Journal of Surveillance, Security and Safety

Original Article

Open Access

TENNER: intrusion detection models for industrial networks based on ensemble learning

Nicole do Vale Dalarmelina[1], Pallavi Arora[2], Geraldo Pereira Rocha Filho[3], Rodolfo I. Meneguette[1], Marcio Andrey Teixeira[4]

[1]Department of Computer System, University of Sao Paulo, ICMC, São Carlos 13566-590, Brazil.
[2]Department of Computer Science and Engineering, GPC GTB GARH, Moga 142001, India.
[3]Department of Computer Science, Universidade Estadual Do Sudoeste Da Bahia, UESB, Vitória da Conquista 3293-3391, Brail.
[4]Department of Computer Science, Federal Institute of Education, Science and Technology of São Paulo, IFSP, Campus Catanduva, 15808-305, Brazil.

Correspondence to: Nicole do Vale Dalarmelina, Department of Computer Science, University of São Paulo, Avenida Trabalhador São-carlense, 400, São Carlos, São Paulo 13566-590, Brazil. E-mail: nicole.dalarmelina@usp.br; ORCID: 0000-0003-4074-3143

How to cite this article: Dalarmelina NDV, Arora P, Filho GPR, Meneguette RI, Teixeira MA. TENNER: intrusion detection models for industrial networks based on ensemble learning. *J Surveill Secur Saf* 2024;5:80-94. http://dx.doi.org/10.20517/jsss.2023.51

Received: 14 Dec 2023 **First Decision:** 20 Feb 2024 **Revised:** 13 Mar 2024 **Accepted:** 9 Apr 2024 **Published:** 20 Apr 2024

Academic Editor: Leandros Maglaras **Copy Editor:** Yanbing Bai **Production Editor:** Yanbing Bai

Abstract

In the pursuit of discerning patterns within computer network attacks, the utilization of Machine Learning and Deep Learning algorithms has been prevalent for crafting detection models based on extensive network traffic datasets. Furthermore, enhancing detection efficacy is feasible by applying cluster learning techniques, wherein multiple Machine Learning models collaborate to yield detection outcomes. Nevertheless, it is imperative to discern the optimal features within the dataset for training the intrusion detection model. In the present study, we proffer a novel framework for feature selection and intrusion detection within industrial networks, employing Ensemble Learning to achieve commendable performance in terms of both high predictive accuracy and efficient learning duration. The outcomes evince that the proposed model exhibits an accuracy of 99.93%, with a mere one h and 34 min required for comprehensive training. In contrast, a model trained without the framework presented in this paper attains an accuracy of 99.94%, necessitating an extensive training period of 156 h. Notably, the detection model derived from the proposed solution demonstrates superior results in prediction time, accomplishing predictions within 0.0009 seconds, compared to the alternative model which requires 0.0076 seconds for predictions.

Keywords: Feature extraction, internet, security, ensemble learning

INTRODUCTION

Some incidents such as the interruption of Google services in 2020 and the crash of social media in 2021 brought questions about the importance of the Internet, especially in pandemic situations, where companies and employees depend on this resource for remote work. This technology has been inside people's lives also with the rise of the Internet of Things (IoT), which shows us that devices such as TVs, fridges, and even cars can connect to the Internet[1]. The Industrial IoT (IIoT) is also growing, allowing manufacturing and energy management machines to connect to an industrial network and Industrial Control Systems (ICSs) that manage the behavior of those machines[2,3].

With the progress of the Internet, cybersecurity has become a concern because once a device or system is connected to it, it turns into a potential target for cyberattacks[4]. Some attacks that we can cite in this paper are the reconnaissance attack and the Denial of Service (DoS) attack[5]. The reconnaissance attack is usually used to learn about the target, so the cybercriminal studies its behavior such as vulnerabilities the target may have[6,7]. On the other hand, the DoS makes services unavailable by overloading the target network, as presented by Murini[8]. It has a variant called Distributed DoS (DDoS) that takes over multiple machines to overload even more the network and crash services, as shown by Zargar *et al.*[9].

Thinking about these attacks and the IIoT scenario, Intrusion Detection Systems (IDSs) have been developed to detect potential attacks on networks. Discussing the differences between IoT and IIoT networks, it is possible to say that the main difference between an industrial and a home network is the traffic patterns, since the industrial traffic is more deterministic, meaning that a small variation in the usual network flow can define if it is a malicious or normal traffic. This differentiation can be done by observing the network features such as the destination port, total packets sent or received, *etc*. This can make the detection easier, especially if Machine Learning (ML) and Deep Learning were used in the model training. Besides, using Ensemble Learning (EL) can also increase the detection because it can combine different ML algorithms to run the same task.

To train a machine learning (ML) model, it is necessary to utilize a dataset that reflects the current scenario. This dataset can consist of either real or simulated data. In real-world IIoT network data, the occurrence of attacks and intrusions is typically low compared to the volume of normal traffic, as observed in the study by Dalarmelina *et al.*[10].

Safhin *et al.* highlight the increasing frequency and sophistication of attacks targeting industrial systems, establishing attacks on the hardware limitations of devices that involve IIoT but also in relation to the entire flow of data and information exchanged by IIoT elements[11]. The cyber criminals aim to avoid detection; this often results in imbalanced data, where over 50% of the dataset comprises a single sample. This imbalance can pose challenges, particularly when training an IDS, as it may lead to a significant number of false alarms.

To address the imbalance issue, popular solutions such as Over Sampling and Under Sampling are employed. These approaches take different paths to achieve the same objective. Under Sampling involves reducing the majority class, while Over Sampling involves increasing the minority class, thereby equalizing the number of instances for each class within the dataset. The advantage of Over Sampling is that no data is discarded, but it comes at a high computational cost. In contrast, Under Sampling may result in data loss but offers a lower computational burden.

Thinking about the performance of models trained using ML, the use of datasets with ”unnecessary” features can show critical challenges in the data modeling[12]; besides, it is needed to care about the data balancing at the time we train the model, since an imbalanced dataset can generate biased models[13].

Considering the model performance, electing dataset features and addressing data imbalance during training are critical challenges. This paper aims to develop an industrial intrusion detection model named TENNER, based on EL. The model intends to exhibit high predictive accuracy and efficiency in learning time, utilizing datasets containing real attacks from an industrial water storage tank control network[13]. The objective is to contribute to the field by achieving satisfactory performance in industrial intrusion detection, considering the unique challenges posed by imbalanced datasets and the need for efficient model training. Therefore, the contributions of this paper are:

- Novel Intrusion Detection Model for Industrial Networks: The primary contribution of this work lies in the development of a novel intrusion detection model named TENNER, specifically tailored for industrial networks. By leveraging EL, the model aims to achieve heightened predictive accuracy and efficient learning time, addressing the unique challenges posed by the characteristics of industrial network traffic.
- Addressing Imbalance Challenges in IIoT Datasets: This research addresses the prevalent issue of imbalanced datasets in the context of IIoT and investigates and implements solutions such as Over Sampling and Under Sampling to mitigate the challenges associated with imbalanced data, ensuring the robustness of the intrusion detection model by equalizing class instances within the dataset.
- Practical Application of ML and EL in Cybersecurity: By utilizing ML and, specifically, EL techniques, the study contributes to practically applying advanced technologies in the domain of industrial cybersecurity. The integration of these methodologies aims to enhance the efficiency of IDSs, demonstrating their potential utility in addressing the evolving threat landscape of cyberattacks in industrial networks.

RELATED WORKS

In the literature, some works focus on ML for cybersecurity. Below, we will describe some of them.

Apruzzese *et al.* analyzed technologies for problems encountered in cybersecurity, such as malware analysis, intrusion and spam detection, in order to discover if these techniques would supply the prevention and identification needs of these problems[14]. According to other authors, the results showed that the technologies analyzed were still affected by some shortcomings that can reduce the security effectiveness in the cyberspace. The approaches showed vulnerability to attacks and needed constant retraining and non-automated parameters adjustments. The authors also highlight that when the same classifier is applied to identify different threats, the spam and intrusion detection decline.

Shafin *et al.* proposed a robust and efficient Android malware detection and classification system capable of detecting popular malicious attacks of recent times such as banking malware and riskware[15]. For this, the authors assessing the importance of features gives us an insight into contributions of individual features in detecting malware. They accomplished this by combining ensemble methods and stacking algorithm.

An intrusion detection model based on decision tree was developed by Sarker *et al.*, where the features of the network flow are initially ranked according to their importance, which, according to the authors, minimizes the computational complexity of the model[16]. The approach proved to be effective for the objective of the work. Teixeira *et al.* developed datasets of real attacks in an industrial environment. The authors developed a Supervisory Control And Data Acquisition (SCADA) testbed which offers resources that can be used for training, validating and comparing results in cyber security research[17]. They also highlight that the testbed provides a satisfactory understanding about the consequences and effects of attacks in real SCADA environments.

Using the K-Nearest Neighbors (KNN) classifier, suspect behaviors were identified in IoT network backbones by Pajouh *et al.*[18]. Component and linear discriminant analyses of the dimension module were performed to select features. According to the authors, the approach overcame previous models in the User to Root (U2R) and Remote to Local (R2L) attack detection. In the data balancing context for IDS training, Zolanvari *et al.* presented a study about the effects of an imbalanced dataset in training models using ML in the IIoT area[13]. An approach using EL is demonstrated by Arya and Gupta[19], where they leveraged Pearson Correlation Coefficient (PCC) and Isolation Forest to select the most appropriate features, testing the built IDS performance using the Random Forest classifier and Bot-IoT e NF-UNSW-NB15-v2 datasets. Additionally, Mohy-Eddine *et al.* propose EL utilization for feature selection employing four reducing irrelevant features techniques[20]. Consequently, two reduced feature groups were generated using union and intersection techniques. Verma Abhishek and Ranga Virender[21] investigated the prospects of using ML classification algorithms for securing IoT against DoS attacks. A comprehensive study is conducted on the classifiers that can advance the development of anomaly-based IDSs. Performance of classifiers is assessed in terms of prominent metrics and validation methods. Popular datasets CIDDS-001, UNSW-NB15, and NSL-KDD are used for benchmarking classifiers. Friedman and Nemenyi tests are employed to analyze the significant differences among classifiers statistically. Khoei *et al.* investigate the performance of three different ensemble learning techniques: bagging-, boosting-, and stacking-based[22]. Their results are compared to those of three traditional ML techniques, namely K nearest neighbor, decision tree, and Naive Bayes; for this, the authors used two features

Liang *et al.* introduced an industrial network intrusion detection algorithm rooted in a multi-feature data clustering optimization model[23]. The algorithm lays down the foundation of a data clustering optimization model tailored explicitly for identifying intrusion attacks within industrial networks. Subsequently, the authors proposed a cluster center selection algorithm complemented by an intrusion detection algorithm designed to handle multi-feature data. Notably, within the algorithm, despite the trace procedure augmenting the training center of the model, there is a significant enhancement in detection accuracy, particularly for attacks characterized by high levels of overlap and disguise.

Khan *et al. investigated an IDS model termed federated-simple recurrent units (SRUs) to safeguard IoT-based security of ICSs*[24]. More precisely, the federated-SRUs IDS model employs an enhanced SRUs architecture to diminish computational expenses and mitigate the gradient vanishing challenge in recurrent networks.

The works found in the literature, described in this section, mostly aim to develop an IDS model capable of meeting the needs of each specific scenario. Unlike these works, the goal of the present work is to analyze the better approach in terms of feature selection, data balancing and the use of EL to generate an efficient IDS model for industrial networks. Table 1 compares works found in the literature and the proposed solution. We can observe that TENNER addresses issues such as feature selection, which is not considered by Apruzzese*et al.*[14], Teixeira *et al.*[17], Verma Abhishek and Ranga Virender[21], Arya and Gupta[19], and Khan *et al.*[24] and the data balancing that, among the cited works, is treated by Zolanvari *et al.*[13] and Liang *et al.*[23].

TENNER aims to develop a model that IDS can use - represented in the IDS column -, as proposed by Sarker *et al.*[16], Teixeira *et al.*[17], and Pajouh *et al.*[18]. This model is also centered on industrial environment where IIoT is used, which can only be seen in the research of Zolanvari, Teixeira and Jain[17], Khan *et al.*[24], and Zolanvari *et al.*[13]; despite dealing with the industrial scenario, these studies do not have the development of the model as their main proposal. The present proposal also uses ML, as can be seen in the works of Apruzzese *et al.*[14], Teixeira *et al.*[17], Pajouh *et al.*[18], Khoei *et al.*[22], Liang *et al.*[23], and Shafin *et al.*[15], but specifically, using EL to train a more robust model can only be seen in the study of Mohy-Eddine *et al.*[20]. To reduce the model training time, there is still a reduction in the feature size while analyzing the optimization of model training time and possible model retraining without losing performance in predicting attacks on network flows.

Table 1. Comparison of the proposal with the works cited considering the specific objectives

Work	Feature selecion	IDS	IIoT	ML	Dataset balancing	EL	Database reduction
Apruzzese *et al.* [14]	-	-	-	✓	-	-	-
Khoei *et al.* [22]	✓	✓	-	-	-	-	-
Sarker *et al.* [16]	✓	-	-	-	- ✓	-	-
Shafin *et al.* [15]	✓	✓	-	-	-	-	-
Khan *et al.* [24]	-	✓	✓	✓	-	-	-
Liang *et al.* [23]	✓	✓	✓	✓	-	-	-
Teixeira *et al.* [17]	-	✓	✓	✓	-	-	-
Verma Abhishek and Ranga Virender [21]	-	✓		✓	-	-	-
Pajouh *et al.* [18]	✓	✓	-	✓	-	-	-
Zolanvari, Teixeira and Jain [13]	✓	-	✓	✓	✓	-	-
Arya e Gupta [19]	✓	-	✓	✓	✓	✓	-
Mohy-Eddine *et al.* [20]	✓	✓	✓	✓	-	✓	-
TENNER	✓	✓	✓	✓	✓	✓	✓

TENNER: INTRUSION DETECTION MODELS FOR INDUSTRIAL NETWORKS BASED ON ENSEMBLE LEARNING

TENNER is an industrial IDS leveraging EL to optimize the performance of IDSs used in industrial networks [Figure 1]. TENNER is composed of:

- (A) Database
  - To assemble the model proposed in this work, it was necessary to use databases containing flows from industrial networks, which were subjected to DDoS and Reconnaissance attacks, as found in the work of Teixeira *et al.* [17].
- (B) *FSAnalysis*
  - With the databases containing flows - normal and under attack - from industrial networks, it was necessary to implement the preprocess module *FSAnalysis* (selection of *features* and analysis, *Feature Selection Analysis*) [25] for selecting the best *features* for model prediction, in addition to balancing the data, so that there would be no bias in model training.
- (C) *Ensemble Learning*
  - Two different models were trained using ensemble learning, called *Ensemble Learning*, to compare the performance of trained models with a reduction in the size of each *feature* against those without this reduction. At the end of these tests, the best-performing model was selected, deciding whether the input stream was malicious and classifying it as ”normal” or ”attack”.

To build TENNER, initially, the *dataset* was submitted to the preprocessing module *FSAnalysis*; this determined the best approach for selecting *features* and deciding whether or not to balance the data to create the model. The features selected include *DstPort*, *SrcLoss*, *DstLoss*, and *PLoss*, and the model training carried out with the balanced database also obtained better results.

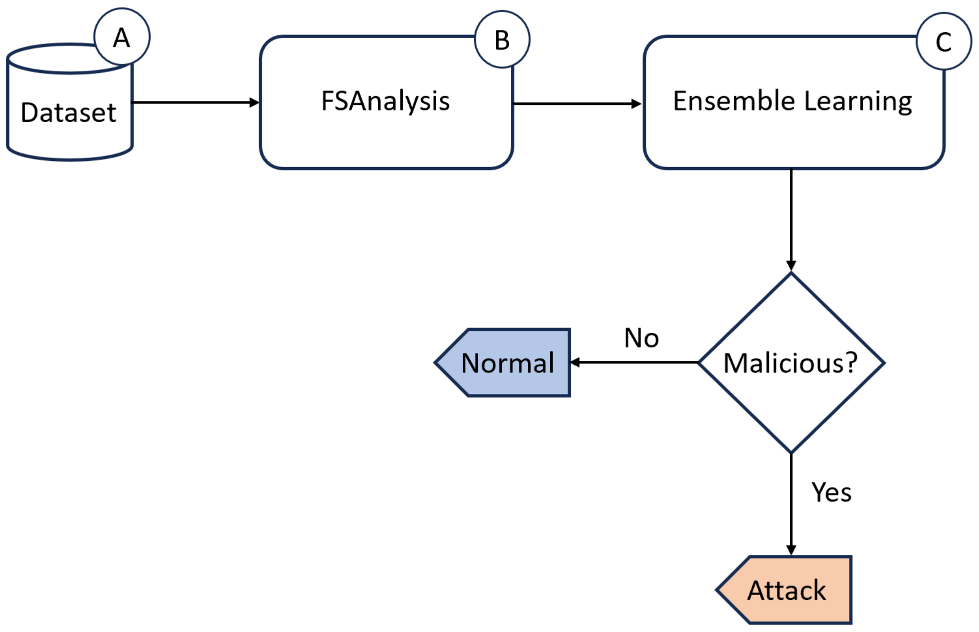


Figure 1. TENNER. TENNER: intrusion detection models for industrial networks based on ensemble learning framework.

It was observed that the database containing reconnaissance attacks used 3.4GB of memory, having five *features* of type *object*, eight of type *float64*, and 12 of type *int64*. With this information, a way to reduce the size of this database without losing any data was analyzed. Therefore, the *pandas* function, "*to_numeric*", was used to convert the types of these *features*, so that they could occupy less memory; so, *features* of type *float64* were converted to *float32*, and *features* of type *int64* were converted to *int32* and *int8*, according to the size of its value. This way, the memory usage of this database dropped to 2.8GB. To compare whether this reduction in *dataset* would make a significant difference in model training, the data processing step was segmented into a sub-step divided into two modes, the fast mode (FM) and the default mode (DM), as illustrated in Figure 2. In this setup, the databases, (D) the database that contains the DDoS attacks and (R) includes the Reconnaissance attacks, were submitted separately.

Fast Mode (FM) consists of (a) reducing the *features* using the method described previously to reduce the size they occupy. After lowering the *features*, the data is balanced (b) using the *Under Sampling* method so that the computational cost in generating the model is reduced. Then, *EL* is used, more specifically, the *Stacking Classifier* - using the *learners Decision Tree Classifier*, *Random Forest Classifier*, *KNeighbors Classifier*, and *XGB Classifier* -, to create the model (c) from the treated database. Finally, the completed model has its performance evaluated (d) using the *LogisticRegression* algorithm, from the "*linear_model*" module of the open source library "*sklearn*"[26], extracting the accuracy and the confusion matrix so that other calculations can be carried out. Once you have the trained model, it is possible to move on to the testing stage, where the model can identify malicious network flows, classifying them as attack or normal. However, Default Mode (DM) comprises conducting the same processes as FM, without process (a), so that the *features* continue to occupy the same amount of memory. At the end of the DM and FM experiments, the FM approach was selected as it obtained better results than DM using calculations such as accuracy, sensitivity, False Alarm Rate (FAR), and Undetected Rate (UR) extracted from the confusion matrix, along with learning and prediction time.

Dataset

To execute the current project proposal, two *datasets* found in the study of Teixeira *et al.* were used, where databases were created with real industrial network flows, which were subjected to DDoS and Reconnaissance

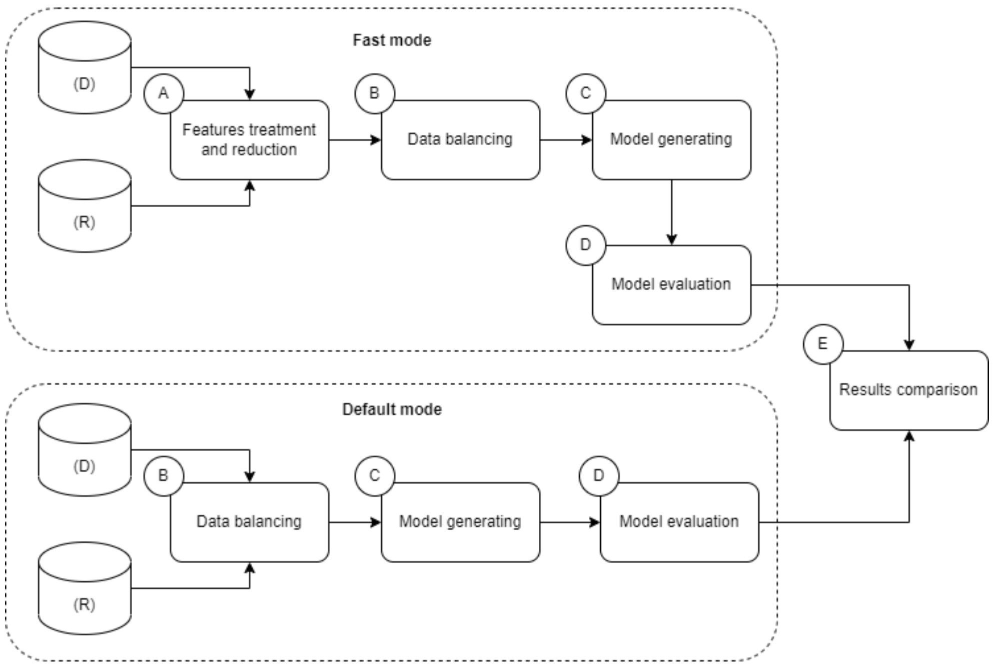


Figure 2. Database processing stage.

attacks: a database containing DDoS attacks - 51.73% of DoS attacks; 11.19% of DDoS attacks; 37.07% of DDoS attacks *Spoofing* - and another database containing 13 different Reconnaissance attacks[17].

Each database used contains 25 *features* that represent characteristics of the flow, such as destination *bits* per second, destination packets, percentage of dropped packets, and destination port number, among other essential characteristics for the network flow. Additionally, it includes the classification of this flow, with the value ”0” being the ”normal” flow classification and ”1” being the flow under attack.

FSAnalysis

This stage of the project involved conducting experiments to select the best approach in the context of choosing the best *features* and setting the *dataset* approach (balanced or unbalanced). To select the best *features*, the *Random Forest* algorithm and PCC were used. The *FSAnalysis* architecture flowchart, as illustrated in Figure 3, has the database (A) as its centralizing element. Two databases that contain samples of expected flows and attacks in an industrial network will be used. Each database has a different type of attack, namely reconnaissance and DDoS attacks.

Using the database, experiments will be performed using the *Random Forest* algorithm (B) and, in parallel, the PCC (C) to find the *features* of the most significant relevance for each method.

To use the *Random Forest* algorithm, the *RandomForestClassifier* method from the “*scikit-learn*” library was used. Even with the same training data, the best prediction combination found from this method may vary. In the present work, this could not happen as one of the objectives was to see the best *features* to be used in training an IDS for industrial networks, and experiments must be able to be replicated later. Therefore, to control the randomness of the sample initialization when building the decision trees, the value “0” was defined for the “*random_state*” parameter of the method, and the value “10” in the “ *n_estimators*” indicated that the model would generate only ten decision trees for training.

After the experiments using *Random Forest*, it was observed that the *features DstBytes*, *DstLoad*, and *DstRate*

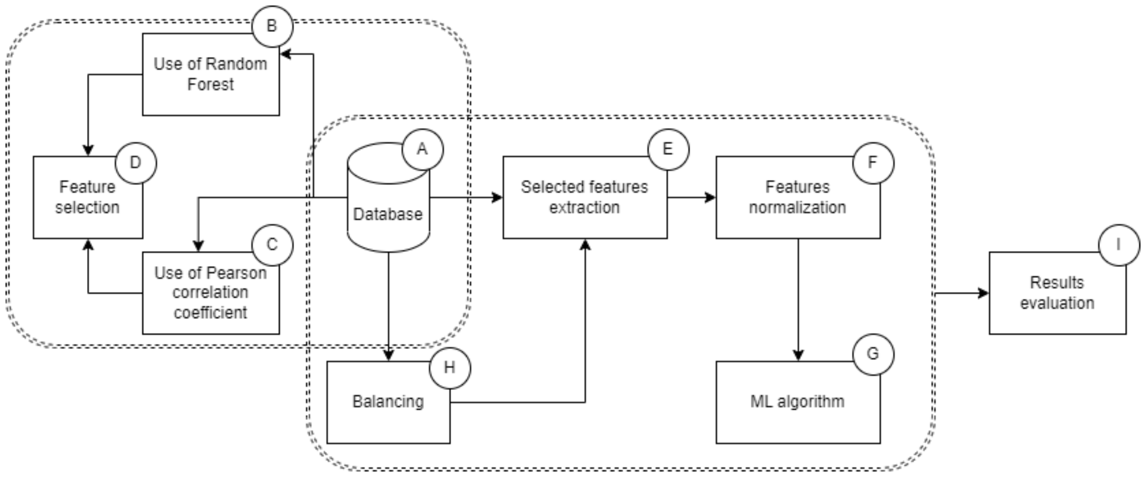


Figure 3. Flowchart of *FSAnalysis* adapted from Dalarmelina *et al.* [25].

are the ones that bring the most important characteristics when evaluating the database containing DDoS attacks (D), as these presented *scores* above 0.15. In the database containing Recognition (R) attacks, it was observed that *features pLoss*, *Dport*, *SrcPkts* and *TotPkts* were the most relevant, obtaining *scores* above 0.15. In the literature, it is possible to find several metrics for evaluating the best *scores* resulting from the PCC calculation. Cohen [27] considers that values between 0.50 and 1 can be interpreted as of great magnitude. At the same time, other authors believe that this range is a little smaller, from 0.70 or even 0.80 to 1. The value range for choosing the best *score* can vary according to each context and purpose. Therefore, in this work, for the selection criterion of the best *features*, it was determined that only *features* that obtained *score* greater than 0.55 would be selected. With this, it was noted that the best *features* using the database (D) were *DstLoss*, *pLoss*, *Mean* and *SrcLoss*, and using the database (R), *DstLoss*, *Dport*, *SrcLoss* and *pLoss*, achieving *scores* greater than 0.55. Details of the results can be found in the work by Dalarmelina *et al.* [25].

With the most relevant *features* for each attack, the data present in the databases used were normalized using the *StandardScaler* class from the "*preprocessing*" module of the "*sklearn*" library [26] and divided into two parts, 80% for training and 20% for testing. After this preprocessing, the *LogisticRegression* class, from the "*linear_model*" module of the open source library "*sklearn*" [26], created the network flow prediction model using the most important *features*, selected as previously described.

To conduct the prediction tests to choose the best approach, using the bases in complete and the balanced commands, three scenarios were applied: (1) application of the *features* selected using the algorithm *Random Forest*; (2) application of the selected *features* using PCC; (3) application of all *features* present in both databases. The same training routine was performed using *datasets* (D) and (R). Some calculations, including the *Logistic Regression* prediction algorithm, learning time, accuracy, sensitivity, FAR, non-detection rate (UR), and Matthews correlation coefficient (MCC) are employed, which are extracted from the confusion matrix that can be interpreted as follows:

- Input stream: Value to be analyzed by the algorithm. This value can be 0 or 1, with 0 being the numerical representation of the normal network flow and 1 being the flow under attack.
- True Negative: Represents the number of normal flows correctly classified as normal.
- False Positive: Indicates the number of normal flows wrongly classified as flow under attack.
- False Negative: Means the number of flows under attack mistakenly classified as normal flow.
- True Positive: Denotes the number of abnormal flows correctly classified as attacks.

The calculations used were FAR, Accuracy, MCC, Sensitivity or True Positive Rate, and the UR. The FAR

Table 2. Confusion matrix

		Prediction	
		Normal	**Under attack**
Input	**Actual normal**	True Negative (TN)	False Positive (FP)
	Actual under attack	False Negative (FN)	True Positive (TP)

calculates the normal traffic misclassified as attack percentage, which means the FP divided by FP plus TN, expressed as

$$FAR = \frac{FP}{FP + TN} \times 100 \tag{1}$$

The accuracy calculation results in the percentage of the model hits, determined by

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \tag{2}$$

The MCC measures the classification quality, denoted as

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \times 100 \tag{3}$$

To measure the effectiveness in predicting attacks correctly classified as attacks, the sensitivity is used, as given in

$$Sensitivity = \frac{TP}{TP + FN} \times 100 \tag{4}$$

Finally, the UR results in the predictions and real values correlation, established as

$$UR = \frac{FN}{FN + TP} \times 100 \tag{5}$$

The results have shown that, although the difference is not considerably large, the model trained with balanced databases performed better, reaching superior results, especially in terms of learning time, MCC and sensibility. Table 2 describes the confusion matrix.

Ensemble Learning

The EL can be composed of simple algorithms, the "weak learners", and complex algorithms, the "strong learners", which can result in better performances; a weighted average measures contributions of each learner from its best predictions using linear or logistic regression. In the present work, this learning method, named "Stacking", was used from "*sklearn.ensemble*" library, and the stacked algorithms were *Decision Tree*, *Random Forest*, *XGBoost* and the supervised learning algorithm KNN.

At the end of the *Stacking*, the *Logistic Regression* algorithm, one of the most commonly used methods for modeling binary data responses, was employed to train the generated model. With the model in hand, it was possible to evaluate its performance by comparing it with the real values, generating the confusion matrix.

Algorithm 1 demonstrates how the algorithm operates. It begins by populating the possibilities from two datasets, "ddos" containing DDoS attacks and "reconnaissance" containing reconnaissance attacks. At line 3, the learning modes "fast" (FM) and "default" (DM) are loaded. Subsequently, at lines 4 and 5, two for loops are executed to iterate through both datasets and both modes. Each time, the ExecutingCode method is called at line 6, passing the dataset name and the execution mode (FM or DM).

Within the ExecutingCode, the dataset CSV file is loaded according to the dataset passed through the parameter. In the condition presented on line 15, the dataset is loaded in the indicated mode. At line 16, the dataset is loaded without any changes. Additionally, at line 18, the ExecuteFastMode function is called, passing the dataset, where the feature treatment and dataset reduction are performed, returning the modified dataset.

At lines 20 and 21, the PCC features are retrieved. Once this is done, at line 22, the data is balanced using Under Sampling to reduce the dataset samples. Subsequently, at line 24, the Stacking method is used to employ EL using four different algorithms: DecisionTreeClassifier, RandomForestClassifier, KNeighborsClassifier, and XGBClassifier. The model is evaluated using the EvaluateModel function. This function receives the previously selected features using the feature importance calculation. Once the features are received, the function uses the trained model to verify if it is an attack. As these features are constant, it is possible to say that this function has an O(1) complexity.

TESTS AND RESULTS

The main focus of this work is to validate Tenner against EL. For that, it was made a reduction of the database where tests were made for measuring the model performance trained without the reduction (DM) and with the model trained with the reduction (FM) using the best approach found on *FSAnalysis* module; in other words, the models were trained using the selected features using PCC and balanced datasets. Further details about FSAnalysis module and the features selection can be found in the research of Dalarmelina *et al.*[10,25]. From the confusion matrix, some metrics were used in the present work to evaluate the trained model performance. Specifically, the metrics used are FAR, UR and sensibility.

The calculating accuracy is formulated in Equation 2 and the results are illustrated at 4(a), where it is possible to observe the results having a difference of 0.03% e 0.01%, respectively, for datasets (D) and (R). The FAR calculation used is determined by Equation 1 and the results are presented in 4(b), which also obtained a difference of 0.03% for the database (D) and 0.02% for the database (R). In the industrial network context, the UR can be considered more important than FAR, since this metric represents the under attack traffic portion that was wrongly classified as normal, and can be calculated using Equation 5. Analyzing the chart illustrated in 4(c), it is possible to observe the results having a bigger difference; despite still a little difference, FM obtained 0.1% and 0.09% more than the DM in the non-detection rate.

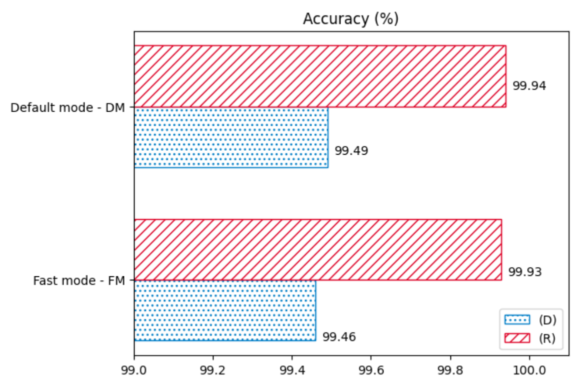
The sensibility models were evaluated by calculating the True Positive Rate to measure abnormalities correctly classified as attacks. For this task, Equation 4 was employed. The results with this formula are shown in 4(d), where it is possible to note that FM performed 0.03% better than DM with dataset (D). The MCC was also considered using Equation 3; the result is shown in 5(a); using this metric, the DM performed 0.09% and 0.15% better.

As can be seen in 4(d), both modes obtained similar results in all calculations performed, presenting a difference below 0.1% in three of the four calculated equations; both the learning time and the prediction time were also compared to evaluate the performance of the modes (DM and FM). 5(b) illustrates the time, in hours, covered by the models in the two modes presented. In this scenario, FM had an advantage, as it completed model training in 1.34 h, equivalent to approximately 80 min, and in 0.92 h, equivalent to approximately 55

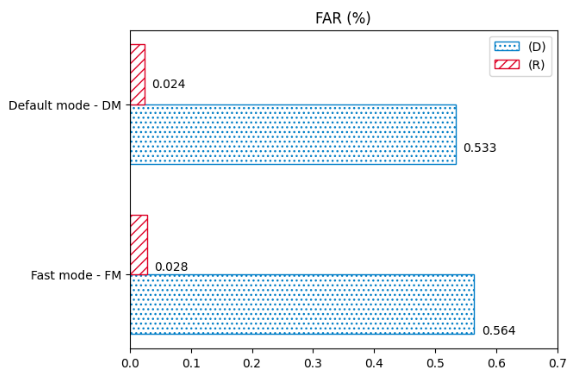
Algorithm 1 TENNER main function

```
$xres, xtrain, xtest, yres, ytrain, ytest \leftarrow []$
$dbNames \leftarrow ['ddos,' reconnaissance']$
$learningModes \leftarrow ['fast,' default']$

for i is 0 to 1 do
    for j is 0 to 1 do
        ExecutingCode(dbNames[i], learningModes[j])
    end for
end for

function ExecutingCode(dbName, learningMode)
    if dbName is 'ddos' then
        $dataset \leftarrow datasetDdos.csv$
    else
        $dataset \leftarrow datasetReconnaissance.csv$
    end if

    if learningMode is 'default' then
        $executionDB \leftarrow dataset$
    else
        $executionDB \leftarrow$ ExecuteFastMode(dataset)
    end if

    $y \leftarrow$ GetY$(executionDB)$
    $X \leftarrow$ GetX$(executionDB)$

    BalanceDataset(X, y)

    $xtrain, xteste, ytrain, ytest \leftarrow$ SplitDataset$(X, y)$

    $clf \leftarrow$ StartStacking

    CalculateScore$(clf, xtrain, ytrain)$

    $model \leftarrow$ CalculateAccuracyScore$(clf, X, y)$

    $classifierLR \leftarrow$ EvaluateModel$(xtest, xtrain, ytest, ytrain)$

    $logisticPreds \leftarrow$ Comparing$(classifierLR, xtest, ytest)$

    GenerateConfusionMatrix$(ytest, logisticPreds)$
end function
```

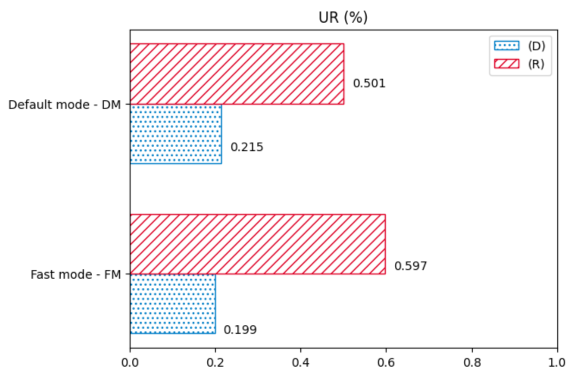
min, in databases (D) and (R), respectively, while DM took 156 h, equivalent to approximately six days for (D) and 128 h , equivalent to approximately five days for (R). Considering a proposal in which a model can be retrained with each new attack presented, FM can be a good choice due to its ability to train in a reduced time. Furthermore, analyzing the comparison of the prediction time of the two modes, as presented in 5(c),



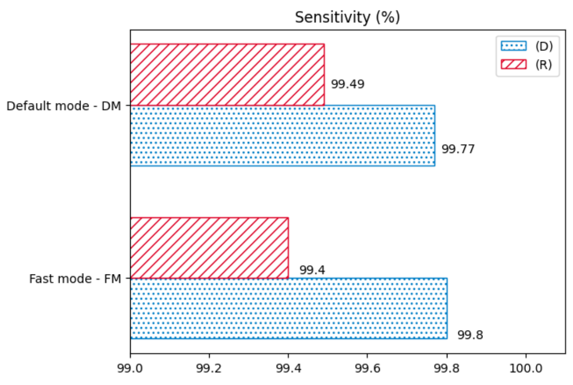
(a) Proposed models Accuracy in percentage.



(b) Proposed models FAR.



(c) Proposed models UR.



(d) Proposed models Sensitivity

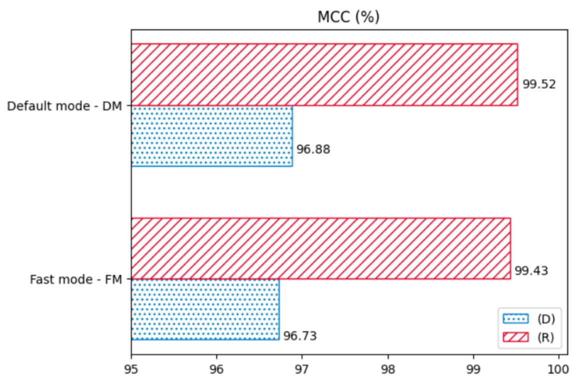
Figure 4. Solution performance regarding accuracy and FAR. FAR: False alarm rate.

the FM also stands out, since it concluded the prediction in 0.0067% and 0.0065% less than the DM using (D) and (R), respectively; these values may seem minimal, but they can be crucial in a critical network, such as an industrial one.

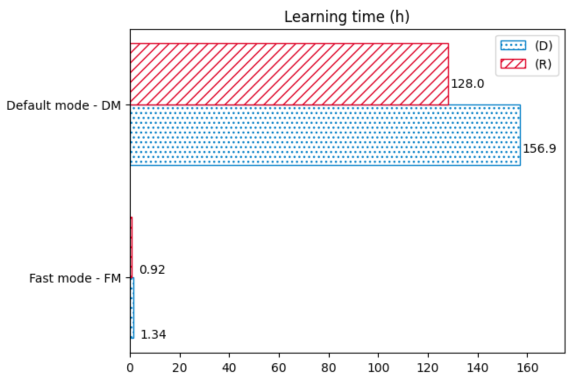
CONCLUSION

This study elucidates the methodologies employed to ascertain the optimal approach for crafting a resilient and effective model tailored for deployment in industrial networks using ML. Leveraging datasets from existing literature, pivotal aspects such as feature selection for model training and the comparative analysis of training methodologies using unbalanced and balanced datasets were thoroughly explored.

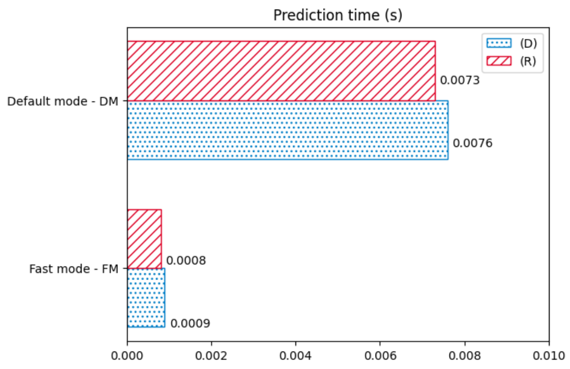
The culmination of our research efforts has led to the development of a specialized intrusion detection model designed for industrial networks, harnessing the power of EL. A significant focus was placed on anticipating potential scenarios that would necessitate model retraining in response to newly identified attacks, considering the time investment involved in model training. The model that emerged as the most promising in this study demonstrated an impressive accuracy rate of 99.93%, achieving training completion in a remarkably brief duration of 1 h and 34 min. In contrast, the model trained without the implemented treatments exhibited a slightly higher accuracy of 99.94% but required an extensive 156 h for training. Moreover, regarding prediction time, the superior model executed predictions in a swift 0.0009 seconds, while the alternative model required 0.0076 seconds for the same task. The marginal difference in accuracy between our proposed model and the comparison model is small and we recognize this difference. However, it is important to highlight that accu-



(a) Proposed models MCC.



(b) Proposed models learning time in hours.



(c) Proposed models prediction time in seconds.

Figure 5. Solution performance regarding models Sensitivity and models MCC. MCC: Matthews correlation coefficient.

racy is just one of many metrics to consider when evaluating an industrial network intrusion detection model. To help understand the performance and interpretability of the model, we also used other metrics to evaluate it, such as FAR, UR, Sensitivity, and MCC. The proposed model showed better results in all used metrics. The FSAnalysis selected the best features to build the proposed model. These findings underscore the effectiveness and practical applicability of the EL-based approach in developing a resilient intrusion detection model for industrial networks.

DECLARATIONS

Authors' contributions

Made substantial contributions to the conception and design of the study and performed data analysis and interpretation: Dalarmelina NDV, Arora P, Filho GPR, Meneguette RI, Teixeira MA

Availability of data and materials

Not applicable.

Financial support and sponsorship

None.

Conflicts of interest

All authors declared that there are no conflicts of interest.

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Copyright

REFERENCES

1. de O. Paula A, Meneguette RI, Gonçalves VP, Andrade AO, Peixoto MM, Rocha Filho GP Melhorando a integridade de sistemas de automação e comunicação em smart grids - uma arquitetura de combate a ciberataques. Available from: https://sol.sbc.org.br/index.php/wcge/articl e/view/24860 [Last accessed on 16 Apr 2024].
2. Cristiani AL, Lieira DD, Meneguette RI, Camargo HA. A fuzzy intrusion detection system for identifying cyber-attacks on IoT networks. In: 2020 IEEE Latin-American Conference on Communications (LATINCOM); 2020. pp. 1–6.
3. Maschi LFC, Pinto ASR, Meneguette RI, Baldassin A. Data summarization in the node by parameters (DSNP): local data fusion in an IoT environment. *Sensors* 2018;18:799. DOI
4. ISO/IEC 27000:2018(en). Information technology - security techniques - information security management systems - overview and vocabulary. Available from: https://www.iso.org/obp/ui/#iso:std:iso-iec:27000:ed-5:v1:en [Last accessed on 16 Apr 2024].
5. de Oliveira JA, Gonçalves VP, Meneguette RI, et al. F-NIDS - a network intrusion detection system based on federated learning. *Comput Netw* 2023;236:110010. DOI
6. Mazurczyk W, Caviglione L. Cyber reconnaissance techniques. *Commun ACM* 2021;64:86–95. DOI
7. Pastori Valentini E, Ipolito Meneguette R, Alsuhaim A. An attacks detection mechanism for intelligent transport system. In: 2020 IEEE International Conference on Big Data (Big Data); 2020. pp. 2453–61.
8. Murini CT. Análise dos sistemas de detecção de intrusão em redes: snort e suricata comparando com dados da darpa Available from: http://www.redes.ufsm.br/docs/tccs/CleberMurini.pdf [Last accessed on 16 Apr 2024].
9. Zargar GR, Baghaie T. Category-based intrusion detection using PCA. *J Inf Secur* 2012;3:259-71. DOI
10. do Vale Dalarmelina N, Arora P, Kaur B, Meneguette RI, Teixeira MA. Using ML and DL algorithms for intrusion detection in the industrial internet of things. In: AI, Machine Learning and Deep Learning. CRC Press; 2023. pp. 243–56.
11. Shafin SS, Karmakar G, Mareels I. Obfuscated memory malware detection in resource-constrained IoT devices for smart city applications. *Sensors* 2023;23:5348. DOI
12. Ambusaidi MA, He X, Nanda P, Tan Z. Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE Trans Comput* 2016;65:2986–98. DOI
13. Zolanvari M, Teixeira MA, Jain R; IEEE. *Effect of imbalanced datasets on security of industrial IoT using machine learning* 2018 IEEE International Conference on Intelligence and Security Informatics (ISI), Miami, FL, USA, 2018, pp. 112-117 DOI
14. Apruzzese G, Colajanni M, Ferretti L, Guido A, Marchetti M. On the effectiveness of machine and deep learning for cyber security. In: 2018 10th International Conference on Cyber Conflict (CyCon); 2018. pp. 371–90.
15. Shafin SS, Ahmed MM, Pranto MA, Chowdhury A. Detection of android malware using tree-based ensemble stacking model. In: 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE); 2021. pp. 1–6. DOI
16. Sarker IH, Abushark YB, Alsolami F, Khan AI. Intrudtree: a machine learning based cyber security intrusion detection model. *Symmetry* 2020;12:754. DOI
17. Teixeira MA, Salman T, Zolanvari M, Jain R, Meskin N, Samaka M. SCADA System Testbed for Cybersecurity Research Using Machine Learning Approach. Available from: https://arxiv.org/abs/1904.00753 [[Last accessed on 16 Apr 2024].
18. Pajouh HH, Javidan R, Khayami R, Dehghantanha A, Choo KKR. A Two-Layer Dimension Reduction and Two-Tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks. *IEEE Trans Emerg Top Comput* 2019;7:314– 23. DOI
19. Arya L, Gupta GP. Ensemble Filter-based Feature Selection Model for Cyber Attack Detection in Industrial Internet of Things. In: 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS). vol. 1. IEEE; 2023. pp. 834–40.
20. Mohy-Eddine M, Guezzaz A, Benkirane S, Azrour M, Farhaoui Y. An ensemble learning based intrusion detection model for industrial IoT security. *Big Data Min Anal* 2023;6:273–87. DOI
21. Verma A, Ranga V. Machine learning based intrusion detection systems for IoT applications. *Wireless Pers Commun* 2020;111:2287–310. DOI
22. Khoei TT, Aissou G, Hu WC, Kaabouch N. Ensemble learning methods for anomaly intrusion detection system in smart grid. In: 2021 IEEE International Conference on Electro Information Technology (EIT); 2021. pp. 129–35.
23. Liang W, Li KC, Long J, Kui X, Zomaya AY. An industrial network itrusion detection algorithm based on multifeature data clustering optimization model. *IEEE Trans Ind Inform* 2020;16:2063–71. DOI
24. Khan IA, Pi D, Abbas MZ, Zia U, Hussain Y, Soliman H. Federated-SRUs: a federated-simple-recurrent-units-based IDS for accurate

detection of cyber attacks against IoT-augmented industrial control systems. *IEEE Internet Things J* 2023;10:8467–76. DOI 25.
Dalarmelina NDV, Teixeira MA, Andrade FRH, Júnior LAP, Filho GPR, Meneguette RI. FSAnalysis: a feature selection and analysis mechanism considering balanced and unbalanced bases. In: 2022 17th Iberian Conference on Information Systems and Technologies (CISTI); 2022. pp. 1–7.

26. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: machine learning in python. Available from: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf [Last accessed on 16 Apr 2024].
27. Cohen J. Statistical power analysis. Available from: https://journals.sagepub.com/doi/abs/10.1111/1467-8721.ep10768783 [Last accessed on 16 Apr 2024].