International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Developing an Intrusion Detection System (IDS) For Network Security Using Machine Learning

Ghousun Ayed Alsharari[1], Ayman Mohamed Mostafa[2,3]

Submitted:12/03/2024 **Revised**: 27/04/2024 **Accepted**: 04/05/2024

Abstract: The Internet of Things (IoT) technologies have become so widespread that they are now the main cause of network security management problems. With the ever-growing integration of IoT into both consumer and industrial applications, security plays a very important role. The research in this paper is about the creation of a modern Intrusion Detection System (IDS) that can be applied to IoT network security using Machine Learning (ML) and Deep Learning (DL). The very heart of this system is the use of ML to discover and deal with possible threats in a quick and efficient way, though the focus here is on feature engineering that enhances detection accuracy. Through the use of mRMR for feature selection and PCA for feature reduction, our system is able to optimize the processing and analysis of network behaviors so that it can differentiate between normal operations and different types of attacks efficiently. This paper is about the comparison of effectiveness between ML and DL models in detecting threats inside IoT environments, which are tested on the ToN-IOT dataset. The findings show that our ML and DL-driven IDS not only have a high level of accuracy in threat detection but also significantly reduces the computational demands, therefore enabling more efficient real-time applications in IoT security.

Keywords: *Internet of Things (IoT); Intrusion Detection System (IDS); Machine Learning (ML); Deep Learning (DL); and Minimum Redundancy Maximum Relevance (mRMR).*

1. Introduction

The Internet of Things (IoT) is the network of physical things — "things" — which are equipped with sensors, software and other technologies that aim to connect and exchange data with other devices and systems through the Internet. These devices fall in the category from home appliances to complex industrial machines. In the industrial field, IoT technology is a basic element of the industrial Internet of Things (IIoT), which makes it possible to achieve extremely high levels of efficiency, productivity and performance. It is widely applied in the manufacturing processes to control machine operations, coordinate supply chain management and perform predictive maintenance. In the medical field, IoT has led to the Internet of Medical Things (IoMT), thus changing healthcare by enabling the remote monitoring of patients, which makes it possible for doctors to intervene in time and improve clinical treatment outcomes. The IoT devices in healthcare are wearable fitness bands and advanced units like heart rate monitoring cuffs [1].

The architecture of the Internet of Things (IoT) is formed by three main layers, each playing an important role in the operation of IoT systems. The Perception Layer, also known as the physical layer, is made up of sensors and actuators which collect all environmental parameters and interact with the real world. Then the Network Layer becomes the communication spine, which connects these sensors and actuators to the internet through technologies such as Wi-Fi, Bluetooth, and LTE; this is what enables the transmission and processing of collected data. At the top is the Application Layer, which uses the processed data to create personalized services for users and it supports a range of applications from business analytics to medical diagnostics, thus enabling people to make decisions based on real-time IoT data.

The IoT networks have many security issues because of the large number of connected devices which create multiple weaknesses and ways for cyberattacks. The risks are the Data Breaches, where unauthorized people get into data either in transit or at rest within the network, which can lead to serious information leak. Man-in-the-Middle Attacks is another threat in which attackers intercept and change the communication between two people who have no idea about it, thus they compromise the integrity of data exchange. Besides, Denial of Service Attacks will make the system collapse by an excess load of data so that legitimate users do not get service and the operations in the network are disrupted. These weaknesses force the IoT ecosystems to be protected by strong security measures from such threats [3, 4].

To guard IoT networks against the above-mentioned security hazards, the deployment of IDS that works well is a must. IDS fall into three main categories: HIDS, which are

[1] *Department of Computer Science College of Computer and Information Sciences Jouf University, Sakaka 72388, Saudi Arabia*
Email: 441205000@ju.edu.sa
[2] *Department of Information Systems College of Computer and Information Sciences Jouf University, Sakaka 72388, Saudi Arabia*
Email: amhassane@ju.edu.sa
[3] *Department of Information Systems, College of Computers and Informatics – Zagazig University - Egypt*

installed on individual devices to monitor both the traffic entering and exiting the device as well as system logs for signs of malicious activity; NIDS, which check the entire network's traffic to find unusual or suspicious behavior; and Hybrid IDS that combine features of both HIDs and NIDS to provide a more comprehensive coverage. This security system has many layers that help in the identification and reduction of potential threats at both the device and network levels which increases the resistance level of IoT systems against cyber-attacks [5]. Machine learning boosts IDS by letting systems learn from data patterns and thus detect anomalies more accurately. Feature engineering is the main factor here that helps to identify which data characteristics are most important for detecting threats [6].

This research deals with the security issues in IoT networks that are innate by building a new Intrusion Detection System (IDS) which exploits machine learning and deep learning. The main objective is to develop a smart IDS system that combines different ML models and compares their performance with the DL model in detecting and dealing with possible security threats. The gist of our method is the feature engineering, which consists mainly of two parts: feature selection and feature reduction. We evaluate the effect of feature engineering on the IDS by measuring the performance of various ML and DL models, both with and without Minimum Redundancy Maximum Relevance (mRMR) for feature selection and Principal Component Analysis (PCA) for dimensionality reduction. The IDS's efficiency is thus increased by these feature engineering techniques and at the same time, the computational resources and time required to analyze IoT network behaviors are significantly decreased. This fast way lets our system to precisely tell the difference between the usual operations and possible security threats in both binary levels (normal or attack) and multiple levels (normal and various kinds of attacks).

2. RELATED WORK

IDS can be broadly categorized into two main types: The Network Intrusion Detection Systems (NIDS) and Host Intrusion Detection System (HIDS). NIDS are intended to watch the communication between nodes in a network, analyzing both incoming and outgoing traffic flow that includes packet headers and payload content [7]. On the contrary, HIDS concentrates on what is going inside every node or system like operating system files and log files and also can analyze encrypted network communication that either comes from or goes to the host [8].

Besides, IDS can be divided into three types: signature-based, anomaly-based and specification-based [9]. Signature-based IDS, also known as Misuse Detection, are based on the predefined signatures of known intrusion patterns. They are good at identifying the known attacks, but they can also be deceived by new, undefined ones [9]. Anomaly-based IDS, or Behavior-based Detection, determine the discrepancies from the normal behavior patterns which are set. Those systems, usually based on AI techniques like Machine Learning (ML) and Deep Learning (DL), can both detect known and unknown attacks, but they have a higher False Positive Rate (FPR) [9]. Specification-based IDS is a combination of the advantages of both signature-based and anomaly-based systems, which aims to detect a wide range of attacks by using different AI techniques [10].

The figure 1 in the paper is a kind of comparison between signature-based and anomaly-based IDS, which makes it clear that either stateless or stateful operational mode can be employed by these systems, and what types of network intrusions they are good at detecting.

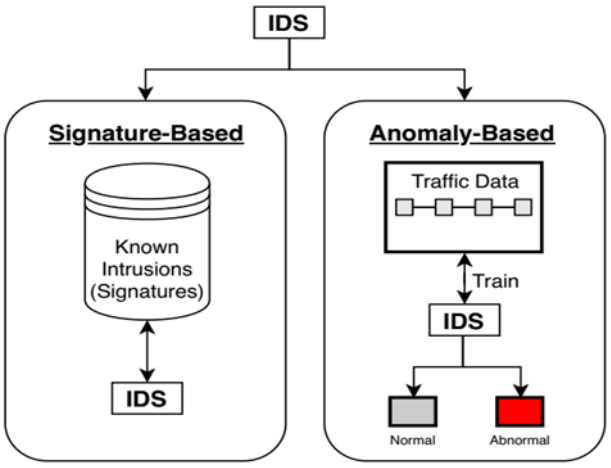


Fig 1: Signature-based versus Anomaly-based IDS**.**

This classification of IDS is very important for the comprehension of their use in healthcare cybersecurity especially when it comes to IoT-based ICUs. Although IDS are the ones that detect intrusions, it is necessary to separate them from Intrusion Prevention Systems (IPS), which can apply corrective and preventive measures [10]. The research of IDS in the healthcare field, particularly with the growing use of IoT technologies, is what our study is based on. We

are aiming at improving cybersecurity in critical healthcare areas.

The research by Chandola et al. [11] is a detailed study of the application of machine learning in anomaly detection, mainly in intrusion detection systems (IDS). Machine learning is applied in misuse detection, where classifiers are trained with pre-labeled datasets to identify certain types of intrusions. Anomaly detection is done through unsupervised learning where the system learns from data without pre-existing labels to identify deviations from the norm. Semi-supervised learning is like a mixture of both, it uses some labeled data to boost the learning process.

Yu Xue and his colleagues [12] introduced a highly developed feature selection method for the Intrusion Detection Systems (IDS) based on the self-adaptive differential evolution (SaDE) algorithm and k-nearest neighbors (k-NN) for evaluation. The KDDCUP99 dataset, which is the standard in IDS research was divided into four smaller subsets for feature selection and classification. The k-NN's classification accuracy was checked by putting together three of the four subsets to test if the feature selection is stable in different parts of the data.

Akashdeep et al. in their study [13] deal with the application of Artificial Neural Networks (ANNs) to Intrusion Detection (ID), mainly showing that feature selection and reduction are crucial for improving system performance. The methodology is to rank the data features according to their correlation and information gain, then combine them properly and preprocess redundant and irrelevant data in order to optimize the resource usage. Thus, time complexity will be reduced. Nevertheless, the study's use of the KDD99 dataset makes us wonder about its relevance and reliability in today's network security problems.

Simone A. Ludwig's [14] investigation on DNNs in the ensemble methods for IDS is a great contribution to cybersecurity. The NSL-KDD dataset, which is a better version of the KDD'99 was used to test IDS models. The research discovered that some attacks although were highly performing, they also revealed the need for more studies and development of new techniques to improve the detection capabilities especially in U2R and R2L types which are considered to be complex.

The research conducted by Chaopeng Li [15] on the Intrusion Detection (ID) using Recurrent Neural Networks (RNNs) and Restricted Boltzmann Machines (RBMs) is a great step forward in cybersecurity methods. The approach utilizes the network traffic data that includes potential signatures of nefarious activities as input data. RBMs are applied to the construction of network packet representations, which take in the main features of data for analysis. RNNs are good at processing the sequential data, which is why they are perfect for analyzing the temporal correlations between neighboring network packets.

Chuanlong Yin's [16] study on the use of Recurrent Neural Networks (RNNs) for Intrusion Detection Systems (IDS) shows the possibility of deep learning in cybersecurity. The research works on the NSL-KDD dataset to transform non-numeric data into a numeric one for better processing and analysis of the data. The model shows good results in detecting various attacks, however it has some limitations in the detection of User to Root (U2R) and Remote to Local (R2L) which are often complex and low-frequency. Training time is another limitation, since RNNs can take a long time to train, especially when the dataset is big like NSL-KDD. Model complexity and overfitting are also the problems. RNNs are inherently complex models that tend to be overfitted if not properly regularized or trained on datasets which do not represent the full spectrum of possible intrusions. Real-time IDS deployment of RNNs could be hard to do because their computation is very intensive which might not be possible in all network environments.

Yihan Xiao's study [17] comes with an Intrusion Detection System (IDS) using Convolutional Neural Network (CNN), a major cybersecurity strategy. The model is evaluated however it is outperformed by more than one other approach in the field. Nevertheless, it has some drawbacks such as computational complexity, the risk of overfitting, the problem of generalization to unseen attacks, real-time processing difficulties and compatibility with existing systems.

Dimitrios Papamartzivanos's [18] "rule injection method" for Intrusion Detection Systems (IDS) is a new way that combines Decision Trees (DT) with Genetic Algorithms to come up with an efficient IDS. The method is used to improve the classification performance of Decision Trees in IDS with GAs optimization power. The main results prove that the IDS detection abilities can be improved by generating a strong set of rules with the help of GAs. Nevertheless, the issues related to computational requirements, generalization abilities and practical application in real-world situations are still the key areas that need more research and improvement.

Kai Peng and his team [19] have come up with an IDS for big data environments which is based on Decision Trees (DT) as the core. The IDS is successfully dealing with and categorizing the big datasets through a structured way, which proves that machine learning techniques are really flexible to handle complex, large-scale data. Nevertheless, the method's scalability, adaptability to new threats, computational efficiency, overfitting risks and model complexity are all limitations.

Gisung Kim and the team [20] have built a hybrid intrusion detection model which fuses together the concepts of misuse detection and anomaly detection via Decision Trees (DT)

and Support Vector Machines (SVM). The model initially concentrates on misuse detection using the C4. 5 Decision Tree algorithm, which is based on the segmentation of normal training data into smaller subsets for certain misuse detection cases. In the second phase, binary SVMs are used on each subset formed by the DT model to add more sensitivity of unknown or new intrusions.

Hassan I. Ahmed and his colleagues [21] have carried out the full research on machine learning techniques in Intrusion Detection Systems (IDS), analyzing Decision Trees (DT), Artificial Neural Networks (ANN) and Random Forest. The main datasets that were used are KDDCUP99, ISC2012, and CICIDS2017. The comparative analysis was very useful as it helped to understand how different machine learning algorithms work in IDS, thus pointing out the strong and weak sides of each method when dealing with various types of intrusions. Nevertheless, the research had a number of drawbacks, for instance dataset limitations, generalization concerns, model complexity and scalability problems as well as overfitting risks.

Wajdi Alhakami and his team have come up with a new way for anomaly-based Intrusion Detection (ID) in the field of Internet of Things (IoT) security. The model does both feature selection and classification which is very important for efficient ID, especially in complex environments like IoT. The model's capability is tested with three different datasets (KDDCUP99, Kyoto2006+, and ISC2012), which prove that the model can be applied in any network environment.

Abdulhammed and others [23] have proposed a technique to improve the efficiency of Intrusion Detection Systems (IDS) by means of feature reduction using autoencoders and principal component analysis (PCA). The autoencoders and PCA duo can boost the IDS performance by simplifying data analysis and making classifiers more efficient in detecting intrusions. Nevertheless, the way must cope with issues associated with complexity, data loss, generalization and adaptability in order to be widely applicable in various IDS contexts.

Xiao et al. [24] created an IDS that is more advanced than the previous ones by using a Bayesian Network Model Averaging (BNMA) classifier which combines BMA and k-best BN classifiers to improve classification accuracy. Nevertheless, the decision of which feature subset to use is crucial for the performance of the classifier and thus it determines the accuracy. The model averaging is a complex task and it can be computationally intensive, especially when the datasets are large like in NSL-KDD. Another problem is the overfitting risk which happens, when you train and test on different subsets of the same dataset multiple times. Discretization bias is caused by the human factor in deciding how to group continuous variables.

Wenying Feng et al. [25] combined an ant colony optimization (ACO) algorithm with a Support Vector Machine (SVM) to come up with a new IDS. The integrated apporach is better than using these methods one by one and it works well in classfying different types of network intrusions. Nevertheless, the drawbacks are related to dependence on clustering quality, computational complexity, dealing with evolving threats, generalization to new data, parameter tuning and overfitting risk.

Table 1: Comparison between reference studies

REF	Approach Method	Detailed Summarize Method	Machine Learning Method	Key Limitations
Chandola et al. [11]	Anomaly Detectors	Comprehensive analysis of ML techniques for ID, covering supervised, unsupervised, and semi-supervised learning, focusing on anomaly and misuse detection.	Supervised & Unsupervised Learning	Need for labeled datasets, generalization issues, overfitting
Yu Xue et al. [12]	Feature Selection	Utilization of the SaDE algorithm for optimal feature selection and k-NN to evaluate the quality of	SaDE Algorithm & k-NN	Lack of generalizability to entire dataset

		these features in IDS.		
Akashdeep et al. [13]	Feature Selection & Classification	Strategic ranking and combination of data features using ANNs to build a classification system optimized for ID.	Artificial Neural Networks (ANN)	Dependency on KDD99 dataset, overfitting risk
Simone A. Ludwig [14]	Ensemble Methods	Evaluation of DNNs in ensemble methods focusing on backpropagation neural networks and assessing false alarm and detection rates.	Deep Neural Networks (DNNs)	Less effective in U2R and R2L attacks
Chaopeng Li [15]	Traffic Data Analysis	Use of RNNs and RBMs to analyze network traffic data, capturing temporal correlations and building packet representations.	RNNs & RBMs	Model complexity, overfitting, adaptability
Chuanlong Yin [16]	Binary & Multi-Class Classification	RNNs applied to IDS for binary and multi-class classification tasks, assessing performance on accuracy, detection rate, and FP rate.	Recurrent Neural Networks (RNNs)	Shortcomings in U2R/R2L detection, training time
Yihan Xiao [17]	Data Transformation & Model Training	Transformation of symbolic data to numeric and training of CNN model to enhance intrusion detection performance.	Convolutional Neural Network (CNN)	Surpassed by other methods
Dimitrios Papamartzivanos [18]	Rule Injection	Synergy of Decision Trees and Genetic Algorithms to create a rule-based intrusion detection model,	Decision Trees & Genetic Algorithms	Complexity, overfitting, generalization

		optimizing detection rules.		
Kai Peng et al. [19]	Data Preprocessing & Classification	Processing and classifying big data in IDS using DT, involving pre-processing, normalization, and classification steps.	Decision Trees (DT)	Scalability, adaptability, efficiency
Gisung Kim et al. [20]	Misuse & Anomaly Detection	Hybrid model using DT for misuse detection and binary SVMs for anomaly detection, focusing on segmenting and classifying network data.	Decision Trees & SVM	Complexity, generalization, detection capabilities
Hassan I. Ahmed [21]	Comparative Analysis	Analysis of various ML techniques for IDS, comparing their performance across different datasets.	DT, ANN, Random Forest	Dataset limitations, generalization, adaptability
Wajdi Alhakami [22]	Feature Selection & Classification	Bayesian methods used for both feature selection and classification, tailoring the model for IoT ID systems.	Bayesian Methods	Complexity, scalability, adaptability
Abdulhammed et al. [23]	Feature Reduction	Employing autoencoders and PCA for dimensionality reduction in IDS, enhancing classifier efficiency.	Autoencoders & PCA	Complexity, data loss, adaptability

Xiao et al. [24]	Classifier Development	Development of BNMA classifier through BMA and k-best BN classifiers, enhancing classification accuracy in IDS.	Bayesian Network Model Averaging (BNMA)	Feature selection, model complexity
Wenying Feng et al. [25]	Data Clustering & Classification	Combination of ACO for clustering and SVM for classification, improving intrusion detection in IDS.	Ant Colony & SVM	Clustering quality, computational complexity
Enamul Kabir et al. [26]	Sampling & Detection	Two-stage decision-making using sampling for representative data selection and LS-SVM for intrusion detection.	Least Square SVM (LS-SVM)	Sample representativeness, initial subgrouping
Vijayanand et al. [27]	Feature Selection & Classification	GA for feature selection and multiple SVM classifiers for sequential data processing and classification in mesh IoT networks.	SVM & Genetic Algorithm	Complexity, specificity to mesh IoT
Kuang et al. [28]	Feature Reduction & Optimization	SVM model integration with KPCA for feature reduction and CPSO for parameter optimization in IDS.	SVM, KPCA, & CPSO	Optimization complexity, dataset limitations
Bamakan et al. [29]	Feature Selection & Optimization	Focus on feature selection and SVM parameter optimization in IDS to maximize detection rate and minimize false alarms.	Feature Selection & SVM Optimization	Limitations in R2L and U2R detection

Enamul Kabir et al. [26] proposed a new method to IDS that is based on the sampling technique and the Least Square Support Vector Machine (LS-SVM). The two-stage decision-making process begins by dividing the dataset into subgroups, then selecting samples that are representative and truly reflect the overall characteristics of the dataset. The LS-SVM is then used to the samples to find intrusions. The technique is tested on the KDDCUP99 dataset, which is a well-known benchmark in intrusion detection.

Vijayanand et al. [27] invented the new way of IDS for mesh IoT networks, which is a combination of SVM and GA. The

model is created for the mesh IoT networks, and it focuses on its importance in the modern connected network environment. Nevertheless, the model's real-world application could be affected by factors like computational complexity, the linear arrangement of classifiers and its adaptability to changing network threats and environments.

SVM and KPCA were integrated by Kuang et al. [28] in their novel method of IDS along with the improved CPSO to make it more efficient. The model applies SVM together with KPCA for the feature dimensionality reduction, which is then fine-tuned to minimize computational time and enhance accuracy. Nevertheless, the possible drawbacks of this method are the difficult optimization process, dependence on KPCA for feature reduction, data set limitations, potential overfitting. Computational resource requirements as well as adaptability to changing threats and generalization across different network environments are also among them.

The paper of Bamakan et al. [29] presents an Intrusion Detection System (IDS) framework which merges the feature selection and Support Vector Machine (SVM) parameter optimization techniques. The framework tries to have a high detection rate and at the same time avoid false alarms as much as possible using the least features. Tested on the NSL-KDD and KDDCUP99 datasets, the framework showed significant reductions in false alarm rates and reasonable detection rates, especially for Probe and Denial of Service (DoS) attacks. However, it showed limitations in detecting R2L and U2R attacks, which often involve more sophisticated intrusion methods. The framework's performance is evaluated on widely used datasets, which may not fully represent the current network attack landscape. Table 1 provides a comparison between previous studies.

Our work seeks to address several key gaps in the field of Intrusion Detection Systems (IDS) for the Internet of Things (IoT) by leveraging advanced machine learning (ML) and deep learning (DL) techniques:

- Comparative Analysis of ML and DL Models: we are exploring how different ML models stack up against DL models in terms of effectiveness in detecting and responding to security threats within IoT environments.
- Efficient Data Processing with Feature Engineering: By implementing MrMr based feature selection and PCA based feature reduction for feature selection, our work significantly reduces the dimensionality of data, enhancing the IDS's performance and lowering the computational demands, which is crucial for real-time applications in IIoT.
- Multi-level Threat Detection: Unlike many existing systems that only categorize activities as normal or malicious, our IDS is designed to identify various levels of threats, providing a nuanced and effective security mechanism.

3. METHODOLOGY

The proposed system framework consists of a recorder, an AI-based intrusion detection system, and a decision-making module. The recorder records IoT network behavior, while the AI-based system analyzes network properties to identify patterns. The final component makes decisions based on the assessed behavior, determining if the observed behavior is normal or a potential attack. A visual representation of these components is provided in Figure 2.

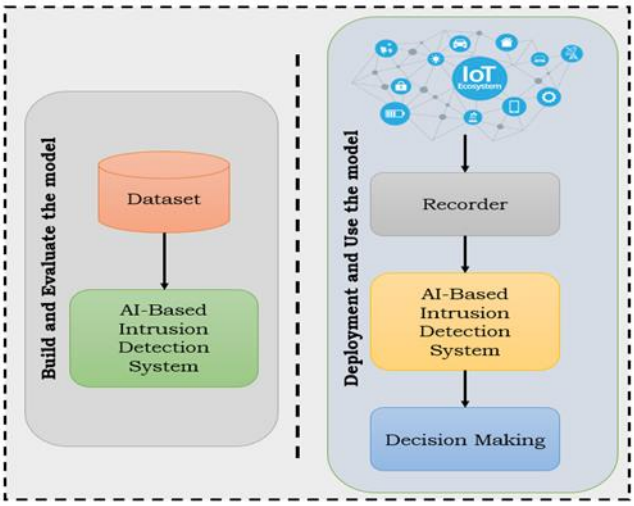


Fig 2: Components of the framework.

The proposed model for an intrusion detection system structured as sequential steps as depicted in Figure 3:

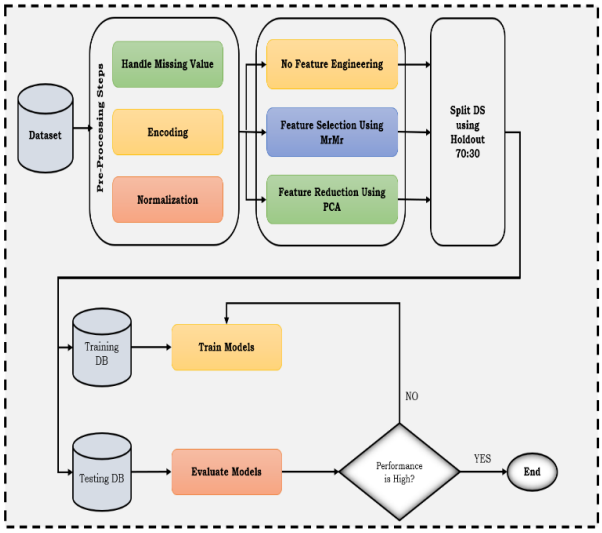
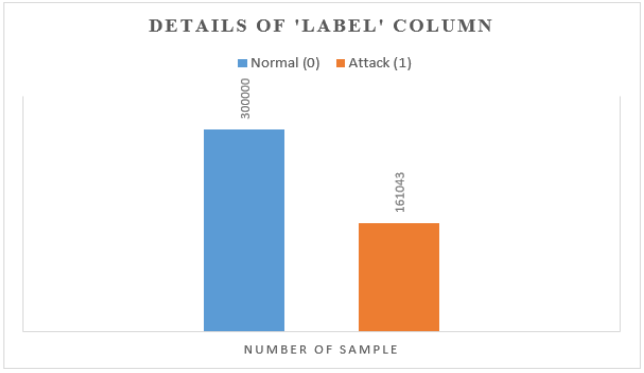
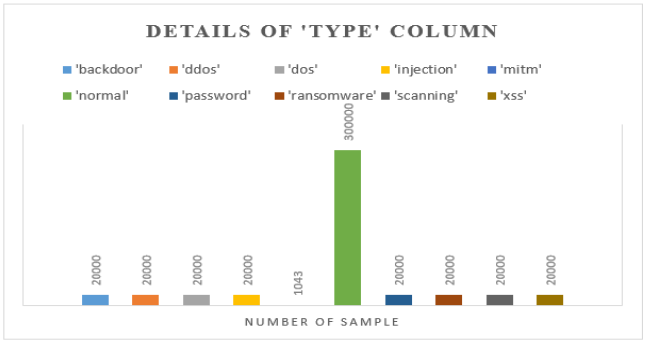


Fig 3: Proposed model of our smart IDS.

Step 1: Choose the Dataset: In our work, we will use ToN-IOT dataset of network traffic from an industrial IoT application case. The ToN-IOT [30] are newer generations of Internet of Things IoT and Industrial datasets to assess the accuracy as well as efficiency of different cybersecurity applications derived from Artificial Intelligence. The dataset consists of 461,034 records with a total of 52 features which are used in the machine learning model and work as functioning as an Intrusion Detection System (IDS). It includes two labels: 'label' indicating whether the record represents normal behavior or an attack (0 for normal, 1 for attack), and 'Type', classifying records to specific categories of attacks such as DoS; DDoS; and backdoor alongside normal data. The dataset had ten classes, where nine of them presented distinct type attacks, as well as one class, described the normal condition. This is a rich structure to be used in developing a sufficiently strong IDS system able of differentiating between various network activities and identifying threats properly. The statistical descriptions of Label and Type columns are depicted in Figure 4.



a



b

Fig 4: Details of (a) Label and (b) Type columns**.**

Step 2: Data Pre-processing: The preprocessing stage includes the following steps:

a. Handle Missing Values: To ensure the dataset is complete without gaps, which might lead to biased

or incorrect model predictions. Knowing that Numeric columns with missing values are filled using the mean value of the column, which preserves the central tendency. Whereas, categorical columns are filled using the mode value, which is the most frequent category, ensuring that the original distribution of categories is maintained as closely as possible [31]. The dataset contains 9476837 missing values in different columns as shown in Table 2.

Table 2: number of missing values for each column.

Column	Number of Missing
'service'	280216
'dns_query'	366019
'dns_AA'	365158
'dns_RD'	365158
'dns_RA'	365158
'dns_rejected'	365158
'ssl_version'	460737
'ssl_cipher'	460737
'ssl_resumed'	460352
'ssl_established'	460352
'ssl_subject'	461034
'ssl_issuer'	461034
'http_trans_depth'	460796
'http_method'	460809
'http_uri'	460809
'http_version'	460801
'http_user_agent'	460809
'http_orig_mime_types'	461029
'http_resp_mime_types'	460883
'weird_name'	459749
'weird_addl'	460290
'weird_notice'	459749

b. One-Hot Encoding: Machine learning models generally work better with numerical input. Categorical data are transformed into a numerical format that models can interpret without introducing ordinality [32].

c. Min-Max Normalization: To scale numerical features in the dataset to a common scale without distorting differences in the ranges of values [32]. This rescales the feature to a fixed range of 0 to 1, using the equation (1):

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (1$$

(1)

Step 3: Feature Engineering is crucial for enhancing model performance and efficiency. It utilizes two main techniques: Principal Component Analysis (PCA) and Maximum Relevance Minimum Redundancy (mRMR). PCA minimizes the number of features, therefore simplifying the dataset while mRMR chooses only those most impactful and informative from this reduced set.

- PCA is used for feature reduction, where it changes the original features into a new set of variables which are called principal components. These elements are made to get the most variance or information from the data while being uncorrelated with each other. The reduction of the data thus not only simplifies it but also helps in getting rid of noise and enhancing computational [illegible]cy [33].
- On the contrary, mRMR is applied to feature selection which emphasizes on choosing features that are most relevant to the predictive task and at

the same time ensuring that there is a minimum redundancy in them. This procedure selects those features that are the most informative about the target variable and at the same time, they are not related to each other. Thus, it prevents a model from learning any redundant information. This contributes to the increase of the predictive accuracy and generalization capability of the model [34].

Step 4: Split DS using Holdout: The dataset is split into training (70%) and testing (30%) sets. The training set is used to build and train the model, while the testing set is used purely for evaluating its performance, simulating how the model would perform on unseen data.

Step 5: Classification: In our work we use three levels of classification:

- First Level - Normal or Attack: Determine whether each instance in the dataset is a normal operation or an anomalous attack.
- Second Level - Type of Attack: Further classify each detected attack into types, for instance, DOS, malware, or phishing, based on the characteristics of the attack.

Regarding classifier, we used three types of classifiers:

- Ensemble Classifiers: Use methods like Bagging Trees and AdaBoost, which combine the predictions of several base estimators to improve robustness and accuracy over a single estimator.
- Standard Classifiers: Use the algorithms like Support Vector Machine, K Nearest Neighbor, Naive Bayes and Decision Trees that have their own strengths in dealing with different types of data and classification problems.
- Deep Neural Networks: Use the complicated models like LSTM which are good at processing sequences and time-series data, thus they are suitable for tasks such as anomaly detection in time-dependent data.

Step 6: Assessment: The model is to be tested on the new, unseen data which are practical and accurate for it. This is achieved through the use of different indicators like accuracy, precision, recall, F1-Measure and ROC curves to assess the model's performance from various aspects so that it meets the set standards for deployment.

A. Classification Algorithm

AdaBoost (Adaptive Boosting) [35] is a technique that uses multiple weak learners to produce a strong classifier. The ensemble, usually a single decision tree for each learner is constructed sequentially and the next one focuses on those instances which were not classified correctly by the previous ones. The learners are given weights according to their accuracy and each instance in the dataset is also assigned a weight that increases if it is misclassified. As a result, the ensemble becomes able to adjust itself to the harder cases in the dataset. AdaBoost is specialized in binary classification problems, and it is famous for its high accuracy of classification.

Bagging Trees (BagTree) or Bootstrap Aggregating [36], is a different ensemble technique in which multiple decision trees are trained on the same training set but on its different subsets. These subsets are produced by the random selection of some training samples with replacement. Every tree is trained separately, and their predictions are then combined usually by the majority voting for classification or averaging for regression. This technique minimizes the variance and thus, the chance of overfitting which in turn makes the ensemble more robust than individual decision trees.

Support Vector Machine (SVM) [37] is a very efficient classification method that does it by finding the hyperplane which best separates two classes of data with the maximum margin. In a nutshell, it determines the broadest "street" between classes. SVM is very good in high-dimensional spaces, and it can be used for both linear and non-linear boundaries by the kernel trick.

K-Nearest Neighbors (KNN) [38] is an easy, instance-based learning algorithm. In KNN, the classification of a new instance is determined by a plurality vote of its neighbors, with the instance being assigned to the class most common among its k nearest neighbors measured by a distance function. KNN is easy to implement and understand but can become computationally expensive as the size of the data grows.

Decision Tree (DT) classifiers [39] are intuitive models that split data by learning decision rules inferred from the features. Trees are formed by nodes representing tests on features and leaf nodes representing classes. Decision trees are easy to interpret and can handle both numerical and categorical data but are prone to overfitting, especially with complex trees.

Naive Bayes (NB) classifiers [40] are probabilistic models that apply Bayes' Theorem, assuming strong (naive) independence between the features. They are particularly well-suited for classification tasks where high dimensionality is present, such as text classification. Despite their simplicity and the naive assumption, Naive Bayes classifiers often perform very well under many complex real-world situations.

Long Short-Term Memory (LSTM) [41] networks are a type of recurrent neural network (RNN) suitable for sequence prediction problems. Unlike standard feedforward neural networks, LSTMs have feedback connections and

can process data sequences irrespective of input sequence length They are of great help in those cases where the context from the input data is very important, for example, speech recognition or anomaly detection in time series data. LSTMs solve the problem of vanishing gradient that is common to RNNs, so they can learn longer dependencies.

Our LSTM model configuration is designed to effectively handle sequence classification tasks and consists of following layers:

- Input Layer: The model processes sequences where each sequence element has only one feature.
- LSTM Layer: The LSTM layer has 200 hidden units, providing it with substantial capacity to learn from the data by capturing complex temporal dependencies within the sequence. It's configured to output only the last hidden state, which is typical for sequence-to-label tasks where the final state represents the culmination of learned temporal features relevant to the prediction task.
- Fully Connected Layer: This layer has as many neurons as there are unique classes in the dataset, ensuring each class can be predicted.
- Softmax Layer: Following the fully connected layer, the softmax layer normalizes the output to a probability distribution over the predicted classes, making the model's output interpretable as class probabilities.
- Classification Layer: The final layer computes the cross-entropy loss during training, which helps in optimizing the model by comparing the predicted output with the true class labels.

The LSTM model utilizes the Adam optimizer with settings optimized for GPU execution. Training is now set to 10 epochs at a learning rate of 0.001. Verbose output and progress plots are enabled to monitor the training process. This setup strikes a balance between achieving convergence in a reasonable number of training cycles and providing detailed insights into training progress through visual feedback.

Each of these models has its strengths and particular contexts where it excels as shown in Table 3, making them suitable for various kinds of data and predictive modelling challenges.

Table 3: Advantage and Disadvantage of each used model.

Classifier	Advantages	Disadvantages
AdaBoost	Enhances the accuracy of weak learners, robust to overfitting, less prone to overfitting than DT.	Sensitive to noisy data and outliers, performance depends on data and weak learner.
Bagging Trees	Reduces variance, less prone to overfitting, works well for high dimensional data.	Computationally intensive, less interpretable, not the best choice for very large datasets.
SVM	Effective in high-dimensional spaces, works well with clear margin separation, versatile (kernel methods).	Requires full data in memory, can be slow to train, choosing a correct kernel can be challenging.
KNN	Simple and effective, no training period, naturally handles multi-class cases.	Slow query time as dataset grows, sensitive to irrelevant features, needs homogeneous features.
Decision Trees	Easy to interpret and explain, can handle both numerical and categorical data.	Prone to overfitting, can be unstable, sensitive to small changes in the data.
Naive Bayes	Fast and efficient, performs well with large datasets, good baseline for text classification.	Assumes feature independence, poor estimates of probability can be a drawback.
LSTM	Excellent for sequential data, can model time series data, capable of learning long-term dependencies.	Computationally intensive, difficult to train, and can suffer from overfitting without proper regularization.

(2)

4. PERFORMANCE MEASURES

Classification accuracy serves as an effective metric to gauge learning performance of proposed models [42-44]. The evaluation utilizes a standard confusion matrix technique, with the following metrics listed below:

Confusion Matrix: A confusion matrix provides a concise summary of prediction outcomes in classification problems. It offers insight into classification accuracy by

distinguishing between correct and incorrect predictions as well as errors made within each class, distinguishing between true positives (true negatives) and misclassifications (false positives and false negatives). Diagonal elements indicate this information while off-diagonal ones represent misclassification errors like false positives/false negatives respectively.

Accuracy can be defined as the percentage of correctly classified samples when measured values are compared with known ones; expressed using equation (2):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

Where true positive (TP), true negative (TN), false positive (FP), and false negative (FN) represent key components of confusion matrices.

Error Rate measures the proportion of misclassified samples within the dataset as defined by equation (3). Ultimately, lower error rates mean better model performance.

$$Error\ Rate = \frac{FP + FN}{TP + FP + FN + TN} \quad (3)$$

Precision measures the model's ability to correctly classify positive values as shown by equation (4):

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

Recall measures the proportion of true positives (instances that correctly identify a particular class) among all positives, both true positives and false positives combined. As shown by equation (5):

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \quad (5)$$

Specificity refers to a model's ability to predict negative values accurately as indicated by equation (6):

$$Specificity = \frac{TP}{TN + FP} \quad (6)$$

Area Under the Curve (AUC) serves as an evaluation metric of binary classification models. It measures the area beneath a Receiver Operating Characteristic curve which illustrates true positive rates versus false positive rates at various classification thresholds; an AUC value of one denotes ideal classification while one or less indicates random classification.

Receiver Operating Characteristic (ROC) curves are visual illustrations depicting True Positive Rate versus False Positive Rate in binary classification models. Their proximity to the upper-left corner often correlates to better model performance.

F Measure is an often-utilized performance metric in binary classification tasks. It represents the harmonic average between precision and recall - providing an assessment that considers both aspects. Equation (7) illustrates this calculation process of the F1 score:

$$\text{F Measure} = 2 \times \left(\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}\right) \quad (7)$$

Training time is the duration required to train a model on a specific dataset. Fast training times are desirable as they facilitate rapid model development and deployment.

Testing time pertains to the duration taken to assess a trained model on a new dataset. Efficient testing times are advantageous as they enable prompt inference of model predictions.

5. RESULTS AND DISCUSSIONS

In In this section, we conduct two experiments: the first experiment focuses on classifying network behavior as normal or attack, and the second experiment extends the classification to distinguish between normal operations and various types of attacks. For both experiments, we will evaluate different models with and without the application of feature engineering (PCA based feature engineering and mRMR based feature selection).

A. First Experiment for First Level (Binary) Classification

In the initial experiment, the details of the full dataset along with its training and testing splits using 70:30% is shown in Figure 5. The metrics used for comparison will include accuracy, error rate, precision, recall, specificity, and the F-measure across both the training and testing phases. This setup will allow us to assess the impact of PCA and mRMR on model effectiveness comprehensively.

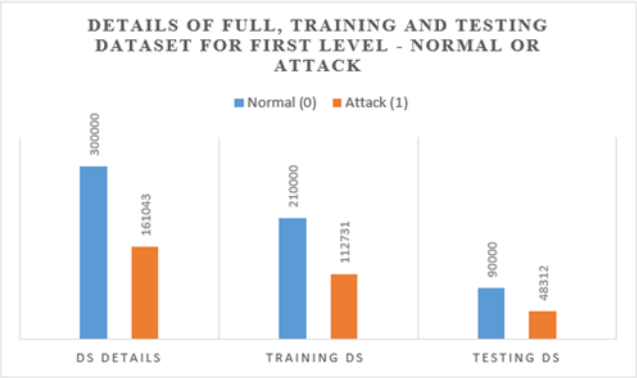


Fig 4: Details of Full, Training and testing datasets for first level of classification.

When applying mRMR to the dataset, we can specify a specific score threshold for dropping columns that score below this threshold. Figure 5 illustrates the feature scores for each column using mRMR, both in their original order and ranked from highest to lowest importance. These visuals provide valuable insights into which features are most influential in the dataset when mRMR is applied.

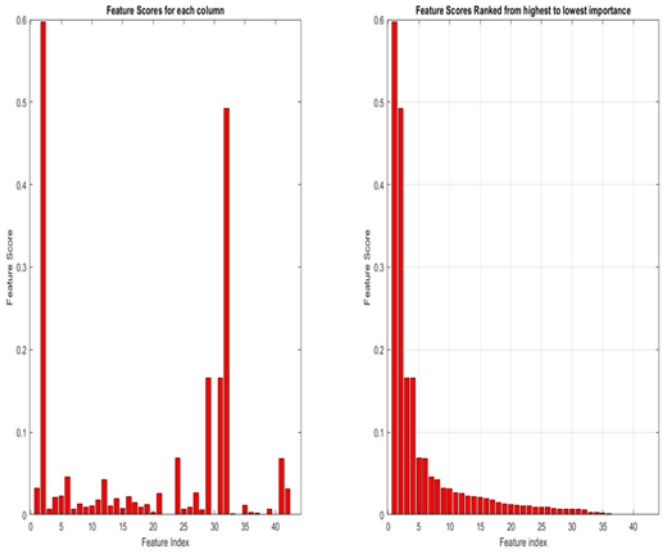


Fig 5: The feature scores for each column using MRMR, both in their original order and ranked from highest to lowest importance, for first level of classification.

Knowing that, setting a threshold value for feature scores in MRMR can significantly influence model performance and computational efficiency. Our choice of a threshold value of 0.01, which results in selecting the top 23 highest-scoring features and dropping the other 20, appears to be a strategic compromise between performance and efficiency.

Knowing that, choosing a higher threshold value will lead to fewer features being selected. This can simplify the model further and reduce training and testing times, but at the risk of dropping potentially important features, which might degrade the model's performance. It's crucial to ensure that the selected features still capture the essence of the data without sacrificing the ability to generalize well. Whereas, a lower threshold value means more features will be retained, which might maintain or even slightly enhance performance due to the richer feature set. However, this comes with increased computational costs, as more features require more processing power and time during both training and testing phases. This might not be desirable, especially in scenarios where rapid decision-making is critical. The columns that their score high or equal threshold ranking from highest to lowest score are: src_ip, http_method, ssl_subject, http_trans_depth, dns_rejected, weird_name, proto, missed_bytes, ts, weird_addl, ssl_resumed, dns_AA, dst_port, dst_ip_bytes, dst_ip, src_ip_bytes, conn_state, dns_query, duration, dns_qtype, http_request_body_len, dst_bytes, src_pkts.

Figure 6 demonstrates the cumulative variance explained by each principal component in the dataset. Based on this, we select the first 18 principal components, as they capture a significant portion of the variance (99.99%).

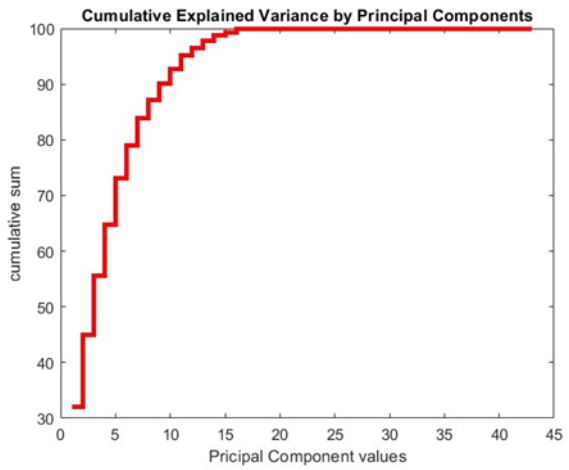


Fig 6: Cumulative Explained Variance by Principal Components.

Principal Component Analysis (PCA) is a statistical technique aimed at reducing the complexity in high-dimensional data while preserving essential trends and patterns. This reduction is achieved by transforming the original variables into a new set of variables, known as principal components, which are orthogonal and ordered by the amount of variance they capture from the data. Since PCA is an unsupervised method, it is applied to the feature set without considering the label. This characteristic makes PCA versatile across different types of classification tasks—whether for binary classification (first level) or multiclass classification (second level). The method does not differentiate between these scenarios because it focuses exclusively on the input features and their variance, not the outcome variable. Thus, the application of PCA is consistent regardless of the classification level, simplifying the feature space and ensuring uniformity in how the data is processed for model training.

The performance of various classification models without feature engineering, with mRMR feature selection, and with PCA feature reduction is shown in Table 3 (a)- (c) respectively.

From Table 3, we can observe the following:

- Without Feature Engineering: Models such as Naïve Bayes exhibit limited capability in this scenario, managing only 65.07% accuracy with a high error rate of 34.93%, underscoring its challenges with complex, unprocessed datasets. Conversely, KNN achieves a reasonable performance level at 86.83% accuracy, indicating a better handling of the raw data. The ensemble methods (Adaboost and BagTree) and other robust models like Decision Trees, SVM, and DL-LSTM demonstrate excellent performance, with accuracies nearing or achieving 99.99%. This suggests that more sophisticated or ensemble-based approaches are well-suited to manage the dataset's complexity effectively without prior feature engineering.
- With mRMR Feature Selection: The application of mRMR significantly improves the performance of Naïve Bayes, boosting its accuracy to 85.27%, which highlights the benefits of selecting relevant and minimally redundant features. KNN also sees an enhancement in its performance to 91.06%, benefiting from the streamlined feature set that enhances both precision and recall. Ensemble methods such as Adaboost and BagTree continue to perform at an optimal level with accuracies close to 99.99%, underscoring the complementarity of mRMR with these algorithms. - The high performance is kept intact with the Decision Trees, SVM and DL-LSTM that all score around 99 across the board. The 99% accuracy, therefore reinforcing the effectiveness of mRMR in improving model accuracy while maintaining the essential data integrity.
- With PCA Feature Reduction: Naïve Bayes illustrates a slight increase to 84. PCA applied gave a 99% accuracy which means it really reduced the feature dimensionality though not as much as mRMR did. KNN is a bit towards 90. 87% accuracy could be the result of PCA's reduction that may have removed the discriminative features which are important for its classification strategy.
- A slight performance decrease is also observed in ensemble methods, with Adaboost dropping to 99.68% and BagTree to 99.80%, suggesting that the loss of certain critical features can adversely affect their performance. Decision Trees, SVM, and DL-LSTM, although experiencing slight decreases in performance to accuracies like 99.91% and 99.95%, still indicate strong capabilities, albeit slightly hampered by the PCA process.

The analysis underscores that mRMR feature selection generally offers the most consistent performance improvements across models, likely due to its effectiveness in focusing on relevant and non-redundant features. PCA, while beneficial for simplifying the data structure, may not always be the best choice for all models, particularly those that depend on specific feature interactions.

The analysis of training and testing times across various classification models with and without using feature engineering techniques are shown in figure 7.

PCA consistently reduces both training and testing times more significantly across all models compared to mRMR. This reduction is because PCA reduces the dataset size to only 18 columns, which simplifies calculations by decreasing the complexity of the data. mRMR also reduces training and testing times in most cases, but not as dramatically as PCA because it selects a dataset size of 23 columns only. This suggests that while mRMR helps by

selecting the most relevant features, it does not reduce the computational load as much as PCA.

- Naïve Bayes and KNN: Both models show reduced testing times with PCA, indicating faster performance due to fewer and more compact features. However, Naïve Bayes experiences a slight increase in training time with mRMR.
- Ensemble Models (Adaboost and BagTree): These models benefit greatly from both techniques, with PCA yielding the most significant time reductions, facilitating faster model training and testing.
- Decision Tree and SVM: Experience substantial reductions in both training and testing times with PCA, suggesting these models are highly sensitive to the number of features, and benefit from the simplified feature space.
- DL-LSTM: Shows a pronounced reduction in training time with PCA, highlighting the advantage of reduced feature dimensions in complex neural network architectures.

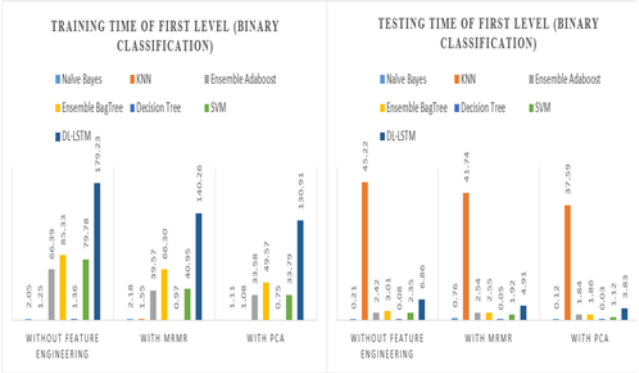


Fig 7: Training and Testing times across various classification models with and without feature engineering.

B. Second experiment for Second Level (MultiClass) Classification

In the second experiment, the focus shifts to classifying data into categories of 'normal' or various 'types of attack.' This experiment utilizes mRMR with a threshold value of 0.1 to strategically select the top 31 features for the classification task. For PCA, the 3. (a) Results of the different model without Feature Selection or reduction -First Level

Methods	Accuracy	Precision	Recall	Specificity	F Measure	AUC	Error Rate
Naïve Bayes	0.65070	NaN	0.50000	0.50000	NaN	0.752	0.340
KNN	0.86831	0.85261	0.86406	0.86406	0.85758	0.94760	0.131369
Ensemble Adaboost	0.99187	0.99186	0.99187	0.99186	0.99260	0.99991	0.999874
Ensemble BagTree	0.99979	0.99973	0.99979	0.99976	0.99978	0.99992	0.999992
Decision Tree	0.99996	0.99996	0.99996	0.99996	0.99996	1	0.000004
SVM	0.99999	0.99999	0.99999	0.99999	0.99999	1	0.000001
DL-LSTM	0.99999	0.99999	0.99999	0.99999	0.99999	1	0.000001

3. (b) Results of the different model with MRMR Feature Selection -First Level

Methods	Accuracy	Precision	Recall	Specificity	F Measure	AUC	Error Rate
Naïve Bayes	0.85272	0.83581	0.85109	0.85109	0.84187	0.93649	0.147228
KNN	0.91057	0.93909	0.87222	0.87222	0.89452	0.93659	0.089443
Ensemble Ada	0.99994	0.99992	0.99996	0.99996	0.99994	0.99999	0.000006

boost							
Ensemble BagTree	0.99998	0.99998	0.99997	0.99997	0.99998	1	0.00002
Decision Tree	0.99996	0.99996	0.99995	0.99995	0.99995	1	0.00006
SVM	0.99999	0.99999	0.99999	0.99999	0.99999	1	0.00001
DL-LSTM	0.99999	0.99999	0.99999	0.99999	0.99999	1	0.00001

3. (c) Results of the different model with PCA Feature Reduction -First Level

Methods	Accuracy	Precision	Recall	Specificity	F Measure	AUC	Error Rate
Naïve Bayes	0.84987	0.83281	0.84670	0.84670	0.83846	0.91978	0.150153
KNN	0.90873	0.93793	0.86961	0.86961	0.89217	0.93308	0.091227
Ensemble Adaboost	0.99679	0.99827	0.99679	0.99753	0.99776	0.9990	0.999999
Ensemble BagTree	0.99803	0.99776	0.99790	0.99790	0.99783	0.99997	0.001977
Decision Tree	0.99913	0.99900	0.99909	0.99909	0.99905	0.99944	0.000887
SVM	0.99935	0.99923	0.99933	0.99933	0.99928	0.99995	0.000065
DL-LSTM	0.99954	0.99946	0.99954	0.99954	0.99950	1.00000	0.000046

4. (a) Results of the different models without Feature Selection or reduction -Second Level

Methods	Accuracy	Precision	Recall	Specificity	F Measure	AUC	Error Rate
Naïve Bayes	0.80913	0.90329	0.99152	0.82837	0.90975	0.949	0.99319
KNN	0.89539	0.76736	0.91962	0.94532	0.75087	0.958	0.10461
Ensemble Adaboost	0.94199	0.43624	0.52882	0.95470	0.42764	0.9645	0.05801
Ensemble BagTree	0.98159	0.93175	0.72843	0.97443	0.78316	0.99221	0.01841
Decision Tree	1	1	1	1	1	1	0
SVM	0.99997	0.99974	0.99962	0.99997	0.99968	1	0.00003
DL-LSTM	1	1	1	1	1	1	0

4. (b) Results of the different models with MRMR Feature Selection -second Level

Methods	Accuracy	Precision	Recall	Specificity	F Measure	AUC	Error Rate
Naïve Bayes	0.93014	NaN	0.10000	0.90000	NaN	0.952	0.06986

KNN	0.95277	0.52897	0.60451	0.964 58	0.526 71	0.97	0.04723
Ensemble Adaboost	0.97174	0.70254	0.89480	0.985 16	0.751 93	0.983 89	0.02826
Ensemble BagTree	0.98346	0.99128	0.99950	0.987 29	0.999 55	0.999	0.99995
Decision Tree	1	1	1	1	1	1	0
SVM	0.99999	0.99992	0.99985	0.999 99	0.999 88	1	0.00001
DL-LSTM	1	1	1	1	1	1	0

4 (c) Results of the different models with PCA Feature Reduction -Second Level

Methods	Accuracy	Precision	Recall	Specificity	F Measure	AUC	Error Rate
Naïve Bayes	0.89550	0.76934	0.91970	0.945 38	0.752 50	0.942	0.10450
KNN	0.90389	0.94481	0.99715	0.917 16	0.994 98	0.968	0.99878
Ensemble Adaboost	0.94276	0.45157	0.52517	0.954 72	0.429 56	0.975	0.05724
Ensemble BagTree	0.98164	0.92579	0.72130	0.974 52	0.775 61	0.998	0.01836
Decision Tree	0.99973	0.99224	0.99385	0.999 80	0.993 04	0.997 42	0.00027
SVM	0.99272	0.96698	0.84340	0.991 72	0.876 01	0.993 02	0.00728
DL-LSTM	0.99990	0.99925	0.99640	0.999 92	0.997 80	0.999 96	0.00010

approach remains consistent with the first experiment, selecting the first 18 principal components to capture 99.99% of the cumulative variance. Detailed distributions of the full dataset, along with the training and testing splits set at a 70:30 ratio, are outlined in Figure 8.

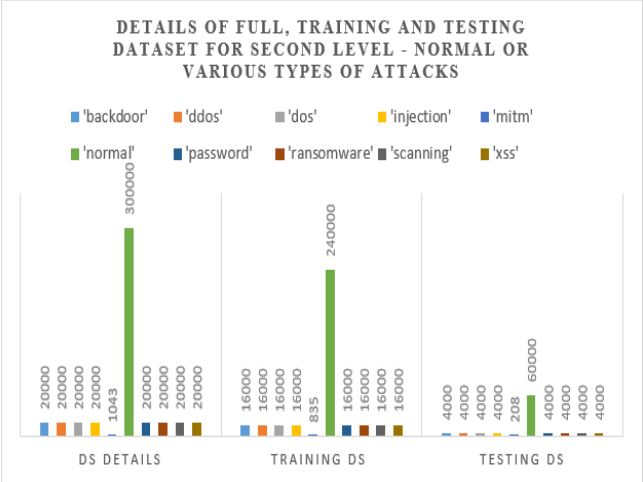


Fig 8: Details of Full, Training and testing datasets for second level (Multiclass) of classification.

Figure 9 illustrates the feature scores for each column using mRMR, both in their original order and ranked from highest to lowest importance, when applied to the second level (multiclass classification).

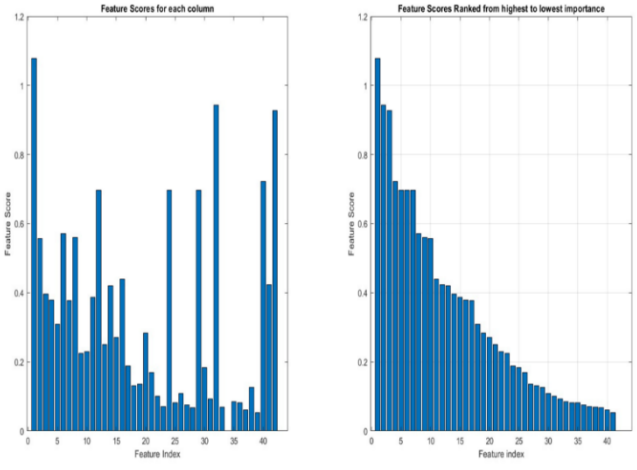


Fig 9: The feature scores for each column using MRMR, both in their original order and ranked from highest to lowest importance, for second level of classification.

The performance of various classification models without feature engineering, with mRMR feature selection, and with PCA feature reduction is shown in Table 4 (a)- (c) respectively.

- Without Feature Engineering: Naïve Bayes achieves a moderate accuracy of 80.91% but struggles with a high error rate of 99.32%, indicating difficulties in handling unprocessed datasets. KNN performs better with 89.54% accuracy, suggesting it can manage the raw data more effectively. Ensemble models like Adaboost and BagTree show good results, achieving accuracies of 94.20% and 98.16% respectively, though Adaboost's precision at 43.62% points to some classification challenges. Decision Trees, SVM, and DL-LSTM excel, displaying nearly perfect or perfect metrics, underscoring their robustness in complex scenarios.
- With mRMR Feature Selection: Naïve Bayes significantly improves to 93.01% accuracy but shows unstable metrics, such as NaN for precision and a low recall of 10.00%, likely indicating issues with outlier sensitivity or feature selection. KNN sees enhanced performance at 95.28% accuracy, benefiting from the more targeted feature set. Both Adaboost and BagTree perform robustly, with BagTree nearly achieving perfect precision and recall. Decision Trees, SVM, and DL-LSTM maintain their high performance, demonstrating mRMR's effectiveness in enhancing accuracy while preserving data integrity.
- With PCA Feature Reduction: Naïve Bayes shows an improved accuracy of 89.55%, benefiting from the reduced complexity of the dataset. KNN also improves, with accuracy at 90.39% and high precision of 94.48%. However, Ensemble Adaboost and BagTree, while still performing well, exhibit slight variations in their metrics. Decision Trees, SVM, and DL-LSTM perform exceptionally, albeit with some minor reductions in certain metrics, suggesting a slight loss of critical information due to the dimensionality reduction.

Finally, we conclude that the results highlight that PCA generally aids in simplifying model computations, but mRMR consistently delivers the most robust performance improvements by effectively selecting relevant and minimal redundant features.

The analysis of training and testing times across various classification models with and without using feature engineering techniques are shown in figure 10.

- Naïve Bayes experiences a significant reduction in training time with mRMR, dropping from 2.5 seconds to 0.59 seconds, though it increases to 1.95 seconds with PCA. This suggests that while mRMR significantly reduces the complexity, PCA slightly increases the processing time compared to mRMR. In testing, both methods reduce times, with PCA showing a more considerable reduction to 0.44 seconds from 0.6 seconds, indicating faster performance due to a more compact feature set.
- KNN sees a notable increase in training time with mRMR to 39.8 seconds from 1.4 seconds, likely due to the computational demand of processing relevant features. However, PCA dramatically reduces this time to 1.16 seconds, demonstrating the efficiency of dimensionality reduction. The testing time also decreases under PCA to 33.5 seconds, showing enhanced performance with fewer dimensions.
- Ensemble Adaboost and BagTree benefit from both techniques in training times, particularly with mRMR, which cuts down their times significantly from the original. - PCA also saves time but not as much as mRMR. The application of PCA and mRMR for testing the models is effectively done under both conditions, with PCA mostly giving some improvements or at least maintaining the times as that of mRMR which enables to have quicker model evaluations.
- Decision Tree and SVM training times are greatly reduced with PCA, which proves the efficiency of decreasing the feature space in these algorithms. PCA makes the testing times of Decision Trees almost negligible which means that these trees are extremely efficient at processing a reduced feature set. SVM has a small rise in testing time under PCA, which probably shows different computational dynamics during the model evaluation phase.
- DL-LSTM, a complicated neural network model, has a drastic cut in the training time with PCA to 157. 5 seconds from 198. 2 seconds. This shows the advantages of dimensionality reduction in training deep learning models, although there is a slight increase in testing time under PCA compared to mRMR.

On the whole, PCA mostly helps to cut down significantly on both training and testing times for all models since it is so good at reducing data complexity by lowering the dimensionality to only 18 columns. Besides, mRMR also decreases time but not to the same extent as PCA does and this shows that although it helps in concentrating on the most relevant features, it is still not enough for reducing computational load.

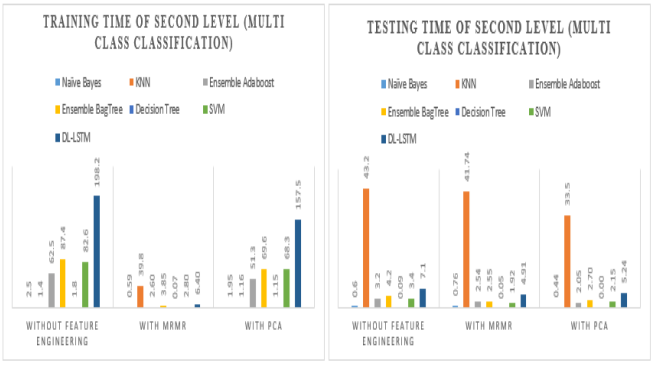


Fig 10: Training and Testing times across various classification models with and without feature engineering.

6. CONCLUSION

This study confirms the huge potential of combining Machine Learning (ML) and Deep Learning (DL) methods with advanced feature engineering techniques such as Minimum Redundancy Maximum Relevance (mRMR) and Principal Component Analysis (PCA) for improving Intrusion Detection Systems in Internet of Things environments. Thus, from the extensive experimentations it has been proved that ML and DL models optimized with feature engineering not only enhance in performance metrics like accuracy, precision, recall and AUC but also show significant reductions of both training as well as testing times. The effectiveness of PCA in reducing the dimensionality of the feature space, thus, simplifying the computational requirements without decreasing detection capabilities is very remarkable. Hence, the data processing becomes faster which is very important in real-time intrusion detection when it comes to quick response. In the same vein, mRMR has proved to be very effective in model enhancement by selecting only the most relevant features and thus providing a balanced way between model complexity and performance. The models like SVM, Decision Trees and DL-LSTM have proved their efficiency even in the context of security-critical issues. The DL-LSTM model, in particular, has proved to be very effective in dealing with sequence data which is crucial for IDS network traffic analysis.

The future research in the development of Intrusion Detection Systems (IDS) for IoT environments should focus on hybrid models that combine different ML and DL methods to deal with complex attack patterns efficiently. Federated learning can be a way to enhance privacy and scale in distributed environments, while the development of autonomous and adaptive systems will enable IDS to update and react to new threats in real-time. The coverage of tests should be extended to cover broader and more diverse datasets so as to confirm the strength and scalability of models. Besides the above aspects, the use of XAI practices would boost transparency and trustworthiness which are actually crucial for operational acceptance especially in critical infrastructure sectors. These advancements will make sure that IDS systems are not only efficient but also flexible and in line with the changing cybersecurity threat landscape.

REFERENCES

[1] Alotaibi, B. (2023). A survey on Industrial Internet of Things security: Requirements, attacks, AI based solutions, and edge computing opportunities. Sensors, 23(17), 7470. https://doi.org/10.3390/s23177470

[2] Mirani, A. A., Velasco-Hernandez, G., Awasthi, A., & Walsh, J. (2022). Key challenges and emerging technologies in Industrial IoT architectures: A review. Sensors, 22(15), 5836. https://doi.org/10.3390/s22041586

[3] X. Gming, S. Xiaorui, Z. Zhihua, and X. Bertino, "Advances in Artificial Intelligence and Security," in Proceedings of the 27th International Conference, ICAIS 2021, Dublin, Ireland, July 2021.

[4] Suzen, "Developing a multi-level intrusion detection system using hybrid-DBN," Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 2, pp. 1913–1923, 2021.

[5] Ayodeji, Y.-k. Liu, N. Chao, and L. Q. Yang, "A new perspective towards the development of robust data-driven intrusion detection for industrial control systems," Nuclear Engineering and Technology, vol. 52, no. 12, pp. 2687–2698, 2020.

[6] Eid, A., Nassif, A., Soudan, B., & Injadat, M. (2023). IIoT Network Intrusion Detection Using Machine Learning. 2023 6th International Conference on Intelligent Robotics and Control Engineering (IRCE), 196-201. https://doi.org/10.1109/IRCE59430.2023.10255088.

[7] S.M. Bridges and R.B. Vaughn, Fuzzy data mining and genetic algorithms applied to intrusion detection, USA, in in Proceedings of 12th Annual Canadian Information Technology Security Symposium, 2000. https://www.csee.umbc.edu/csee/research/cadip/readings/DMID/005slide.pdf

[8] Raghav, S. Chhikara, and N. Hasteer, Intrusion detection and prevention in cloud environment: a systematic review, Int. J. Comput. Appl., Vol. 68, 2013, pp. 7-11.

[9] R. Singh, H. Kumar, and R.K. Singla, An intrusion detection system using network traffic profiling and online sequential extreme learning machine, Expert Syst. Appl., Vol. 42, 2015, pp. 8609-8624.

[10] K. Peng, V.C.M. Leung, L. Zheng, S. Wang, C. Huang, and T. Lin, Intrusion detection system based on

decision tree over big data in fog environment, Wireless Commun. Mob. Comput., Vol. 2018, 2018, pp. 1-10.

[11] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection, ACM Comput. Surv. 41 (2009), 1–58.

[12] Y. Xue, W. Jia, X. Zhao, W. Pang, "An evolutionary computation based feature selection method for intrusion detection," Security and Communication Networks, 2018, (2018).

[13] Akashdeep, I. Manzoor, N. Kumar, A feature reduced intrusion detection system using ANN classifier, Expert Syst. Appl. 88 (2017), 249–257.

[14] Simone .A. Ludwig, Applying a neural network ensemble to intrusion detection, J. Artif. Intell. Soft Comput. Res. 9 (2019), 177–188.

[15] Chaopeng. Li, J. Wang, X. Ye, Using a recurrent neural network and restricted Boltzmann machines for malicious traffic detection, NeuroQuantology. 16 (2018), 823–831.

[16] Chuanlong. Yin, Y. Zhu, J. Fei, X. He, A deep learning approach for intrusion detection using recurrent neural networks, IEEE Access. 5 (2017), 21954–21961.

[17] Yihan Xiao, C. Xing, T. Zhang, Z. Zhao, An intrusion detection model based on feature reduction and convolutional neural networks, IEEE Access, , 7 (2019), pp. 42210–42219.

[18] Dimitrios. Papamartzivanos, F.G. Mármol, G. Kambourakis, Dendron: Genetic trees driven rule induction for network intrusion detection systems, Future Generation Computer Systems, 79 (2018), pp. 558–574.

[19] Kai. Peng, V.C.M. Leung, L. Zheng, S. Wang, C. Huang, T. Lin, Intrusion detection system based on decision tree over big data in fog environment, Wireless Commun. Mob. Comput. 2018 (2018), 1–10.

[20] G. Kim, S. Lee, S. Kim, A novel hybrid intrusion detection method integrating anomaly detection with misuse detection, Expert Syst. Appl. 41 (2014), 1690–1700.

[21] H.I. Ahmed, N.A. Elfeshawy, S.F. Elzoghdy, H.S. El-Sayed, O.S. Faragallah, A neural network-based learning algorithm for intrusion detection systems, Wireless Personal Communications, 97 (2017), pp. 3097–3112.

[22] W, Alhakami, A. ALharbi, S. Bourouis, R. Alroobaea, N. Bouguila, Network anomaly intrusion detection using a nonparametric Bayesian approach and feature selection, IEEE Access, 7 (2019), pp. 52181–52190.

[23] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, A. Abuzneid, Features dimensionality reduction approaches for machine learning based network intrusion detection, Electronics. 8 (2019), 322.

[24] L. Xiao, Y. Chen, C. K. Chang, Bayesian model averaging of Bayesian network classifiers for intrusion detection, in IEEE 38th International Computer Software and Applications Conference Workshops, Vasteras, Sweden, 2014.

[25] W. Feng, Q. Zhang, G. Hu, J.X. Huang, Mining network data for intrusion detection through combining SVMs with ant colony networks, Future Gener. Comput. Syst. 37 (2014), 127–140.

[26] E. Kabir, J. Hu, H. Wang, G. Zhuo, A novel statistical technique for intrusion detection systems, Future Gener. Comput. Syst. 79 (2018), 303–318.

[27] R. Vijayanand, D. Devaraj, B. Kannapiran, Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection, Comput. Secur. 77 (2018), 304–314.

[28] F. Kuang, S. Zhang, Z. Jin, W. Xu, A novel SVM by combining kernel principal component analysis and improved chaotic particle swarm optimization for intrusion detection, Soft Comput. 19 (2015), 1187–1199.

[29] S.M.H. Bamakan, H. Wang, T. Yingjie, Y. Shi, An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos particle swarm optimization, Neurocomputing. 199 (2016), 90–102.

[30] Alsaedi, A.; Moustafa, N.; Tari, Z.; Mahmood, A.; Anwar, A.N. TON-IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems. IEEE Access 2024, 1, 165130–165150.

[31] Abiri, N., Linse, B., Edén, P., & Ohlsson, M. (2019). Establishing strong imputation performance of a denoising autoencoder in a wide range of missing data problems. ArXiv, abs/2004.02584. https://doi.org/10.1016/j.neucom.2019.07.065.

[32] Lopez-Arevalo, I., Aldana-Bobadilla, E., Molina-Villegas, A., Galeana-Zapién, H., Muñiz-Sánchez, V., & Gausin-Valle, S. (2020). A Memory-Efficient Encoding Method for Processing Mixed-Type Data on Machine Learning. Entropy, 22. https://doi.org/10.3390/e22121391.

[33] Abdulhammed, R., Faezipour, M., Musafer, H., & Abuzneid, A. (2019). Efficient Network Intrusion

Detection Using PCA-Based Dimensionality Reduction of Features. 2019 International Symposium on Networks, Computers and Communications (ISNCC), 1-6. https://doi.org/10.1109/ISNCC.2019.8909140.

[34] Wang, C., Ye, X., He, X., Tian, Y., & Gong, L. (2019). Two-Level Feature Selection Method for Low Detection Rate Attacks in Intrusion Detection. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. https://doi.org/10.1007/978-3-030-21373-2_58.

[35] Tsiapoki, S., Bahrami, O., Häckell, M., Lynch, J., & Rolfes, R. (2020). Combination of damage feature decisions with adaptive boosting for improving the detection performance of a structural health monitoring framework: Validation on an operating wind turbine. Structural Health Monitoring, 20, 637 - 660. https://doi.org/10.1177/1475921720909379.

[36] Yu, H., Xu, C., Geng, G., & Jiang, Q. (2024). Multi-Time-Scale Shapelet-Based Feature Extraction for Non-Intrusive Load Monitoring. IEEE Transactions on Smart Grid, 15, 1116-1128. https://doi.org/10.1109/TSG.2023.3285117.

[37] Chen, L., Dong, X., Wang, B., Shang, L., & Liu, C. (2024). An Edge Computing-Oriented Islanding Detection Using Differential Entropy and Multi-Support Vector Machines. IEEE Transactions on Smart Grid, 15, 191-202. https://doi.org/10.1109/TSG.2023.3288361.

[38] Pujar, P., Kumar, A., & Kumar, V. (2024). Efficient plant leaf detection through machine learning approach based on corn leaf image classification. IAES International Journal of Artificial Intelligence (IJ-AI). https://doi.org/10.11591/ijai.v13.i1.pp1139-1148.

[39] Wang, Y., Yan, Z., Sang, L., Hong, L., Hu, Q., Shahidehpour, M., & Xu, Q. (2024). Acceleration Framework and Solution Algorithm for Distribution System Restoration Based on End-to-End Optimization Strategy. IEEE Transactions on Power Systems, 39, 429-441. https://doi.org/10.1109/TPWRS.2023.3262189.

[40] Nordin, S., Wah, Y., Haur, N., Hashim, A., Rambeli, N., & Jalil, N. (2024). Predicting automobile insurance fraud using classical and machine learning models. International Journal of Electrical and Computer Engineering (IJECE). https://doi.org/10.11591/ijece.v14i1.pp911-921.

[41] Vaiyapuri, T., & Binbusayyis, A. (2024). Deep self-taught learning framework for intrusion detection in cloud computing environment. IAES International Journal of Artificial Intelligence (IJ-AI). https://doi.org/10.11591/ijai.v13.i1.pp747-755.

[42] Alrahhal, M., & Supreethi K.P. (2020). Multimedia Image Retrieval System by Combining CNN With Handcraft Features in Three Different Similarity Measures. International Journal Of Computer Vision And Image Processing, 10(1), 1-23. DOI: 10.4018/ijcvip.2020010101.

[43] Alrahhal, M., & K.P, S. (2021). Full Direction Local Neighbors Pattern (FDLNP). International Journal Of Advanced Computer Science And Applications, 12(1). DOI: 10.14569/ijacsa.2021.0120116.

[44] Alrahhal, M., & K P, S. (2021). COVID-19 Diagnostic System Using Medical Image Classification and Retrieval: A Novel Method for Image Analysis. The Computer Journal. DOI: 10.1093/comjnl/bxab051.