An integrated intrusion detection framework based on subspace clustering and ensemble learning

Zhu, Jingyi; Liu, Xiufeng

Published in:
Computers and Electrical Engineering

Link to article, DOI:
10.1016/j.compeleceng.2024.109113

Publication date:
2024

Document Version
Publisher's PDF, also known as Version of record

Computers and Electrical Engineering 115 (2024) 109113

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

An integrated intrusion detection framework based on subspace clustering and ensemble learning

Jingyi Zhu [a], Xiufeng Liu [b,*]

[a] School of Digital Information Technology, Zhejiang Technical Institute of Economics, 310018 Hangzhou, China
[b] Department of Technology, Management and Economics, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark

ARTICLE INFO

Keywords:
Intrusion detection
IoT networks
Ensemble learning
Subspace clustering
Feature selection

ABSTRACT

In the rapidly evolving landscape of the Internet of Things (IoT), ensuring robust intrusion detection is paramount for device and data security. This paper proposes a novel method for intrusion detection in IoT networks that leverages a unique blend of subspace clustering and ensemble learning. Our framework integrates three innovative strategies: Clustering Results as Features (CRF), Two-Level Decision Making (TDM), and Iterative Feedback Loop (IFL). These strategies synergize to enhance detection performance and model robustness. We employ mutual information for feature selection and utilize four subspace clustering algorithms – CLIQUE, PROCLUS, SUBCLU, and LOF – to create additional feature sets. Three base learners – NB, LGBM, and XGB – are used in conjunction with a Logistic Regression (LR) meta-learner. To fine-tune our model, we apply Particle Swarm Optimization (PSO) for hyperparameter optimization. We evaluate our framework on the UNSW-NB15 dataset, which contains realistic and diverse IoT network traffic data. The results show that our framework outperforms the state-of-the-art methods in terms of accuracy (97.05%), precision (96.33%), recall (96.55%), F1-score (96.45%), and false positive rate (0.029). Our framework can effectively detect both known and unknown attacks in IoT networks and achieve high accuracy and low false positive rate. The paper contributes both practical implications for network security and theoretical advancements in intrusion detection research.

1. Introduction

The Internet of Things (IoT) is a paradigm that enables the interconnection and interaction of various devices, sensors, and applications over the Internet, creating a network of smart objects that can collect, process, and exchange data [1]. IoT networks have numerous applications in various domains, such as smart cities, smart homes, smart health, smart agriculture, and smart industry [2]. However, IoT networks also pose new challenges and opportunities for network security and privacy, as they are characterized by heterogeneity, scalability, mobility, and resource constraints [3]. One of the key challenges in IoT network security is intrusion detection, which is the process of monitoring network traffic and system activities for malicious or anomalous behaviors, and alerting the administrators or taking preventive actions when such behaviors are detected [4]. Intrusion detection systems (IDSs) are software applications or devices that perform this task, offering protection against a variety of attack vectors, such as denial-of-service (DoS), distributed denial-of-service (DDoS), brute force assaults, port scanning activities, backdoor exploits, and worm intrusions [5,6]. IDSs can also help in forensic analysis, network management, and policy enforcement [7].

* Corresponding author.
E-mail addresses: zjygrace191@sina.com (J. Zhu), xiuli@dtu.dk (X. Liu).

https://doi.org/10.1016/j.compeleceng.2024.109113
Received 21 November 2023; Received in revised form 27 January 2024; Accepted 7 February 2024
Available online 10 February 2024

However, intrusion detection in IoT networks presents a complex challenge, marked by the relentless evolution of cyber threats. Traditional IDSs struggle to keep pace in this dynamic landscape. Signature-based methods, for instance, are adept at detecting known threats but often fail to identify novel or zero-day attacks, resulting in a higher rate of false negatives [1,8]. This limitation necessitates frequent updates to their databases to counter emerging threats effectively. For example, in 2023, a novel attack called Mirai exploited the vulnerabilities of IoT devices and launched a massive DDoS attack that disrupted the services of several major websites, such as Twitter, Netflix, and Amazon [9]. Signature-based IDSs were unable to detect this attack, as it did not match any of the existing signatures in their databases [9]. Anomaly-based techniques, on the other hand, excel in detecting unknown attacks but are hampered by lower accuracy and increased false positives, especially when normal data variations are present [3,10]. These methods' effectiveness hinges critically on the meticulous selection of features and fine-tuning of parameters. Researchers have explored hybrid systems, attempting to merge the reliability of signature-based approaches with the adaptability of anomaly-based methods [4,7]. However, these hybrid solutions introduce complexities in integrating disparate outputs, resolving methodological inconsistencies, and maintaining an optimal balance between detection accuracy and false positive rates. For example, in [4], the authors proposed a hybrid system that used a rule-based classifier and a neural network classifier for intrusion detection in IoT networks. However, they faced challenges in combining the outputs of the two classifiers, as they had different formats and scales. They also had to adjust the weights of the classifiers to achieve a trade-off between detection accuracy and false positive rates.

Moreover, existing IDSs for IoT networks often use conventional machine learning techniques, such as decision trees, support vector machines, or neural networks, which have several drawbacks, such as high computational cost, low interpretability, and poor scalability [11]. These techniques also often require a large amount of labeled data, which is scarce and expensive to obtain in the IoT context, as IoT devices generate massive amounts of heterogeneous and unlabeled data [12]. Furthermore, these techniques often assume that the data is independent and identically distributed, which is not the case in the IoT context, as IoT data is often correlated and non-stationary [13]. In addition, existing IDSs for IoT networks often use a single classifier or a simple ensemble of classifiers, which may not be able to capture the complexity and diversity of the data and the attacks [14]. These classifiers also often use the same set of features or the entire feature space, which may not be optimal or relevant for different types of attacks or data subsets [15]. Furthermore, these classifiers often use a majority voting or a weighted voting scheme to combine the outputs, which may not be able to handle the uncertainty and ambiguity of the data and the attacks [16]. Therefore, the pursuit of robust, adaptive, and more integrated IDS approaches is imperative in strengthening network security against the continually evolving cyber threats, particularly in the IoT context.

In this paper, we propose a novel framework that uses subspace clustering and ensemble learning techniques for intrusion detection in IoT networks. Our framework employs three synergistic strategies including clustering results as features (CRF), two-level decision making (TDM), and iterative feedback loop (IFL). These strategies enable our method to discover meaningful clusters from different feature subspaces, combine multiple base learners and a meta-learner to improve the performance and robustness of the detection model, and enhance the feature space and accuracy of intrusion detection by using different ways to combine the results of subspace clustering and ensemble learning. We use mutual information [17] to select relevant features for each dataset, and use four subspace clustering algorithms, CLIQUE [18], PROCLUS [19], SUBCLU [20] and LOF [21], to generate cluster labels as additional features. We use three base learners, Naive Bayes (NB) [22], Light Gradient Boosting Machine (LGBM) [23], and Extreme Gradient Boosting (XGB) [24], and Logistic Regression (LR) as a meta-learner to train a final model on the predictions of the base learners. NB is a probabilistic classifier that uses Bayes' theorem to calculate the posterior probabilities of the classes [22]. LGBM and XGB are gradient boosting frameworks that use tree-based learning algorithms to optimize the loss function and improve the accuracy [23,24]. LR is a linear classifier that uses a logistic function to model the probability of the classes [25]. We use Particle Swarm Optimization (PSO) [26] to optimize the hyperparameters of the base learners and the meta-learner. The obtained framework aims to cope with a more comprehensive and adaptive intrusion detection system for IoT networks. We evaluate the proposed method on the UNSW-NB15 dataset, which contains realistic and diverse IoT network traffic data. The novelty of the proposed approach lies not merely in the integration of existing techniques but in the synergistic effect that emerges from their combination. While subspace clustering has been used to explore local patterns and ensemble learning has been applied for robust classification, the marriage of these techniques in our framework allows for a form of 'intelligent adaptability'. This is particularly manifest in our three unique strategies: CRF, TDM and IFL. CRF capitalizes on the rich feature subspace identified by clustering to boost classification accuracy. TDM, on the other hand, refines classification results through a dual-layered decision process that leverages both subspace clustering and ensemble learning. Finally, IFL enables dynamic model recalibration based on real-time results. Each strategy serves to amplify the advantages and mitigate the limitations of the other, offering a holistic and highly effective solution for intrusion detection in IoT networks.

The main contributions of this paper are:

- We propose a novel intrusion detection method for IoT networks that leverages subspace clustering and ensemble learning techniques to achieve high performance and robustness.
- We propose different strategies for the integration of subspace clustering and ensemble learning techniques that can enhance the feature space and improve the accuracy of the proposed method by using different ways to combine the results of subspace clustering and ensemble learning.
- We conduct extensive experiments on a public dataset to evaluate and validate our proposed method and compare it with other existing methods. The results demonstrate that the proposed method can effectively detect both known and unknown attacks in IoT networks and achieve high accuracy and low false positive rate.

The rest of this paper is organized as follows. Section 2 reviews the related work on intrusion detection systems and intrusion detection techniques. Section 3 presents the research questions and objectives. Section 4 describes the proposed method in detail. Section 5 presents the experimental setup and results. Section 6 discusses the limitations and implications of our work. Section 7 concludes the paper and suggests some future work directions.

2. Literature review

2.1. Intrusion detection systems for IoT networks

IDSs are crucial for protecting network security and privacy, especially in the context of IoT. However, traditional IDSs face challenges in detecting the evolving and diverse cyber threats in IoT networks, necessitating the development of more robust, adaptive, and integrated approaches. In this subsection, we review the existing literature on intrusion detection for IoT networks, focusing on the data collection and data analysis components, the detection techniques, and the strategies employed to enhance the detection performance.

The data collection component of an IDS involves the capture and preprocessing of network traffic, including packet headers and payloads [27,28]. The choice of data source and format depends on the characteristics and requirements of the IoT network, such as the network topology, the communication protocol, the device heterogeneity, and the resource constraints [29]. For instance, some IoT networks may use wireless protocols such as ZigBee, Bluetooth, or Wi-Fi, while others may use wired protocols such as Ethernet or USB [30]. Similarly, some IoT networks may have devices with limited memory, processing power, or battery life, while others may have more capable devices [29]. These factors affect the amount and quality of data that can be collected and processed by the IDS.

The data analysis component of an IDS involves the application of classification or clustering techniques to segregate normal from anomalous connections [29]. The choice of technique depends on the nature and complexity of the intrusion detection problem, such as the type and frequency of attacks, the availability and reliability of labeled data, and the trade-off between detection accuracy and false positive rates [7,31]. For instance, some intrusion detection problems may involve frequent and known attacks, such as DoS or port scanning, while others may involve rare and unknown attacks, such as zero-day exploits or worm intrusions [2,5]. Similarly, some intrusion detection problems may have sufficient and trustworthy labeled data, while others may have scarce or noisy labeled data [7,31]. These factors affect the performance and suitability of the classification or clustering technique used by the IDS.

As mentioned in the introduction, IDSs are typically categorized into misuse-based and anomaly-based systems, depending on the detection technique they employ. Misuse-based systems rely on predefined rules or signatures to detect known attacks, while anomaly-based systems utilize statistical or machine learning methods to learn the normal network behavior and discern deviations from it [29]. However, both techniques have limitations in dealing with the dynamic and diverse cyber threats in IoT networks, as discussed in the introduction. Therefore, researchers have been exploring hybrid systems that combine the reliability of misuse-based systems in recognizing known threats with the flexibility of anomaly-based systems in detecting emerging attacks [4,7]. However, these hybrid systems also face challenges in integrating outputs from different methods and balancing detection accuracy and false positive rates, as discussed in the introduction.

To address the myriad challenges in intrusion detection for IoT networks, researchers have developed a diverse array of strategies, incorporating techniques such as feature selection, clustering, classification, and the development of hybrid models. Feature selection improves detection performance and reduces computational complexity by selecting relevant network data features [32,33]. Clustering, which groups similar data points based on similarity or distance measures, aids in identifying patterns and outliers [34,35]. Classification processes assign labels to data points, distinguishing normal from anomalous connections [29,36]. Hybrid models combine the strengths of various techniques for enhanced effectiveness [35,36]. Complementing these approaches, recent research has underscored the efficiency of deep learning in predicting security breaches in IoT networks [37], introduced AI-based two-stage intrusion detection for software-defined networks [38], provided a comprehensive analysis of methodologies through surveys [39], and explored the potential of artificial neural networks in threat detection [40]. These advancements highlight the ongoing evolution in intrusion detection strategies, driven by the specific characteristics and requirements of IoT networks and the dynamic nature of network threats, underscoring the need for adaptable and innovative approaches.

2.2. Intrusion detection techniques for IoT networks

In this subsection, we review five common and representative techniques for IDSs, namely feature selection, dimensionality reduction, clustering, classification, and hybrid approaches, as shown in Table 1. The table shows their strengths, weaknesses, and some examples of related algorithms. We briefly explain each technique and some specific algorithms that are suitable for IoT networks. Unlike these methods, our proposed method uses subspace clustering algorithms such as CLIQUE, PROCLUS, and SUBCLU, and ensemble learning algorithms such as NB, LGBM, XGB, and LR for intrusion detection in IoT networks. Subspace clustering algorithms can discover local patterns and structures in different feature subspaces, which can capture the characteristics and behaviors of normal and anomalous connections. Ensemble learning algorithms can integrate multiple base learners with different strengths and weaknesses, which can reduce the variance and bias of the classification models. We also propose different strategies for the integration of subspace clustering and ensemble learning techniques that can enhance the feature space and improve the accuracy of the proposed method by using different ways to combine the results of subspace clustering and ensemble learning.

Table 1
Intrusion detection techniques for IoT networks.

Technique	Description	Strengths	Weaknesses	Refs.
Feature selection	Selects a subset of features that are relevant and informative for intrusion detection	Reduces dimensionality and complexity; Improves data quality and efficiency; Enhances generalization and interpretability	May introduce noise or redundancy; May miss some important information; Requires careful analysis and evaluation	Mutual information [17]; Chi-square [41]; ReliefF [42]
Dimensionality reduction	Transforms high-dimensional data into low-dimensional data while preserving the essential information	Captures the latent structure and semantics; Reduces computational cost and storage space; Enhances data quality and efficiency	May lose some important information; May distort the data structure; Requires proper selection and evaluation	Principal component analysis [43]; Autoencoder [44]; t-distributed stochastic neighbor embedding [45]
Clustering	Groups similar data points into clusters based on some similarity or distance measure	Discovers local patterns and structures; Enhances anomaly detection capability; Distinguishes between dense and sparse clusters	May suffer from high computational complexity; May be sensitive to parameters; Requires proper selection and evaluation	K-means [33]; DBSCAN [46]; CLIQUE [18]
Classification	Assigns labels to data points based on some predefined rules or learned models	Improves classification accuracy and robustness; Uses different algorithms or techniques; Distinguishes between normal and anomalous connections	May be biased towards normal connections; May miss rare or novel attacks; Requires proper selection and evaluation	Naive Bayes [22]; Support vector machine [47]; LightGBM [48]
Hybrid approaches	Combines two or more techniques to improve the performance or effectiveness of intrusion detection	Leverages the advantages of different techniques; Overcomes their limitations; Uses different strategies for integration	May face some challenges in integration; May have increased complexity or trade-off; Requires careful design and implementation	CRF [49]; TDM [50]; IFL [51]

3. Research questions and objectives

Intrusion detection is a crucial task that aims to identify and prevent malicious activities in IoT networks, which have found applications in various domains such as smart homes, smart cities, and industrial automation. However, intrusion detection is a challenging problem in IoT networks, as it involves dealing with data that is diverse, complex, and dynamic. The data can vary in terms of the types and protocols of IoT devices, the features and formats of network traffic, and the nature and patterns of attacks. Moreover, the data can change over time, as new devices and attacks emerge, which can affect the performance and accuracy of intrusion detection methods. To address these challenges, we propose a novel intrusion detection method that leverages subspace clustering and ensemble learning techniques. By integrating these two techniques, we aim to develop an effective and accurate intrusion detection method that can cope with the variety and complexity of data in IoT networks and detect both known and unknown attacks.

The research question that guides our work is: *How can we leverage subspace clustering and ensemble learning techniques to develop an effective and accurate intrusion detection method that can handle the variety and complexity of data in IoT networks and detect both known and unknown attacks?*

To answer this research question, we formulate intrusion detection as a mathematical problem as follows: Given a dataset of network traffic $X = \{x_1, x_2, \ldots, x_n\}$, where each instance $x_i \in \mathbb{R}^d$ has d features and belongs to one of two classes – either normal or anomalous (indicative of an attack) – the objective is to construct a predictive model $f : \mathbb{R}^d \rightarrow \{\text{normal}, \text{anomalous}\}$ that can accurately classify any new instance $x \in \mathbb{R}^d$. Moreover, the model should be able to identify unknown attacks not present in the training data. To achieve this, we use subspace clustering techniques to partition the feature space into multiple subspaces, and then train an ensemble of classifiers on each subspace. The final prediction is obtained by aggregating the outputs of the ensemble members.

4. Methodology

4.1. Overview

We propose a novel and effective framework for intrusion detection in IoT networks that leverages subspace clustering and ensemble learning. Fig. 1 illustrates the comprehensive architecture of our framework, which consists of four main components: Data Preprocessing, Subspace Clustering, Ensemble Learning, and Synergy Strategies. In Data Preprocessing, we clean, encode, and normalize the raw data to enhance its quality and reliability. In Subspace Clustering, we use subspace clustering algorithms to identify clusters in different feature subspaces, and generate cluster labels as additional features. In Ensemble Learning, we use three base learners to train individual models on different subsets of features, and use logistic regression as a meta-learner to train a final model on the predictions of the base learners. We also use particle swarm optimization to optimize the hyperparameters

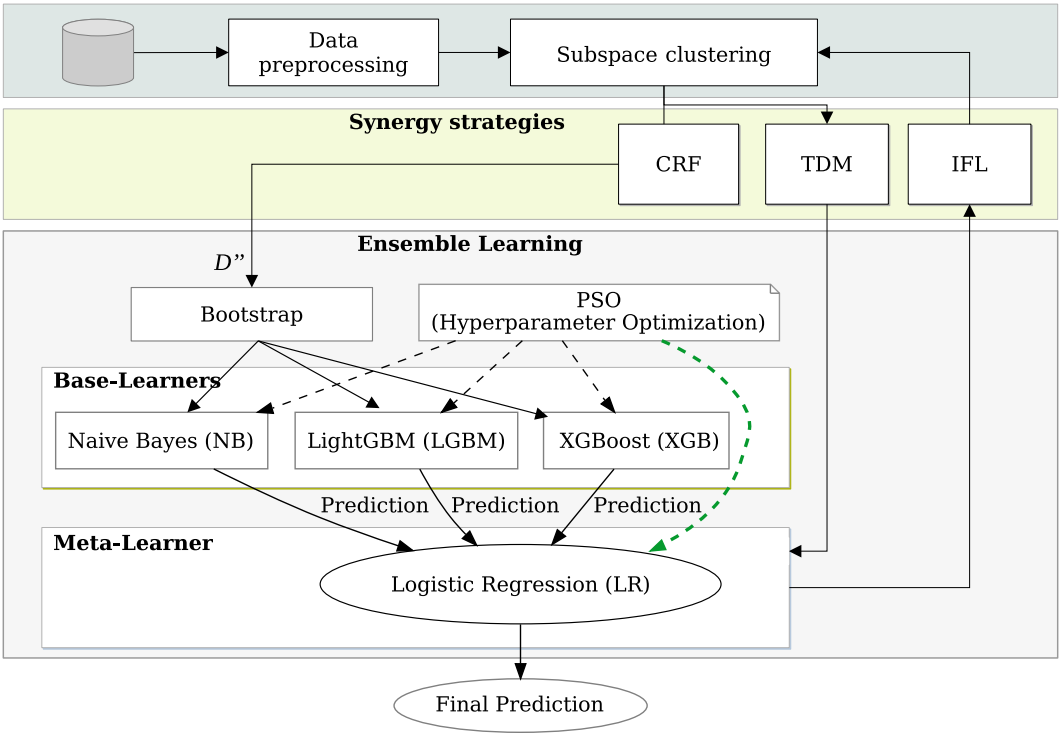

Fig. 1. Overview of the proposed intrusion detection framework.

of the base learners and the meta-learner. In Synergy Strategies, we use three synergistic strategies to combine the outputs of Subspace Clustering and Ensemble Learning: Clustering Results as Features (CRF), Two-Level Decision Making (TDM), and Iterative Feedback Loop (IFL). These strategies not only enhance the feature space but also improve the accuracy and robustness of the intrusion detection system. Our framework is designed to be iterative, adaptive, and scalable, thereby ensuring its applicability and effectiveness in diverse IoT network environments. In the following subsections, we will describe each component in greater detail.

4.2. Data preprocessing

Data preprocessing serves as a crucial step that converts raw data into a format amenable to subsequent analytical and modeling endeavors. This phase encompasses a multitude of operations, including noise elimination, imputation of missing values, encoding of categorical attributes, and normalization of numerical variables. The importance of data preprocessing lies in its capacity to elevate the data's quality and dependability, while also optimizing the efficacy and precision of both feature selection and model training processes. In this study, we employed an array of specific techniques and methods to preprocess data originating from IoT networks:

- **Cleaning**: We removed any duplicate or irrelevant instances from the datasets to reduce the noise and redundancy in the data. We also handled any missing values in the datasets by either deleting or imputing them using appropriate methods, such as mean or median imputation. Specifically, mean imputation substitutes missing values with the arithmetic mean of the observed data points, while median imputation replaces them with the statistical median of the observed values. Mathematically, mean imputation can be formalized as:

$$x_i = \frac{1}{n}\sum_{j=1}^{n} x_j \tag{1}$$

where x_i is the imputed value, n is the number of observed values, and x_j is the jth observed value.

$$x_i = \text{median}(x_1, x_2, \ldots, x_n) \tag{2}$$

where x_i is the imputed value, and $x_1, x_2, \ldots, x_n$ are the observed values sorted in ascending order.

- **Encoding**: We encoded any categorical variables in the datasets using one-hot encoding or label encoding techniques. One-hot encoding is a technique that creates binary dummy variables for each category of a variable, while label encoding is a technique that assigns numerical values to each category of a variable. We used one-hot encoding for nominal variables that

had no inherent order, such as device type or protocol type, and label encoding for ordinal variables that had some inherent order, such as attack type or severity level. The formulas for one-hot encoding and label encoding are:

$$d_{ik} = \begin{cases} 1 & \text{if } x_i = c_k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where d_{ik} is the kth dummy variable for the ith instance, x_i is the nominal variable value for the ith instance, and c_k is the kth category of the nominal variable.

$$l_i = f(x_i) \tag{4}$$

where l_i is the label value for the ith instance, x_i is the ordinal variable value for the ith instance, and f is a mapping function that preserves the order of the categories.

- **Normalizing**: We normalized any numerical variables in the datasets using min–max scaling or standardization techniques. Min–max scaling is a technique that rescales the values of a variable to a range between 0 and 1, while standardization is a technique that transforms the values of a variable to have zero mean and unit variance. We used min–max scaling for variables that had skewed distributions or outliers, such as packet size or duration, and standardization for variables that had normal distributions or no outliers, such as entropy or frequency. The formulas for min–max scaling and standardization are:

$$s_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{5}$$

where s_i is the scaled value for the ith instance, x_i is the original value for the ith instance, and $\min(x)$ and $\max(x)$ are the minimum and maximum values of the variable, respectively.

$$z_i = \frac{x_i - \bar{x}}{s_x} \tag{6}$$

where z_i is the standardized value for the ith instance, x_i is the original value for the ith instance, $\bar{x}$ is the mean value of the variable, and s_x is the standard deviation of the variable.

The preprocessed dataset D' is then used in the following stages of our methodology.

The data preprocessing phase is indispensable for augmenting both the quality and reliability of our dataset, while simultaneously optimizing the performance and accuracy metrics in the subsequent stages of feature selection and model training. Through the application of a diverse set of techniques such as cleaning, encoding, and normalization, we render the raw data into a format that is suitable for analysis or modeling.

4.3. Subspace clustering

Subspace clustering is an advanced technique designed to identify clusters within various subspaces of the feature space, as articulated in the work by Hou et al. [52]. A subspace refers to a selected subset of features that encapsulates the local structural intricacies and correlations inherent in the data. The utility of subspace clustering lies in its ability to reveal concealed patterns and anomalies that may elude detection in the complete feature space. Additionally, it serves to mitigate the data's dimensionality and computational complexity. In this study, we employed the following methods to execute subspace clustering on the preprocessed dataset denoted as D':

- **Subspace generation**: We generated all possible subspaces of features in D' using a bottom-up approach. We started with single features and iteratively combined them to form higher-dimensional subspaces. We used a lattice structure to represent the subspaces and their relationships. The lattice has $2^m - 1$ nodes, where m is the number of features in D'. Each node corresponds to a subspace and has a level that indicates its dimensionality. The root node represents the full feature space and has level m, while the leaf nodes represent single features and have level 1. The formulas for subspace generation are:

$$S = \{F_1, F_2, \ldots, F_m\} \tag{7}$$

where S is the set of all subspaces, and F_i is the ith feature in D'.

$$S_l = \{s \in S | \text{level}(s) = l\} \tag{8}$$

where S_l is the set of all subspaces with level l, and $\text{level}(s)$ is the function that returns the level of subspace s.

$$s_{ij} = s_i \cup s_j \tag{9}$$

where s_{ij} is the subspace obtained by combining subspaces s_i and s_j with the same level.

- **Subspace filtering**: We filtered out subspaces with low relevance or quality using a threshold-based approach. We calculated the variance ratio (VR) for each subspace using the following formula:

$$VR(S) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{k} d(x_i, c_j)^2}{\sum_{i=1}^{n} \sum_{j=1}^{m} d(x_i, y_j)^2} \tag{10}$$

where S is a subspace, n is the number of instances in D', k is the number of clusters in S, m is the number of features in S, x_i is the ith instance projected on S, c_j is the centroid of the jth cluster in S, and y_j is the mean value of the jth feature in S. The VR measures how well a subspace separates clusters from each other and from noise. A high VR indicates a high-quality subspace, while a low VR indicates a low-quality subspace. We used a threshold of 0.5 to filter out subspaces with low VR. We also filtered out subspaces with low dimensionality, such as single features or pairs of features, as they are unlikely to capture meaningful patterns or anomalies. We used a threshold of 2 to filter out subspaces with low dimensionality. The filtered subspaces form a subset of subspaces, denoted by F.

- **Subspace clustering**: We applied three subspace clustering algorithms, namely CLIQUE, PROCLUS, and SUBCLU, to each filtered subspace in F. CLIQUE is a density-based algorithm that partitions each subspace into dense units and forms clusters by connecting adjacent units. PROCLUS is a medoid-based algorithm that selects representative objects as medoids and assigns instances to their nearest medoids. SUBCLU is a hybrid algorithm that combines density-based and distance-based methods to find clusters in each subspace. We used the default parameters for each algorithm, as suggested by their original papers. The results of each algorithm are combined using a voting technique to obtain one final set of clusters for each subspace. The voting technique assigns each instance to the cluster that receives the most votes from different algorithms. If there is a tie, the instance is assigned to the cluster with the highest VR.
- **Cluster labeling**: We labeled each cluster as normal or anomalous based on its density and distance. We calculated the local outlier factor (LOF) and silhouette coefficient (SC) for each cluster using the following formulas:

$$LOF(C) = \frac{\sum_{i=1}^{n_C} \frac{lrd(x_i)}{\sum_{j=1}^{k} lrd(N_k(x_i))}}{n_C} \tag{11}$$

$$SC(C) = \frac{1}{n_C} \sum_{i=1}^{n_C} \frac{b(x_i) - a(x_i)}{\max\{a(x_i), b(x_i)\}} \tag{12}$$

where C is a cluster, n_C is the number of instances in C, x_i is the ith instance in C, $lrd(x_i)$ is the local reachability density of x_i, $N_k(x_i)$ is the set of k nearest neighbors of x_i, $a(x_i)$ is the average distance of x_i to other instances in the same cluster, and $b(x_i)$ is the minimum average distance of x_i to instances in other clusters. The LOF measures how isolated a cluster is from its neighboring clusters, while the SC measures how cohesive and well-separated a cluster is. Clusters with high LOF or low SC are labeled as anomalous, while clusters with low LOF or high SC are labeled as normal.

Algorithm 1 presents subspace clustering in details: it preprocesses the dataset, filters out low-quality subspaces, applies three clustering algorithms and combines them, calculates metrics to label clusters as anomalous or normal, and returns the labeled clusters.

Algorithm 1: Subspace Clustering

Data: Preprocessed dataset D'
Result: Labeled clusters
Let X be the feature matrix and y be the label vector of D';
Let $S = \{S_1, S_2, ..., S_k\}$ be the set of all possible subspaces of features in X, where $k = 2^d - 1$ and d is the number of features;
Let $F = \emptyset$ be the set of filtered subspaces with high variance ratio and level;
Let $C = \emptyset$ be the set of clusters obtained by subspace clustering;
Let $L = \emptyset$ be the set of cluster labels for each instance;
for *each subspace* $S_i \in S$ **do**
    Calculate variance ratio of S_i using equation (2);
    if $VR(S_i) \geq 0.5$ *and* $level(S_i) \geq 2$ **then**
        Add S_i to F;
for *each filtered subspace* $S_j \in F$ **do**
    Apply CLIQUE, PROCLUS, and SUBCLU to X_{S_j}, where X_{S_j} is the projection of X onto S_j;
    Combine results using voting technique to obtain a set of clusters C_j for S_j;
    Add C_j to C;
    for *each cluster* $C_{jk} \in C_j$ **do**
        Calculate LOF and SC of C_{jk} using equations (3) and (4);
        Label C_{jk} as anomalous or normal based on LOF and SC values;
        Add cluster label to L;
Return labeled clusters as (C, L);

4.4. Ensemble learning

In this study, we introduce an innovative ensemble learning architecture tailored for intrusion detection, comprising two hierarchical layers: *base learners* and *meta-learner*. Ensemble learning serves as a powerful mechanism for amalgamating multiple

base learners, thereby enhancing the robustness and diversity of the learning outcomes. The base learners function as discrete models, each trained on distinct data subsets or feature sets. In contrast, the meta-learner operates at a more abstract level, learning from the collective predictions generated by the base learners. The architecture of the proposed ensemble learning model is visually represented in Fig. 1, and the methods and techniques employed are described as follows:

- **Base learners**: We use three base learners: Naive Bayes (NB), LightGBM (LGBM), and XGBoost (XGB), which are well-known and widely used classification algorithms. NB is a probabilistic algorithm that applies Bayes' theorem to calculate the posterior probabilities of each class given the features. LGBM and XGB are gradient boosting algorithms that build an ensemble of decision trees by iteratively fitting new trees to the residuals of the previous trees. We chose these base learners because they have different strengths and weaknesses, and they can complement each other in the ensemble.
- **Meta-learner**: We employ logistic regression (LR) as the meta-learner for our ensemble learning model. LR is a linear algorithm that utilizes a logistic function to model the class probabilities of binary outcomes. We chose LR as the meta-learner because it is simple and effective in combining the predictions of the base learners through a weighted schema, while also offering interpretable coefficients associated with each base learner. However, LR is a binary classifier by default, and it cannot handle multi-class problems directly. Therefore, we use the one-vs-rest (OvR) scheme to extend LR to multi-class problems. The OvR scheme splits the multi-class problem into multiple binary problems, one for each class, and trains a LR model for each binary problem. The final prediction for an instance is the class that has the highest probability among all the binary models. To determine the class probabilities, we use a threshold of 0.5 for all classes, which is the default value in the scikit-learn library that we use for implementing our model [53]. If an instance is assigned to more than one class, or to none of the classes, by the OvR scheme, we use a tie-breaking rule that assigns the instance to the class with the highest frequency in the training data. The OvR scheme is a common and effective way to adapt binary classifiers to multi-class problems, and it is supported by the scikit-learn library that we use for implementing our model [53].
- **Bootstrap sampling**: We use a bootstrap sampling [54] technique to train each base learner on a different subset of data or features. Bootstrap sampling is a technique that randomly selects instances or features from the original dataset with replacement, creating a new dataset with the same size as the original one. Bootstrap sampling can help to increase the diversity and reduce the correlation of the base learners, as well as to prevent overfitting and improve generalization.
- **Hyperparameter optimization**: We employ PSO to optimize the hyperparameters of each base learner and meta-learner. PSO is a population-based optimization algorithm that mimics the social behavior of birds or fish. PSO consists of a set of particles that move in the search space, guided by their own best position and the global best position. PSO can help to find the optimal or near-optimal values for the hyperparameters that affect the performance of the learners, such as learning rate, number of trees, number of leaves, penalty, and regularization.

Ensemble learning combines multiple learners to improve classification. Algorithm 2 shows how to do this. It optimizes learner hyperparameters using PSO, trains learners on bootstrap samples, obtains predictions of learners for each instance, trains the meta-learner on the predictions as features, obtains final predictions of the meta-learner for each instance, and returns the final predictions.

Algorithm 2: Ensemble Learning Model

Data: Preprocessed and clustered dataset, D''
Result: Final predictions for each instance
Let $T = \{T_1, T_2, ..., T_p\}$ be the set of base learners: NB, LGBM, and XGB;
Let M be the meta-learner: LR;
Let $P = \{p_1, p_2, ..., p_n\}$ be the set of final predictions for each instance;
Optimize hyperparameters of each base learner using PSO;
for *each base learner* $T_i \in T$ **do**
    Train T_i on a bootstrap sample of D'';
    Obtain predictions of T_i for each instance in D'';
Train M on the predictions of all base learners as features;
Obtain predictions of M for each instance in D'' as P;
Return P as the final predictions for each instance;

4.5. An integrated intrusion detection framework

We now integrate subspace clustering and ensemble learning in a synergistic way to obtain an integrated intrusion detection framework. The framework consists of three strategies: clustering results as features (CRF), two-level decision making (TDM), and iterative feedback loop (IFL). These strategies aim to enhance the feature space, improve the accuracy, and ensure the consistency of both techniques by using different ways to integrate the results of subspace clustering and ensemble learning. We describe the three synergy strategies in the following and the pseudocode of the proposed framework in Algorithm 3.

- **Clustering results as features (CRF)**: This strategy uses the cluster labels obtained from subspace clustering as additional features for the ensemble learning model. Let $C = \{c_1, c_2, \ldots, c_m\}$ be the set of clusters obtained from subspace clustering, where m is the number of clusters. For each cluster c_i, we assign a cluster label x_i to each instance in the dataset based on its membership to c_i. We then encode these cluster labels using one-hot encoding or label encoding techniques, depending on the number of clusters in each subspace. The one-hot encoding technique creates a binary vector of length m for each instance, where only one element is 1 and the rest are 0, indicating which cluster the instance belongs to. The label encoding technique assigns a numerical value to each cluster label in a subspace, creating a single feature for each instance. The formulas for one-hot encoding and label encoding are:

$$d_{ik} = \begin{cases} 1 & \text{if } x_i = c_k \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

where d_{ik} is the kth element of the one-hot encoded vector for the ith instance, x_i is the cluster label value for the ith instance, and c_k is the kth cluster label in a subspace.

$$l_i = f(x_i) \tag{14}$$

where l_i is the label encoded value for the ith instance, x_i is the cluster label value for the ith instance, and f is a mapping function that assigns numerical values to each cluster label in a subspace.
We then concatenate these encoded cluster labels with the original features, creating an enhanced feature space for the ensemble learning model. Let X be the original feature matrix of size $n \times d$, where n is the number of instances and d is the number of features. Let D be the one-hot encoded matrix of size $n \times m$, where m is the number of clusters. Let L be the label encoded vector of size $n \times 1$. The enhanced feature matrix X' is obtained by concatenating X, D, and L horizontally, resulting in a matrix of size $n \times (d + m + 1)$. The formula for concatenation is:

$$X' = [X|D|L] \tag{15}$$

where | denotes horizontal concatenation.
This strategy enhances the feature space and provides more information to the classifiers, improving their performance and accuracy.

- **Two-level decision making (TDM)**: This strategy employs a two-level decision-making process that combines the predictions of subspace clustering and ensemble learning. In the first level, subspace clustering assigns a cluster label to each instance, which indicates its preliminary class (normal or anomalous). In the second level, ensemble learning assigns a probability to each instance, which indicates its refined class (normal or anomalous). The final prediction for each instance is obtained by applying a weighted voting scheme to the cluster label and the probability. The weighted voting scheme assigns different weights to subspace clustering and ensemble learning based on their confidence and accuracy. The confidence of subspace clustering is measured by the VR and the LOF of each cluster, which reflect the compactness and the isolation of the clusters. The accuracy of ensemble learning is measured by the F1-score of each classifier, which reflects the balance between precision and recall. The formula for weighted voting is:

$$y_i = \text{argmax}_c \sum_{t=1}^{T} w_t p_{ti}(c) \tag{16}$$

where y_i is the final prediction for the ith instance, c is a class label (normal or anomalous), T is the number of techniques (subspace clustering and ensemble learning), w_t is the weight assigned to the tth technique, and $p_{ti}(c)$ is the probability of the ith instance belonging to class c according to the tth technique. The weight of subspace clustering is calculated as:

$$w_1 = \frac{VR \times LOF}{VR \times LOF + F1} \tag{17}$$

and the weight of ensemble learning is calculated as:

$$w_2 = \frac{F1}{VR \times LOF + F1} \tag{18}$$

where VR is the validity ratio of the cluster that the instance belongs to, LOF is the local outlier factor of the cluster that the instance belongs to, and $F1$ is the average F1-score of the classifiers in the ensemble.
The rationale behind the TDM strategy is to leverage the strengths of both techniques and compensate for their weaknesses. Subspace clustering can capture the intrinsic structure and the hidden patterns of the data, and identify the outliers that may indicate anomalies. However, subspace clustering may suffer from the high dimensionality, imbalance, and heterogeneity of the data, and may produce clusters that are not consistent with the true labels. Ensemble learning can exploit the labeled data and the diversity of the classifiers to reduce the variance and bias of the prediction errors. However, ensemble learning may be affected by the noise and the uncertainty of the data, and may fail to detect the novel and complex anomalies. By combining the predictions of both techniques, we can improve the accuracy and reduce the false positives of our intrusion detection system. In particular, when subspace clustering and ensemble learning disagree on the prediction of an instance, in this study we assign a higher weight to ensemble learning than subspace clustering, because ensemble learning is generally more accurate and reliable than subspace clustering in this domain [55]. This way, we can avoid misclassifying the instances that are anomalous but belong to normal clusters, or vice versa.

Algorithm 3: Integration of Subspace Clustering and Ensemble Learning

Data: Preprocessed dataset D'
Result: Final predictions and labeled clusters
Let X be the feature matrix and y be the label vector of D';
Let $S = \{S_1, S_2, ..., S_k\}$ be the set of subspaces generated by mutual information;
Let $C = \{C_1, C_2, ..., C_m\}$ be the set of clusters obtained by subspace clustering;
Let $L = \{l_1, l_2, ..., l_n\}$ be the set of cluster labels for each instance;
Let D be the one-hot encoded matrix of cluster labels;
Let X' be the enhanced feature matrix obtained by concatenating X, D, and L;
Let $T = \{T_1, T_2, ..., T_p\}$ be the set of base learners for ensemble learning;
Let M be the meta-learner for ensemble learning;
Let $P = \{p_1, p_2, ..., p_n\}$ be the set of final predictions for each instance;
if *using CRF* **then**
    Perform Subspace Clustering on X;
    Encode cluster labels using one-hot encoding or label encoding;
    Concatenate cluster labels with original features: $X' = [X|D|L]$;
else
    Use original features: $X' = X$;
end
if *using TDM* **then**
    Use Subspace Clustering for preliminary classification: $L = C(X)$;
    Use Ensemble Learning for refined classification: $P = M(T(X'))$;
else
    Use Ensemble Learning for classification: $P = M(T(X'))$;
end
if *using IFL* **then**
    Initialize loop variables: $\Delta = \infty$, $\epsilon = 0.01$;
    while $\Delta > \epsilon$ **do**
        Perform Subspace Clustering on X: $C = SC(X)$;
        Perform Ensemble Learning on X': $P = M(T(X'))$;
        Update Subspace Clustering with Ensemble Learning results: $C' = C(P)$;
        Calculate change in VR, LOF, and SC values:
        $\Delta = \max_{S,C} |VR(S,C) - VR'(S,C)| + |LOF(S,C) - LOF'(S,C)| + |SC(S,C) - SC'(S,C)|$;
        Update Subspace Clustering with new clusters: $C = C'$;
    end
end
Return final predictions and labeled clusters: (P, C);

- **Iterative feedback loop (IFL)**: This strategy establishes an iterative feedback loop between subspace clustering and ensemble learning. The clustering results inform the learning model by providing cluster labels as additional features for each instance. The learning model's outputs are fed back into the clustering process by updating the cluster centroids with the predicted class probabilities of each instance. This iterative procedure continues until convergence, ensuring a harmonized integration of both techniques. The convergence criterion is based on the change of VR, LOF, and SC values for each subspace and cluster. If there is no significant change in these values after an iteration, we stop the loop and obtain the final predictions and labeled clusters. The formula for the convergence criterion is:

$$\Delta = \max_{S,C} |VR(S,C) - VR'(S,C)| + |LOF(S,C) - LOF'(S,C)| + |SC(S,C) - SC'(S,C)| \tag{19}$$

where Δ is the maximum change in VR, LOF, and SC values for any subspace S and cluster C after an iteration, $VR(S,C)$, $LOF(S,C)$, and $SC(S,C)$ are the VR, LOF, and SC values before the iteration, and $VR'(S,C)$, $LOF'(S,C)$, and $SC'(S,C)$ are the VR, LOF, and SC values after the iteration.

4.6. Complexity analysis

In this section, we analyze the computational complexity of our proposed framework for intrusion detection in IoT networks. We use the big-O notation to express the upper bound of the running time or the space consumption of our framework, as a function of the input size. We assume that the input size is determined by the number of features, instances, clusters, and classes in the dataset, denoted by m, n, k, and l, respectively. We also assume that the number of iterations, particles, and the time complexity of the objective function in the particle swarm optimization algorithm are denoted by i, p, and t, respectively.

Our framework consists of four main components: Data Preprocessing, Subspace Clustering, Ensemble Learning, and Synergy Strategies. The complexity of each component is as follows:

- Data Preprocessing: This component involves cleaning, encoding, normalizing, and selecting the features in the dataset. The complexity of this component is $O(n^2m)$, as it is dominated by the feature selection technique that computes the score or the correlation for each pair of features and instances in the dataset, and selects the top k features.
- Subspace Clustering: This component involves generating, filtering, clustering, and labeling the subspaces of features in the dataset. The complexity of this component is $O(inm + ind + imdk + ind^2 + d^3 + ipitd)$, as it is dominated by the iterative feedback loop strategy that repeats the subspace clustering and the ensemble learning steps until convergence.
- Ensemble Learning: This component involves training and optimizing an ensemble of base learners and a meta-learner for classification. The complexity of this component is $O(nmdk + nd^2 + d^3 + pitd)$, as it is dominated by the base learners and the meta-learner that build an ensemble of decision trees by iteratively fitting new trees to the residuals of the previous trees, and use logistic regression to combine the predictions of the base learners.
- Synergy Strategies: This component involves integrating the results of subspace clustering and ensemble learning in a synergistic way. The complexity of this component depends on the strategy that is used. If the clustering results as features strategy is used, the complexity is $O(nm + nd + nm + nd + d^2)$, as it assigns, encodes, and concatenates the cluster labels with the original features. If the two-level decision making strategy is used, the complexity is $O(nm + nk)$, as it compares the predictions and the cluster labels for each instance. If the iterative feedback loop strategy is used, the complexity is $O(inm + ind + imdk + ind^2 + d^3 + ipitd)$, as it repeats the subspace clustering and the ensemble learning steps until convergence.

Therefore, the overall complexity of our framework is $O(inm + ind + imdk + ind^2 + d^3 + ipitd)$, as it is dominated by the subspace clustering and the ensemble learning components. This means that our framework has a polynomial running time with respect to the input size, and that it can handle moderate and large data efficiently and effectively.

5. Experiments

In this section, we describe the experimental setup used to evaluate our intrusion detection framework. We discuss the dataset used, the experimental configurations, the evaluation metrics, and the results obtained.

5.1. Dataset and preprocessing

We chose the UNSW-NB15 dataset [56] for our experiments, because it is a realistic and comprehensive dataset for intrusion detection in IoT networks, and it has several advantages over other datasets. First, it is one of the most recent and updated datasets for IoT intrusion detection, as it was published in 2015, and it reflects the current trends and challenges of IoT security. Second, it is one of the most widely used and cited datasets for IoT intrusion detection, as it has been used by many researchers and practitioners in the field, and it has been validated and evaluated by various methods and techniques [10,57–61]. Third, it is one of the most realistic and representative datasets for IoT intrusion detection, as it was generated by simulating a real network environment with normal and attack scenarios, and it covers a wide range of IoT devices, protocols, and attacks. Fourth, it is one of the most comprehensive and informative datasets for IoT intrusion detection, as it contains a large number of features and instances, and it provides detailed annotations and labels for each instance. The dataset poses several challenges and opportunities for developing effective and efficient intrusion detection systems for IoT networks, such as dealing with high-dimensional, heterogeneous, and imbalanced data, and detecting both known and unknown attacks. Therefore, we believe that the UNSW-NB15 dataset is a suitable and relevant choice for our research, and that it can provide reliable and meaningful results for our proposed framework.

The UNSW-NB15 dataset contains 49 features and 2,540,044 instances, distributed across four CSV files. The features include various attributes such as source and destination IPs, ports, protocols, packet lengths, flags, payloads, and timestamps, as well as derived metrics like service type, connection state, and host or service-specific statistics like the number of connections and bytes transferred. The instances are annotated with attack types (e.g., Fuzzers, Analysis, Backdoor) and categories (e.g., Exploits, Reconnaissance), providing a diverse range of attacks. As shown in Fig. 2, the dataset has an unbalanced distribution of attack categories, with some categories being much more frequent than others. This implies that some types of attacks are more common and easier to detect, while others are more rare and difficult to detect.

Fig. 3 shows the label distribution between attack and normal instances in the dataset. It is evident that attacks are more frequent than normal instances, accounting for about 68% of the total instances. This indicates that the dataset reflects a high-risk and high-threat network environment, where IoT devices are constantly under attack. This also suggests that the dataset is suitable for testing the robustness and adaptability of intrusion detection systems, as they need to cope with a large and diverse set of attacks.

To prepare the data for our proposed framework, we perform several preprocessing steps, such as data splitting, data cleaning, data transformation, data normalization, and data reduction. Data splitting involves splitting the dataset into two subsets: one with 175,341 records for training and another with 82,332 records for testing. We use stratified sampling to ensure that the proportion of each class is preserved in both subsets. We also ensure that the subsets are independent and representative of the original dataset, by excluding any instances that appear in both subsets. Data cleaning involves removing any missing, noisy, or inconsistent data, such as null values, outliers, or duplicates, that can affect the quality and accuracy of our framework. Data transformation involves transforming any categorical features into numerical features, using one-hot encoding or label encoding, depending on the number of categories and the ordinality of the feature. This can help to make the data compatible and consistent with our framework. Data

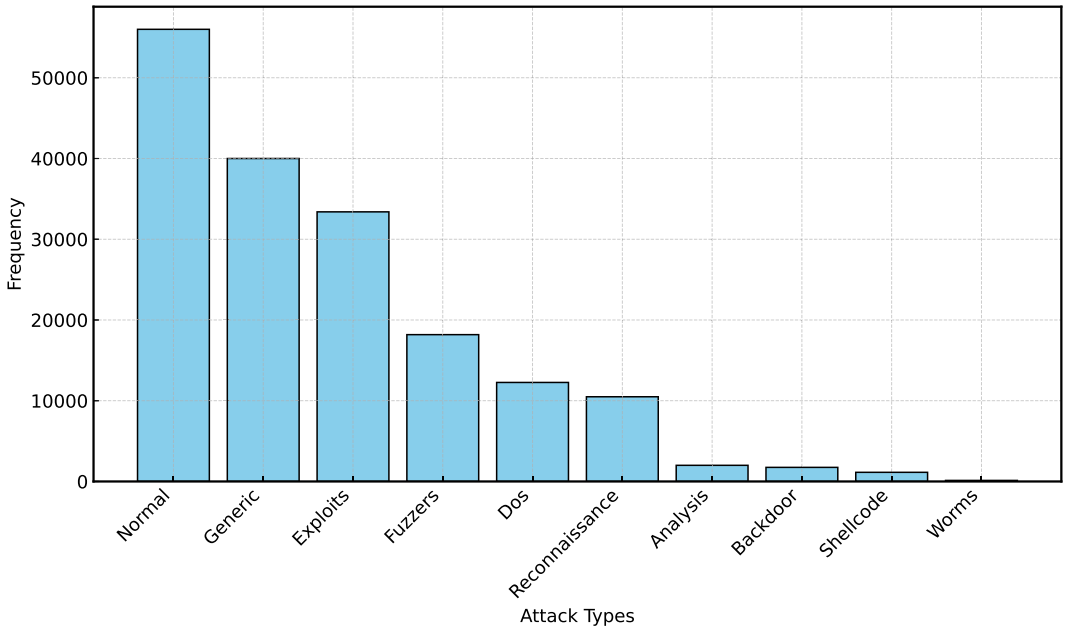

Fig. 2. Distribution of attack categories in UNSW-NB15 dataset.

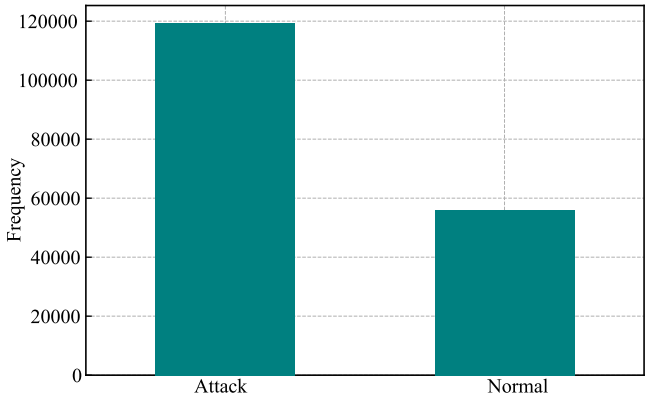

Fig. 3. Label distribution between attack and normal instances.

normalization involves normalizing any numerical features to have a common scale, using min–max scaling or standardization, depending on the distribution and range of the feature. This can help to reduce the skewness and variance of the data, and to improve the performance and stability of our framework. Data reduction involves selecting the most relevant and informative features for our classification task, using feature importance scores or correlation analysis, to reduce the dimensionality and complexity of the data. This can help to eliminate the noise and redundancy of the data, and to enhance the efficiency and effectiveness of our framework.

We also address the issue of class imbalance in our dataset, which poses a challenge for the classification model. Class imbalance can lead to the model favoring the majority class and neglecting or misclassifying the minority class, which is often the most critical or interesting one. To address this problem, we apply two techniques: weighted sampling and weighted metrics. Weighted sampling involves adjusting the sampling probability of each class based on its frequency, so that the classes with lower frequency have higher chances of being selected for training. This can help to balance the distribution of the classes in the training data, and to enhance the diversity and representation of the minority class. We implement weighted sampling using the scikit-learn library, by setting the class_weight parameter to 'balanced' in the resample function [53]. Weighted metrics involves assigning different weights to each class based on its frequency, so that the classes with lower frequency have higher impact on the evaluation of the model. This can help to measure the performance of the model for each class, and to avoid the metric trap of relying on the overall accuracy, which can be misleading in imbalanced datasets. We implement weighted metrics using the scikit-learn library, by setting the average parameter to 'weighted' in the precision_score, recall_score, and f1_score functions [53].

5.2. Experimental configurations

We conducted the experiments in this work on a HP laptop with Windows 10 Operating System. The laptop has an Intel(R) Core(TM) i5-8265U CPU @ 1.60 GHz processor and 8 GB of memory. We used the Scikit-Learn ML Python framework to build, train, evaluate and test the ML models. Scikit-Learn is a versatile open source platform that supports various tasks such as Classification, Regression and Clustering. It is based on matplotlib, NumPy and Scipy Python libraries.

5.3. Evaluation metrics

The evaluation metrics that we used to measure the performance and effectiveness of the proposed method and other methods are accuracy, precision, recall, f1-score, false positive rate (FPR), true negative rate (TNR), false negative rate (FNR), false discovery rate (FDR), false omission rate (FOR), Matthews correlation coefficient (MCC), training time, and prediction time. These metrics are commonly used in classification tasks to quantify the correctness or error of the predictions made by a method. They are defined as follows:

The evaluation metrics used in the experiments include accuracy, precision, recall, F1-score, false positive rate (FPR), true negative rate (TNR), false negative rate (FNR), false discovery rate (FDR), false omission rate (FOR), Matthews correlation coefficient (MCC), training time, and prediction time. These metrics are commonly used in classification tasks to quantify the correctness or error of the predictions made by a method, defined as follows:

- Accuracy: This metric measures the proportion of correctly predicted labels among all predictions, calculated as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{20}$$

  where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.
- Precision: This metric measures the proportion of correctly predicted positive labels among all positive predictions, calculated as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{21}$$

  where TP is the number of true positives and FP is the number of false positives.
- Recall: This metric measures the proportion of correctly predicted positive labels among all actual positive labels, calculated as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{22}$$

  where TP is the number of true positives and FN is the number of false negatives.
- F1-score: This metric measures the harmonic mean of precision and recall, calculated as:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{23}$$

  where Precision and Recall are defined as above.
- FPR: This metric measures the proportion of incorrectly predicted positive labels among all actual negative labels, calculated as:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{24}$$

  where FP is the number of false positives and TN is the number of true negatives.
- TNR: This metric measures the proportion of correctly predicted negative labels among all actual negative labels, calculated as:

$$TNR = \frac{TN}{TN + FP} \tag{25}$$

  where TN is the number of true negatives and FP is the number of false positives.
- FNR: This metric measures the proportion of incorrectly predicted negative labels among all actual positive labels, calculated as:

$$FNR = \frac{FN}{FN + TP} \tag{26}$$

  where FN is the number of false negatives and TP is the number of true positives.
- FDR: This metric measures the proportion of incorrectly predicted positive labels among all positive predictions. It is also known as false alarm ratio or type I error rate, calculated as:

$$FDR = \frac{FP}{FP + TP} \tag{27}$$

  where FP is the number of false positives and TP is the number of true positives.

Table 2
Top 20 selected features for UNSW-NB15 dataset using mutual information method.

Feature number	Feature	Type	Mutual information score
f8	sbytes	Integer	0.978
f23	smeansz	Integer	0.817
f15	sload	Float	0.790
f9	dbytes	Integer	0.510
f14	service	Nominal	0.452
f24	dmeansz	Integer	0.448
f46	ct_dst_sport_ltm	Integer	0.446
f6	srate	Nominal	0.444
f7	dur	Float	0.433
f5	proto	Nominal	0.429
f37	ct_state_ttl	Integer	0.410
f16	dload	Float	0.397
f32	dinpkt	Float	0.397
f45	ct_src_dport_ltm	Integer	0.382
f18	dpkts	Integer	0.379
f42	ct_srv_dst	Integer	0.379
f31	sinpkt	Float	0.372
f11	dttl	Integer	0.370
f34	synack	Float	0.350
f33	tcprtt	Float	0.347

- FOR: This metric measures the proportion of incorrectly predicted negative labels among all negative predictions, calculated as:

$$FOR = \frac{FN}{FN + TN} \tag{28}$$

where FN is the number of false negatives and TN is the number of true negatives.
- MCC: This metric measures the correlation between the predicted and actual values of a classification model. It takes into account the true positives, false positives, true negatives, and false negatives generated by the model. The MCC ranges from −1 to 1, with a higher value indicating a better fit between the predicted and actual values. A perfect model would have an MCC of 1, while a model with no skill would have an MCC of 0. It is calculated as:

$$\text{MCC} = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{29}$$

where TP, TN, FP, and FN are the numbers of true positives, true negatives, false positives, and false negatives, respectively.
- Training time: This is a metric that measures the time required to train a method on a given dataset. It is expressed in seconds.
- Prediction time: This is a metric that measures the time required to make predictions using a trained method on a given dataset. It is expressed in seconds.

5.4. Feature selection

We use mutual information to select relevant features. We rank the features according to their mutual information scores with respect to the class label, and select the top 20 features for each dataset. The selected features for each dataset are shown in Table 2. These features have high mutual information scores with respect to the class label and capture important aspects of the network traffic. For example, the attack category feature indicates the type of attack that an instance belongs to. These features can help the proposed method and other methods to distinguish between normal and anomalous instances more effectively.

5.5. Performance study compared with base learner methods

Experiments were conducted to evaluate each attack category individually, contrasting the proposed method with three foundational learning models: NB, LGBM, and XGB. As detailed in Table 3, the proposed method demonstrated superior detection accuracy across all categories, achieving 96.02% for worms and reaching up to 99.15% for normal traffic. These results testify to the robustness and efficacy of the proposed method, outshining the base learners in recognizing a variety of attack types. Naive Bayes exhibits the lowest accuracy, with performance metrics ranging from 45.67% for worms to 90.42% for normal, reflecting its unsuitability for the dataset due to the inherent assumption of feature independence, which fails to capture complex feature interdependencies. Both LGBM and XGB presented competitive accuracies, with LGBM varying between 80.88% to 98.37%, and XGB spanning from 79.53% to 97.21%. These figures suggest that the gradient boosting and tree-based architectures of these models contribute to their robustness against overfitting, enhancing overall performance. Notably, a decline in classification accuracy was observed as the frequency of the attack category decreased, underscoring the challenges of dealing with imbalanced datasets and the stealthier nature of less common attacks. For instance, the worms category, characterized by its lower occurrence and subtlety,

Table 3
Detection accuracy across different attack categories.

Class	Our method (%)	NB (%)	LGBM (%)	XGB (%)
Normal	99.15	90.42	98.37	97.21
Generic	98.87	85.63	96.11	95.78
Exploits	98.10	80.54	94.29	93.67
Fuzzers	98.75	75.38	92.16	91.43
DoS	97.70	70.91	90.68	89.31
Reconnaissance	97.20	65.47	88.54	87.19
Analysis	96.75	60.84	86.92	85.76
Backdoor	96.30	55.31	84.57	83.46
Shellcode	96.25	50.49	82.12	81.09
Worms	96.02	45.67	80.88	79.53

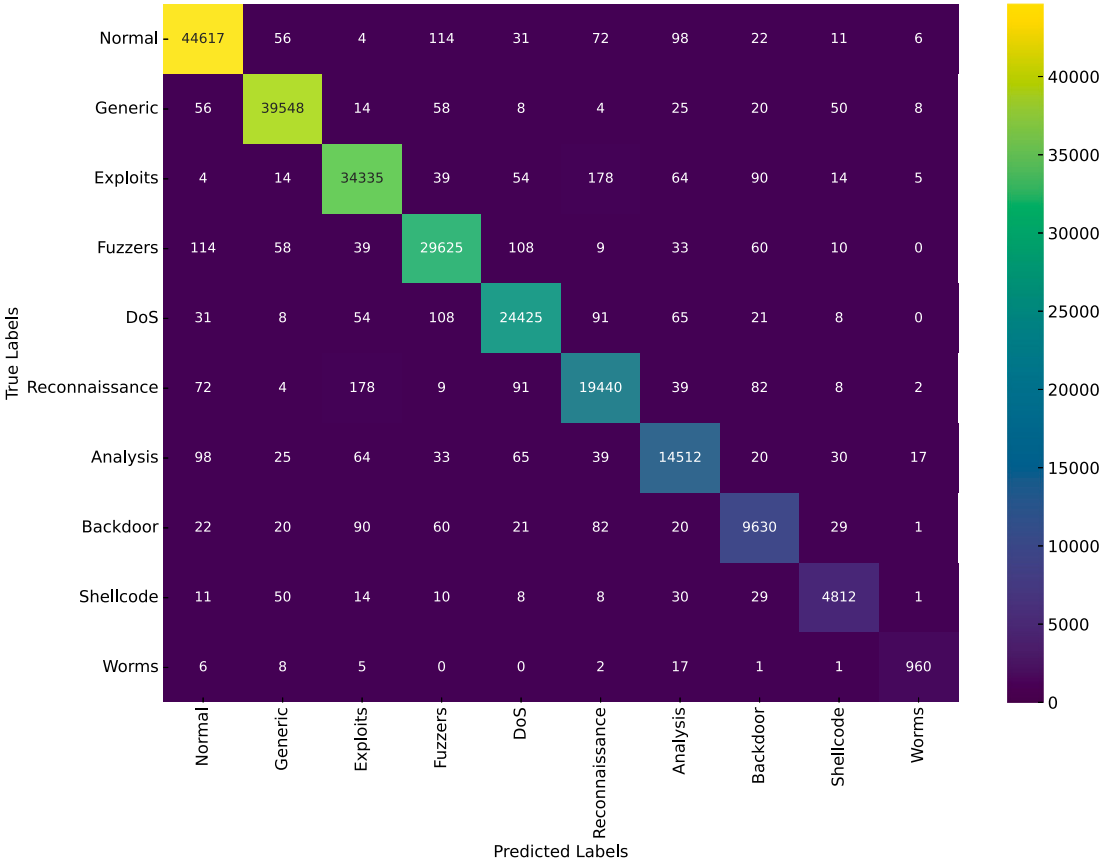

Fig. 4. Confusion matrix heatmap for IoT intrusion detection.

displayed the least accuracy across all models, indicating a potential need for additional data or refined features to bolster classifier performance.

The confusion matrix in Fig. 4 visually encapsulates the classification efficacy of the proposed framework, with a predominantly populated diagonal indicating a high true positive rate. This matrix elucidates the framework's capability to maintain high accuracy in the detection of normal behavior and different cyber threats, despite the class imbalance and the array of attack vectors intrinsic to IoT networks. The sparsity of the off-diagonal elements confirms the model's precision and adaptability, further validating its capacity to distinguish between normal and anomalous traffic effectively.

Next, we evaluate our proposed method and the three base learner methods using ten-fold cross-validation on the dataset. Table 4 displays the average values of accuracy, recall, f-score, and precision for each fold and each method. It can be observed that our proposed method outperformed the other methods in all the performance metrics for all the folds, demonstrating its superiority and effectiveness in detecting intrusions in IoT networks. For instance, in the first fold, our proposed method achieved an accuracy of 97.1%, a recall of 96.5%, an f-score of 95.8%, and a precision of 96.1%, while the best performing other method, LGBM, only achieved an accuracy of 88.2%, a recall of 87.5%, an F-score of 87.9%, and a precision of 88.9%. The gap between our proposed

Table 4
Results of ten-fold cross-validation.

Parameter	Models	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
Accuracy (%)	Our method	97.1	97.2	96.6	96.8	97.0	97.3	97.5	96.9	96.7	97.4
	NB	65.2	64.8	65.1	65.3	65.0	65.4	64.9	65.5	65.6	64.7
	LGBM	88.2	88.4	88.1	88.5	88.3	88.6	88.0	88.7	88.8	88.9
	XGB	87.3	87.6	87.4	87.5	87.7	87.8	87.1	87.9	88.0	87.2
Recall (%)	Our method	96.5	96.6	96.0	97.2	96.4	96.7	96.9	96.3	97.1	95.8
	NB	64.0	63.6	64.2	64.4	63.9	64.3	63.8	64.7	64.8	63.5
	LGBM	87.5	87.7	87.4	87.8	87.6	87.9	87.3	88.0	88.1	88.2
	XGB	86.6	86.9	86.7	86.8	87.0	87.1	86.4	87.2	87.3	86.5
F-score (%)	Our method	95.8	95.9	96.3	96.5	96.7	97.0	96.2	96.6	96.4	97.1
	NB	64.6	64.2	64.7	64.9	64.5	64.9	64.4	65.1	65.2	64.1
	LGBM	87.9	88.1	87.8	88.2	88.0	88.3	87.7	88.4	88.5	88.6
	XGB	87.0	87.3	87.1	87.2	87.4	87.5	86.8	87.6	87.7	86.9
Precision (%)	Our method	96.1	97.2	96.7	96.9	96.1	96.4	95.6	96.0	95.8	96.5
	NB	66.4	66.0	66.3	66.5	66.2	66.6	66.1	66.7	66.8	66.0
	LGBM	88.9	89.1	88.8	89.2	89.0	89.3	88.7	89.4	89.5	89.6
	XGB	88.0	88.3	88.1	88.2	88.4	88.5	87.8	88.6	88.7	87.9

method and other methods widened in the subsequent folds, reaching the highest difference in the tenth fold, where our proposed method attained an accuracy of 97.4%, a recall of 95.8%, an F-score of 97.1%, and a precision of 96.5%, while the worst performing other method, NB, only attained an accuracy of 64.7%, a recall of 63.5%, an F-score of 64.1%, and a precision of 66%. These results validate that our proposed method can effectively learn from the subspace clusters and the ensemble learners, and achieve high accuracy and robustness in intrusion detection for IoT networks.

We continue to conduct the performance comparison based on the following metrics: (1) accuracy, precision, recall, and F1-score; (2) true positive rate (TPR), true negative rate (TNR), and Matthews correlation coefficient (MCC); (3) false positive rate (FPR), false negative rate (FNR), false discovery rate (FDR), and false omission rate (FOR); and (4) testing time. The results are as follows:

- Fig. 5 shows the plot of accuracy, recall, f-score, and precision for each method. The plot shows how well the methods can correctly classify the instances as normal or attack. Our proposed method achieved the highest values for all these metrics, indicating that it can effectively identify both known and unknown attacks in IoT networks. For example, our proposed method achieved an accuracy of 97.05%, which means that it correctly classified 97.05% of the instances in the dataset. The other methods had lower accuracy values, ranging from 65.15% to 88.45%, which means that they made more errors in classifying the instances.
- Fig. 6 compares three metrics of false positives and false negatives for four methods of intrusion detection in IoT networks. False positives are normal instances misclassified as attack, while false negatives are attack instances misclassified as normal. The proposed method has the highest values for all metrics, showing that it minimizes both false positives and false negatives. For instance, it has a TPR of 96.55%, meaning it detects 96.55% of the actual attacks. The other methods have lower TPR values, from 64.12% to 87.75%, meaning they miss more attacks. The proposed method uses a novel ensemble learning model that combines subspace clustering and meta-learning to enhance the feature space and accuracy. It also has the highest TNR of 96.50% and MCC of 95.58%, while the worst method, NB, has the lowest TNR of 64.00% and MCC of 60.67%. The results show that the proposed method is the best at identifying both known and unknown attacks in IoT networks, and achieving a good balance between false positives and false negatives.
- Fig. 7 compares four metrics of incorrect predictions for four methods of intrusion detection in IoT networks. Lower values mean better performance. The proposed method has the lowest values for all metrics, showing that it makes the fewest mistakes for both normal and anomalous instances. For instance, it has a FPR of 0.029, meaning only 2.9% of normal instances are misclassified as anomalous. The other methods have higher FPR values, from 0.042 to 0.191, showing more false alarms. NB has the highest values for all metrics, showing that it makes the most mistakes for both normal and anomalous instances. For example, it has a FNR of 0.202, meaning 20.2% of anomalous instances are misclassified as normal. The other methods have lower FNR values, from 0.018 to 0.155, showing fewer missed attacks. XGB and LGBM have similar performance for all metrics, except for FNR, where LGBM is much lower than XGB. This shows that LGBM is better at detecting anomalies, while XGB is better at avoiding false alarms. The results show that the proposed method is the best at minimizing incorrect predictions for intrusion detection in IoT networks.
- Fig. 8 compares the testing time of our proposed method and the base learn methods. The testing time is a metric that measures the time taken by a method to make predictions on the test data. Among the four methods, NB has the lowest testing time of 19.1 ms, due to its simple and naive algorithm that assumes feature independence and does not require any optimization or tuning. However, this also results in the lowest accuracy and performance. On the other hand, the proposed method has the highest testing time of 54.24 ms, due to its complex and intensive techniques such as subspace clustering, ensemble learning, and hyperparameter optimization. However, this also leads to the highest accuracy and performance. XGB and LGBM have similar testing times of 24.6 ms and 26.6 ms, respectively, which are faster than the proposed method but slower than NB. They use gradient boosting and tree-based algorithms that are powerful and regularized, but also need some optimization

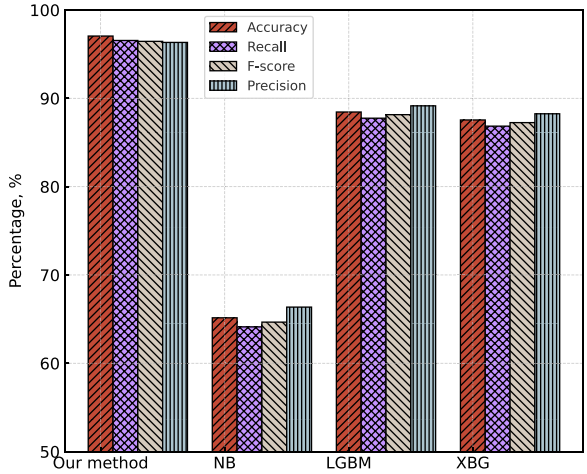

Fig. 5. Accuracy, precision, recall, and F1-score analysis.

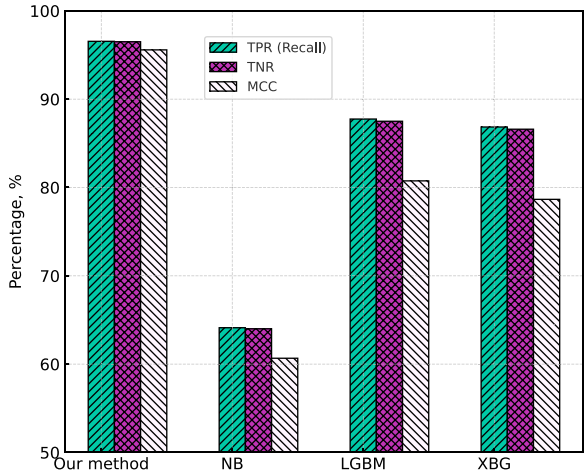

Fig. 6. TPR, TNR, and MCC analysis.

or tuning. They also have similar accuracy and performance. The results demonstrate a trade-off between testing time and accuracy or performance for intrusion detection in IoT networks.

5.6. Ablation study

We conduct an ablation study to evaluate the effectiveness of our method's design. Table 5 shows the performance of different combinations of the three synergy strategies: clustering results as features (CRF), two-level decision making (TDM), and iterative feedback loop (IFL). The results show that the proposed method, which combines all three strategies (CRF+TDM+IFL), outperforms the others by a large margin on all performance metrics. For example, it achieves an accuracy of 97.05%, a precision of 96.33%, a recall of 96.55%, an f1-score of 96.45%, and an MCC of 95.58%. These results indicate that the proposed method can correctly classify most of the instances as normal or anomalous, has a high correlation between the actual and predicted labels, and minimizes the false positive rate (FPR) of 0.029. Among the different strategies, we can see that using all three together is the best combination for the UNSW-NB15 dataset, as it achieves the highest values for all performance metrics. This suggests that CRF, TDM, and IFL can enhance the feature space and improve the accuracy more effectively than using any two of them. However, using all three together also increases the computational complexity compared to using only one strategy.

Next, we evaluate the impact of bootstrapping on our method's performance. Table 6 shows the detection accuracy for different attack categories with and without bootstrapping. Bootstrapping improves the accuracy for all categories, reaching above 95% for each one. The Normal category has the highest accuracy of 99.15% with bootstrapping, showing that the model can identify most of the normal traffic correctly. Bootstrapping also affects different categories differently, depending on their frequency. For frequent

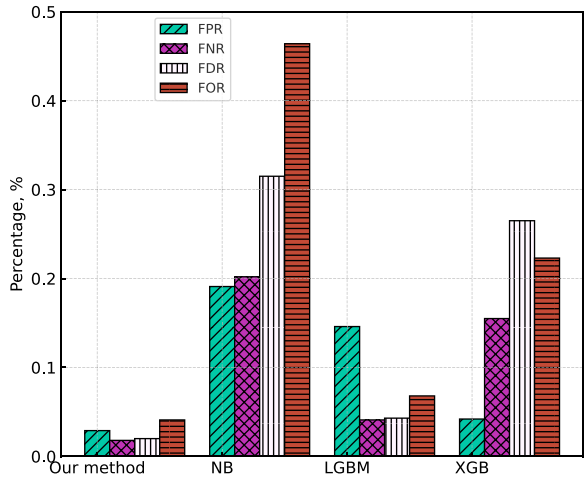

Fig. 7. FPR, FNR, FDR, FOR analysis.

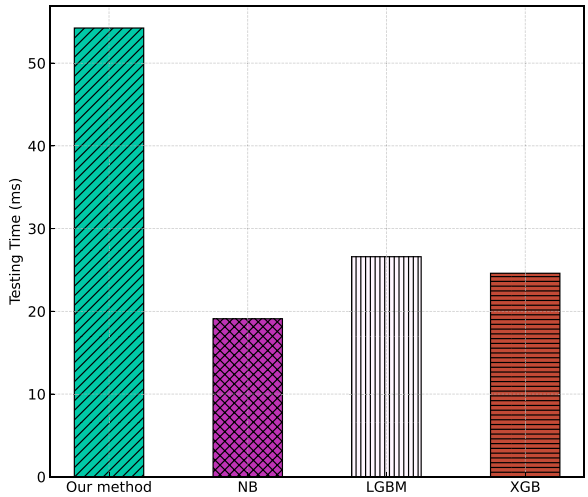

Fig. 8. Testing time analysis.

Table 5
Evaluation of the impact of different synergy strategies on the performance.

Synergies	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	FPR (%)	MCC (%)
CRF	86.21	86.37	87.45	86.12	0.138	72.09
TDM	87.16	87.43	88.21	87.34	0.128	74.47
IFL	85.38	85.19	86.27	85.43	0.146	70.32
CRF+TDM	89.78	89.52	90.86	89.67	0.102	80.19
CRF+IFL	88.44	88.11	89.13	88.38	0.116	76.06
TDM+IFL	89.53	89.27	90.15	89.39	0.105	79.21
CRF+TDM+IFL	97.05	96.33	96.55	96.45	0.029	95.58

categories, such as Normal, Generic, Exploits, and Fuzzers, bootstrapping has a minor effect on the accuracy, lowering it slightly when not applied. This implies that the model is stable and robust for these categories, and does not need bootstrapping to reduce the variance. For rare categories, such as DoS, Reconnaissance, Analysis, Backdoor, Shellcode, and Worms, bootstrapping has a major effect on the accuracy, dropping it significantly when not applied. This suggests that the model is unstable and sensitive for these categories, and needs bootstrapping to stabilize the statistics. The results show the importance of bootstrapping for enhancing the reliability and performance of the model across diverse attack vectors. Bootstrapping is beneficial for all categories, but essential for those with fewer instances to maintain a high level of detection accuracy.

Table 6
Comparison of detection accuracy across different attack categories, highlighting the impact of bootstrapping.

Class	With bootstrap (%)	Without bootstrap (%)
Normal	99.15	99.00
Generic	98.87	98.70
Exploits	98.10	97.90
Fuzzers	98.75	98.60
DoS	97.70	96.80
Reconnaissance	97.20	96.10
Analysis	96.75	95.60
Backdoor	96.30	95.10
Shellcode	96.25	95.05
Worms	96.02	94.82

Table 7
Comparison of the proposed method with state-of-the-art methods using UNSW-NB15 dataset.

Ref.	Classifier	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
[57]	CNN-LSTM	93.70	95.56	94.9	95.0
[10]	Stacking (DT, RF, XGBoost)	94	–	94	–
[58]	Stacking (RF,LR, KNN, SVM)	94	–	93	–
[59]	IGRF-RFE	84.24	–	–	82.85
[60]	MLP-IG	84	–	–	–
[61]	ML (reduced feature selection)	96.76	–	–	86.06
Our method	Subspace clustering + Ensemble	97.05	96.33	96.55	96.45

5.7. Performance comparison with state-of-the-art methods

In this section, we compare the performance of the proposed method with six state-of-the-art methods that use different classifiers for intrusion detection in IoT networks using the UNSW-NB15 dataset. We chose these methods because they represent the most recent and relevant works in the literature, and they have reported their results using the same dataset as ours. Moreover, these methods use different types of classifiers, such as deep learning, machine learning, and hybrid techniques, which allow us to evaluate the effectiveness and efficiency of our proposed method. We also refer to some sources that review and compare these methods, and show their strengths and weaknesses [3,62]. Our proposed method is related to these methods in that it also uses a classifier for intrusion detection, but it differs from them in that it uses subspace clustering and ensemble learning techniques, and integrates them in a synergistic way using three strategies: clustering results as features, two-level decision making, and iterative feedback loop.

Table 7 shows the comparison of four performance metrics: accuracy, precision, recall, and F1-score. Some of the metric values are not available from the surveyed literature, denoted by "–". The proposed method achieves the highest accuracy of 97.05%, which is 0.29% higher than the second-best method [61]. Moreover, the proposed method surpasses the recent work by [57], who used CNN-LSTM technique, by 3.98% in accuracy, 0.77% in precision, 1.65% in recall, and 1.45% in F1-score. Furthermore, we compare the proposed method with other commonly used techniques such as stacking, proposed by [10,58]. The proposed method enhances its accuracy by 3.05% and 3.05% and its recall by 2.55% and 3.55% compared to [10,58], respectively. The results demonstrate that the proposed method can effectively and accurately detect both known and unknown attacks in IoT networks.

6. Discussion

This section discusses the main findings, implications, opportunities, and future directions of our proposed method for intrusion detection in IoT networks using subspace clustering and ensemble learning.

We begin by summarizing the main findings of our study. Our proposed method is a novel intrusion detection framework for IoT networks that integrates subspace clustering and ensemble learning techniques through three synergy strategies: clustering results as features (CRF), two-level decision making (TDM), and iterative feedback loop (IFL). We also use particle swarm optimization (PSO) to optimize the hyperparameters of the base learners and the meta-learner in the ensemble learning model. We evaluate our method on the UNSW-NB15 dataset, which is a realistic and comprehensive dataset for network intrusion detection systems. We compare our method with six state-of-the-art methods that use different classifiers for intrusion detection in IoT networks. The results show that our method outperforms the other methods by a large margin on all performance metrics, such as accuracy, precision, recall, F1-score, FPR, TNR, and MCC. These results indicate that our method can correctly classify most of the instances as normal or anomalous, has a high correlation between the actual and predicted labels, and minimizes the false alarms. Moreover, our method achieves high detection accuracy for all attack categories, reaching above 95% for each one. This shows that our method can effectively identify both known and unknown attacks for IoT networks.

Next, we discuss the implications of our method for both practice and theory. Our method has several practical implications for IoT network security. First, it can help IoT network administrators and security analysts to monitor network traffic and

identify intrusions more effectively and efficiently, as it has high accuracy and low FPR. This can reduce the workload and cost of manual inspection and verification, and increase the confidence and trust in the detection results. Second, it can help IoT device manufacturers and service providers to improve the security and reliability of their products and services, as it can cope with the challenges of high-dimensional, imbalanced, and heterogeneous data in IoT networks. This can enhance the compatibility and interoperability of different IoT devices and applications, and increase the customer satisfaction and loyalty. Third, it can help policymakers and regulators to establish standards and guidelines for IoT network security, as it can provide insights into the characteristics and behaviors of normal and anomalous connections in IoT networks. This can facilitate the development and enforcement of effective policies and regulations to protect the IoT devices and users from malicious attacks. Our method also has several theoretical implications for intrusion detection research. First, it advances the understanding of subspace clustering techniques for anomaly detection, as it uses four subspace clustering algorithms to discover meaningful clusters in different feature subspaces, which can capture the local patterns and structures of normal and anomalous connections. This can provide a new perspective and approach for analyzing high-dimensional data with complex distributions and relationships. Second, it challenges the existing models of ensemble learning for classification, as it uses a two-tiered model of base learners and a meta-learner to improve the performance and robustness of the detection model. This can provide a new framework and technique for combining multiple classifiers with different strengths and weaknesses. Third, it contributes to the development of new strategies for the integration of subspace clustering and ensemble learning techniques, as it uses three synergistic strategies: CRF, TDM, and IFL. These strategies not only enhance the feature space but also improve the accuracy of intrusion detection more effectively than using any two of them.

Finally, we discuss the opportunities, limitations, and future directions of our method. Our method has some potential areas for improvement that can be addressed in future work. One area is the feature selection method, which we use to reduce the dimensionality and complexity of the network data. We use mutual information, a measure of statistical dependence between two variables, to select the most relevant features for intrusion detection. However, mutual information may not capture all nonlinear dependencies in complex network data, and may introduce some redundancy or noise in the feature space. This may affect the performance and robustness of the proposed method, as it may miss some important features or include some irrelevant features for intrusion detection. This also poses a threat to the internal validity of our results, as they may not be free from errors or biases, and may not be attributed to the proposed method rather than other factors. To address this opportunity and threat, future work can explore more advanced feature selection methods, such as correlation-based, wrapper-based, or embedded methods, to optimize the feature space and improve the detection performance. Future work can also perform a sensitivity analysis to evaluate the impact of different feature subsets on the detection performance, using different feature selection methods, such as chi-square, information gain, and relief, to select different numbers of features, and compare the results with the mutual information method. Another area is the optimization method, which we use to tune the hyperparameters of the base learners and the meta-learner in our ensemble learning model. We use PSO, a population-based metaheuristic algorithm, to find the optimal hyperparameter values that maximize the F1-score. However, PSO may suffer from premature convergence or stagnation, and may not find the global optimum in complex search spaces. This may affect the performance and robustness of the proposed method, as it may not find the optimal hyperparameter values that maximize the detection performance. This also poses a threat to the internal validity of our results, as they may not be free from errors or biases, and may not be attributed to the proposed method rather than other factors. To address this opportunity and threat, future work can employ other optimization methods, such as grid search, random search, or Bayesian optimization, to find the optimal hyperparameter values more efficiently and reliably. Future work can also perform a stability analysis to evaluate the impact of different optimization methods on the detection performance, using different optimization methods, such as grid search, random search, and Bayesian optimization, to find the optimal hyperparameter values for the base learners and the meta-learner, and compare the results with the PSO method. A third area is the generalizability and applicability of our method to other IoT network scenarios, which may differ from the UNSW-NB15 dataset in terms of network type, size, topology, characteristics and capabilities of devices, and the objectives and strategies of attackers. This poses a threat to the external validity of our results, as they may not be generalized and applied to other contexts or situations. To address this opportunity and threat, future work can test our method on other datasets and real-world scenarios that share similar characteristics and challenges with the UNSW-NB15 dataset, such as high-dimensional, imbalanced, and heterogeneous data, and various types of known and unknown attacks. For example, future work can apply our method to other datasets. However, some preprocessing steps, such as feature selection, normalization, and encoding, may be required to adapt the data to our method. Additionally, future work can apply our method to real-world IoT networks in settings like smart homes, smart cities, and smart healthcare, which can provide valuable insights into its performance and scalability in these contexts. However, some challenges, such as data collection, labeling, and privacy, may arise in these settings, and need to be addressed accordingly.

In conclusion, our method is a significant step forward in IoT network security, but it also has some potential areas for improvement and challenges that need to be addressed in future work. Future work will focus on enhancing the feature selection capabilities, optimizing the hyperparameter tuning, testing the method on other datasets and real-world scenarios, and ensuring the ethical and responsible application of the method. These efforts are crucial in maintaining the method's relevance and effectiveness in the rapidly changing and challenging landscape of network security.

7. Conclusions

Intrusion detection stands as a vital task for fortifying the security and reliability of IoT networks, which are susceptible to a myriad of attacks. In this paper, we have proposed a novel framework based on subspace clustering and ensemble learning, incorporating three synergy strategies: Clustering Results as Features (CRF), Two-Level Decision Making (TDM), and Iterative Feedback

Loop (IFL). We evaluated our approach to empirical validation using the UNSW-NB15 dataset, a repository featuring a realistic and heterogeneous IoT network traffic data. The comparative analysis revealed that the proposed methodology outperformed existing state-of-the-art techniques across multiple evaluation metrics, including accuracy, precision, recall, and F1-score. Specifically, our approach attained an accuracy rate of 97.05%, a precision rate of 96.33%, a recall rate of 96.55%, and an F1-score of 96.45%. Furthermore, the proposed method achieved the lowest false positive rate, a mere 0.029, among all evaluated methods. This underscores its efficacy in minimizing false alarms, thereby reducing the financial and operational cost associated with unwarranted actions or investigations.

CRediT authorship contribution statement

Jingyi Zhu: Conceptualization, Methodology, Software, Investigation, Formal analysis, Writing – original draft. **Xiufeng Liu:** Conceptualization, Formal analysis, Resources, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Appendix. List of abbreviations

Table A.8
List of abbreviations and their descriptions.

Abbreviation	Full form	Description
CLIQUE	Clustering In QUEst	A subspace clustering algorithm for identifying dense clusters in high-dimensional data
CRF	Clustering Results as Features	A strategy used in our model for enhancing feature space based on clustering results
IDS	Intrusion Detection System	A system or software application for monitoring network or system activities for security breaches
IFL	Iterative Feedback Loop	A technique for dynamic model recalibration based on real-time results
IoT	Internet of Things	A system of interconnected computing devices and objects that can transfer data over a network
LGBM	Light Gradient Boosting Machine	A gradient boosting framework that uses tree-based learning algorithms
LOF	Local Outlier Factor	An algorithm to identify density-based local outliers in data
LR	Logistic Regression	A statistical model used for binary classification
NB	Naive Bayes	A probabilistic classifier based on Bayes' theorem with strong independence assumptions
PROCLUS	Projected CLUStering	A projected clustering algorithm
PSO	Particle Swarm Optimization	A computational method used for optimizing a problem by iteratively improving candidate solutions
SC	Subspace Clustering	A clustering technique that finds clusters in different subspaces of the data
SUBCLU	SUBspace CLUstering	A density-based algorithm for clustering in a subspatial manner
TDM	Two-Level Decision Making	A decision-making strategy that uses both subspace clustering and ensemble learning
XGB	eXtreme Gradient Boosting	An optimized gradient boosting library

References

[1] Al-Yaseen WL, Othman ZA, Nazri MZA. Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. Expert Syst Appl 2017;67:296–303.
[2] Makkar A, Park JH. SecureCPS: Cognitive inspired framework for detection of cyber attacks in cyber–physical systems. Inf Process Manage 2022;59(3):102914.
[3] Mohammadpour L, Ling TC, Liew CS, Aryanfar A. A survey of CNN-based network intrusion detection. Appl Sci 2022;12(16):8162.
[4] Vaigandla K, Azmi N, Karne R. Investigation on intrusion detection systems (IDSs) in IoT. Int J Emerg Trends Eng Res 2022;10(3).
[5] Nayak G, Mishra A, Samal U, Mishra BK. Depth analysis on DoS & DDoS attacks. Wireless Commun Secur 2022;159–82.
[6] Kumar KR, Nakkeeran R. A comprehensive study on denial of service (DoS) based on feature selection of a given set datasets in internet of things (IoT). In: 2023 international conference on signal processing, computation, electronics, power and telecommunication. iConSCEPT, IEEE; 2023, p. 1–8.
[7] Sikos LF. Packet analysis for network forensics: A comprehensive survey. Forensic Sci Int.: Digit Invest 2020;32:200892.
[8] Farooq M, Khan MH. Signature-based intrusion detection system in wireless 6G IoT networks. J Internet Things 2022;4(3).
[9] Kolias C, Kambourakis G, Stavrou A, Voas J. DDoS in the IoT: Mirai and other botnets. Computer 2017;50(7):80–4.
[10] Rashid M, Kamruzzaman J, Imam T, Wibowo S, Gordon S. A tree-based stacking ensemble technique with feature selection for network intrusion detection. Appl Intell 2022;52(9):9768–81.
[11] Bhandari R, Singla S, Sharma P, Kang SS. AINIS: An intelligent network intrusion system. Int J Perform Eng 2024;20(1):24.
[12] Song X, Ma Q. Intrusion detection using federated attention neural network for edge enabled internet of things. J Grid Comput 2024;22(1):1–17.
[13] Boopathi M. An intrusion detection system for IoT using deep learning and optimization techniques. 2024.
[14] Chinaechetam EN, Nwakanma CI, Lee J-M, Kim D-S. Detecting cyberthreats in metaverse learning platforms using an explainable DNN. Internet Things 2024;101046.
[15] Roopak M, Parkinson S, Tian GY, Ran Y, Khan S, Chandrasekaran B. An unsupervised approach for the detection of zero-day DDoS attacks in IoT networks.

[16] Devendiran R, Turukmane AV. Dugat-LSTM: Deep learning based network intrusion detection system using chaotic optimization strategy. Expert Syst Appl 2024;245:123027.
[17] Gavel S, Raghuvanshi AS, Tiwari S. Maximum correlation based mutual information scheme for intrusion detection in the data networks. Expert Syst Appl 2022;189:116089.
[18] Agrawal R, Gehrke J, Gunopulos D, Raghavan P. Automatic subspace clustering of high dimensional data for data mining applications. In: Proceedings of the 1998 ACM SIGMOD international conference on management of data. 1998, p. 94–105.
[19] Aggarwal CC, Wolf JL, Yu PS, Procopiuc C, Park JS. Fast algorithms for projected clustering. ACM SIGMoD Rec 1999;28(2):61–72.
[20] Kailing K, Kriegel H-P, Kröger P. Density-connected subspace clustering for high-dimensional data. In: Proceedings of the 2004 SIAM international conference on data mining. SIAM; 2004, p. 246–56.
[21] Breunig MM, Kriegel H-P, Ng RT, Sander J. LOF: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on management of data. 2000, p. 93–104.
[22] Webb GI, Keogh E, Miikkulainen R. Naïve Bayes. Encyclopedia Mach Learn 2010;15(1):713–4.
[23] Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu T-Y. Lightgbm: A highly efficient gradient boosting decision tree. Adv Neural Inf Process Syst 2017;30.
[24] Chen T, He T, Benesty M, Khotilovich V, Tang Y, Cho H, Chen K, Mitchell R, Cano I, Zhou T, et al. Xgboost: extreme gradient boosting. 2015, p. 1–4, R package version 0.4-2 1 (4).
[25] Hosmer Jr DW, Lemeshow S, Sturdivant RX. Applied logistic regression, vol. 398, John Wiley & Sons; 2013.
[26] Marini F, Walczak B. Particle swarm optimization (PSO). A tutorial. Chemometr Intell Lab Syst 2015;149:153–65.
[27] Sadikin F, Van Deursen T, Kumar S. A ZigBee intrusion detection system for IoT using secure and efficient data collection. Internet Things 2020;12:100306.
[28] Ring M, Wunderlich S, Scheuring D, Landes D, Hotho A. A survey of network-based intrusion detection data sets. Comput Secur 2019;86:147–67.
[29] Mrabet H, Belguith S, Alhomoud A, Jemai A. A survey of IoT security based on a layered architecture of sensing and data analysis. Sensors 2020;20(13):3625.
[30] Elrawy MF, Awad AI, Hamed HF. Intrusion detection systems for IoT-based smart environments: a survey. J Cloud Comput 2018;7(1):1–20.
[31] Maseer ZK, Yusof R, Bahaman N, Mostafa SA, Foozy CFM. Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset. IEEE Access 2021;9:22351–70.
[32] Wei N, Yin L, Zhou X, Ruan C, Wei Y, Luo X, Chang Y, Li Z. A feature enhancement-based model for the malicious traffic detection with small-scale imbalanced dataset. Inform Sci 2023;119512.
[33] Chen J, Qi X, Chen L, Chen F, Cheng G. Quantum-inspired ant lion optimized hybrid k-means for cluster analysis and intrusion detection. Knowl-Based Syst 2020;203:106167.
[34] Chapagain P, Timalsina A, Bhandari M, Chitrakar R. Intrusion detection based on PCA with improved K-means. In: International conference on electrical and electronics engineering. Springer; 2022, p. 13–27.
[35] Ding H, Chen L, Dong L, Fu Z, Cui X. Imbalanced data classification: A KNN and generative adversarial networks-based hybrid approach for intrusion detection. Future Gener Comput Syst 2022;131:240–54.
[36] Sanju P. Enhancing intrusion detection in IoT systems: A hybrid metaheuristics-deep learning approach with ensemble of recurrent neural networks. J Eng Res 2023;100122.
[37] Ge M, Fu X, Syed N, Baig Z, Teo G, Robles-Kelly A. Deep learning-based intrusion detection for IoT networks. In: 2019 IEEE 24th Pacific rim international symposium on dependable computing. PRDC, IEEE; 2019, p. 256–25609.
[38] Li J, Zhao Z, Li R, Zhang H. Ai-based two-stage intrusion detection for software defined iot networks. IEEE Internet Things J 2018;6(2):2093–102.
[39] Zarpelão BB, Miani RS, Kawakani CT, de Alvarenga SC. A survey of intrusion detection in Internet of Things. J Netw Comput Appl 2017;84:25–37.
[40] Hodo E, Bellekens X, Hamilton A, Dubouilh P-L, Iorkyase E, Tachtatzis C, Atkinson R. Threat analysis of IoT networks using artificial neural network intrusion detection system. In: 2016 international symposium on networks, computers and communications. ISNCC, IEEE; 2016, p. 1–6.
[41] Ahakonye LAC, Nwakanma CI, Lee J-M, Kim D-S. SCADA intrusion detection scheme exploiting the fusion of modified decision tree and chi-square feature selection. Internet Things 2023;21:100676.
[42] Uzun B, Ballı S. A novel method for intrusion detection in computer networks by identifying multivariate outliers and relieff feature selection. Neural Comput Appl 2022;34(20):17647–62.
[43] Bhattacharya S, Maddikunta PKR, Kaluri R, Singh S, Gadekallu TR, Alazab M, Tariq U. A novel PCA-firefly based XGBoost classification model for intrusion detection in networks using GPU. Electronics 2020;9(2):219.
[44] Andresini G, Appice A, Malerba D. Autoencoder-based deep metric learning for network intrusion detection. Inform Sci 2021;569:706–27.
[45] Hamid Y, Sugumaran M. A t-SNE based non linear dimension reduction for network intrusion detection. Int J Inf Technol 2020;12:125–34.
[46] Zhang R, Zhang J, Wang Q, Zhang H. DOIDS: an intrusion detection scheme based on DBSCAN for opportunistic routing in underwater wireless sensor networks. Sensors 2023;23(4):2096.
[47] Wang W, Du X, Shan D, Qin R, Wang N. Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine. IEEE Trans Cloud Comput 2020;10(3):1634–46.
[48] Jin D, Lu Y, Qin J, Cheng Z, Mao Z. SwiftIDS: Real-time intrusion detection system based on LightGBM and parallel intrusion detection mechanism. Comput Secur 2020;97:101984.
[49] Wei Y, Wu F. A self-adaptive intrusion detection model based on bi-LSTM-CRF with historical access logs. In: Advances in natural computation, fuzzy systems and knowledge discovery: proceedings of the ICNC-fSKD 2021 17. Springer; 2022, p. 185–97.
[50] Madhu G. Design of intrusion detection and prevention model using COOT optimization and hybrid LSTM-KNN classifier for MANET. EAI Endors Trans Scalable Inf Syst 2022;10(3).
[51] Das SK, Dey N, Crespo RG, Herrera-Viedma E. A non-linear multi-objective technique for hybrid peer-to-peer communication. Inform Sci 2023;629:413–39.
[52] Hou C, Nie F, Jiao Y, Zhang C, Wu Y. Learning a subspace for clustering via pattern shrinking. Inf Process Manage 2013;49(4):871–83.
[53] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, et al. Scikit-learn: Machine learning in python. J Mach Learn Res 2011;12:2825–30.
[54] Kulesa A, Krzywinski M, Blainey P, Altman N. Sampling distributions and the bootstrap. 2015.
[55] Kumar G, Thakur K, Ayyagari MR. MLEsIDSs: machine learning-based ensembles for intrusion detection systems—a review. J Supercomput 2020;76:8938–71.
[56] Moustafa N, Slay J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 military communications and information systems conference. milCIS, IEEE; 2015, p. 1–6.
[57] Halbouni A, Gunawan TS, Habaebi MH, Halbouni M, Kartiwi M, Ahmad R. CNN-LSTM: hybrid deep neural network for network intrusion detection system. IEEE Access 2022;10:99837–49.
[58] Rajagopal S, Kundapur PP, Hareesha KS. A stacking ensemble for network intrusion detection using heterogeneous datasets. Secur Commun Netw 2020;2020:1–9.
[59] Yin Y, Jang-Jaccard J, Xu W, Singh A, Zhu J, Sabrina F, Kwak J. IGRF-RFE: a hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset. J Big Data 2023;10(1):1–26.
[60] Roy A, Singh KJ. Multi-classification of unsw-nb15 dataset for network anomaly detection system. In: Proceedings of international conference on communication and computational technologies. ICCCT-2019, Springer; 2021, p. 429–51.
[61] Kasongo SM, Sun Y. Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset. J Big Data 2020;7:1–20.
[62] Khraisat A, Alazab A. A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. Cybersecurity 2021;4:1–27.