Informed Consent

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.

Availability of data and material

Not applicable

Plagiarism Statement

This article has been scanned by iThenticate™.