Egyptian Informatics Journal 27 (2024) 100521

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Full length article

Enhanced threat intelligence framework for advanced cybersecurity resilience

Moutaz Alazab [a,*], Ruba Abu Khurma [b], Maribel García-Arenas [c], Vansh Jatana [d], Ali Baydoun [e], Robertas Damaševičius [f]

[a] Department of Intelligent System, Faculty of Artificial Intelligence, Albalqa Applied University, Al-Salt, 19117, Jordan
[b] Information Technology Department, Al-Huson University College, Albalqa Applied University, Irbid, 19117, Jordan
[c] Department of Computer Engineering, Automatics and Robotics, University of Granada, Granada, 18071, Spain
[d] School of CSE, SRM University, Kattankulathur, India
[e] School of Computing and Data Sciences, Oryx Universal College with Liverpool John Moores University, Qatar
[f] Center of Real Time Computer Systems, Kaunas University of Technology, Kaunas, 19117, Lithuania

ARTICLE INFO

Keywords:
Cybersecurity
Threat intelligence
Network intrusion
Mitigation and response
Cyber attacks
Data breaches
Threat landscape

ABSTRACT

The increasing severity of cyber-attacks against organizations emphasizes the necessity for efficient threat intelligence. This article presents a novel multi-layered architecture for threat intelligence that integrates diverse data streams, including corporate network logs, open-source intelligence, and dark web monitoring, to offer a comprehensive overview of the cybersecurity threat landscape. Our approach, distinct from previous studies, uniquely integrates these varied features into the machine-learning algorithms (XGBoost, Gradient Boosting, LightGBM, Extra Trees, Random Forest, Decision Tree, K-Nearest Neighbor, Gaussian Naive Bayes, Support Vector Machine, Linear Discriminant Analysis, Logistic Regression, ridge Classifier, AdaBoost and Quadratic Discriminant Analysis) using various feature selection algorithms (information gain, correlation coefficient, chi-square, fisher score, forward wrapper, backward wrapper, Ridge classifier) to enhance real-time threat detection and mitigation. The practical LITNET-2020 dataset was utilized to evaluate the proposed architecture. Extensive testing against real-world cyber-attacks, including malware and phishing, demonstrated the robustness of the architecture, achieving exceptional results. Specifically, XGBoost demonstrated the highest performance with a detection accuracy of 99.98%, precision of 99.97%, and recall of 99.96%, Significantly surpassing traditional methods. Gradient Boosting and LightGBM also exhibited excellent performance, with accuracy, precision, and recall values of 99.97%. Our findings underscore the effectiveness of our architecture in significantly improving an organization's capability to identify and counteract online threats in real-time. By developing a comprehensive threat intelligence framework, this study advances the field of cybersecurity, providing a robust tool for enhancing organizational resilience against cyber-attacks.

1. Introduction

The rapid expansion of the Internet and the proliferation of connected devices have increased cybersecurity concerns for organizations and individuals alike [1]. Cyber attacks, including data breaches, financial losses, and reputation damage, underscore the critical need for robust cybersecurity measures [2,3]. This challenge is exacerbated by diverse networked systems such as Cyber–Physical Systems, Mobile Ad Hoc Networks, Internet of Things, and Wireless Sensor Networks, each introducing unique vulnerabilities and attack vectors [4,5].

In critical infrastructures such as smart grids and autonomous cars, CPS merges physical processes with computer algorithms, resulting in vulnerabilities where software compromises can have physical repercussions [6–8]. Due to its decentralized structure, vulnerability to particular dangers to wireless networks, and susceptibility to attacks such as the Sybil attack, MANET dynamic networks lacking fixed infrastructure face security issues [9]. Despite their ease of use, Internet of Things (IoT) devices frequently have weak security protocols, making them vulnerable to more serious network attacks such as distributed

* Corresponding author.
E-mail addresses: m.alazab@bau.edu.jo (M. Alazab), ruba_abukhurma@bau.edu.jo (R.A. Khurma), Mgarenas@ugr.es (M. García-Arenas), vs7182@srmist.edu.in (V. Jatana), ali.b@oryx.edu.qa (A. Baydoun), robertas.damasevicius@ktu.lt (R. Damaševičius).

https://doi.org/10.1016/j.eij.2024.100521
Received 9 December 2023; Received in revised form 11 July 2024; Accepted 30 July 2024
Available online 10 September 2024

denial of service (DDoS) [10–12]. WSNs, which are intended to monitor physical condition, have challenges related to limited resources and security flaws, such as denial-of-service attacks and data intercept [13].

Traditional cybersecurity approaches struggle against evolving threats [14–16]. Tailored strategies are imperative, leveraging artificial intelligence (AI) to enhance threat detection and response capabilities [17–19]. Artificial intelligence's (AI) capacity for real-time anomaly detection and predictive analysis fills in the gaps in conventional techniques by spotting new dangers and allocating resources optimally for optimal defense [20,21].

This paper presents the Multi-Layered Threat Intelligence Framework (MLTIF), which is a holistic strategy to unify these diverse threat intelligence components and is capable of integrating open-source feeds, dark web monitoring sources as well as internal network log for an enhanced capability in understanding the potential threats against organizations [22,23]. The MLTIF employs multiple detection techniques, including signature-based and machine learning-based approaches, to prioritize threats effectively [24,25]. Furthermore, it integrates mitigation strategies like blocking and incident response for swift threat containment [12].

By offering a thorough method for threat identification and mitigation, the MLTIF fills in the gaps in the present cybersecurity frameworks. This introduction lays the ground for a discussion of that approach. The MLTIF's potential to improve organizational cybersecurity resilience is demonstrated in the following sections, which explore the architecture, implementation, evaluation, and conclusions of the framework.

Developing a multi-layered threat intelligence architecture for cybersecurity raises the following three potential research questions:

- How can a full picture of potential threats be provided by the Threat Intelligence Framework (MLTIF) by efficiently integrating many threat intelligence sources, such as internal network logs, dark web monitoring, and open-source feeds?
- What are the best ways to embrace threat detection and mitigation strategies in MLTIF architecture, to detect cybersecurity including malware, phishing, and social engineering attacks?
- How much does the MLTIF design enhance an organization's capacity to recognize and thwart online threats in actual digital security scenarios? In comparison to other frameworks, how effective is the MLTIF at identifying and reducing cyberthreats? Can it be applied in an actual cybersecurity setting?

1.1. Contributions

The main contributions of this article are:

- Propose a multi-layered threat intelligence framework to improve cybersecurity posture. Our proposed framework employs various feature selection algorithms, including filtering methods (information gain, correlation coefficient, Chi-square, Fisher score) and feature selection techniques (forward wrapper, backward wrapper, Ridge classifier). Additionally, it evaluates several supervised machine learning algorithms, such as XGBoost, Gradient Boosting, LightGBM, Extra Trees, Random Forest, Decision Tree, K-Nearest Neighbor, Gaussian Naive Bayes, Support Vector Machine, Linear Discriminant Analysis, Logistic Regression, Ridge Classifier, AdaBoost, and Quadratic Discriminant Analysis.
- Provide in-depth information on effective threat intelligence integration by complementing various data sources including corporate network logs, open-source intelligence, and dark web monitoring, to offer a comprehensive overview of the cybersecurity threat landscape.
- Demonstrate the real-world effectiveness and practical applicability of the MLTIF in cybersecurity environments. Our system achieved the highest performance with an accuracy of 99.98%, precision of 99.97%, and recall of 99.96%, significantly surpassing traditional methods. Our findings underscore the effectiveness of our architecture in significantly improving an organization's capability to identify and counteract online threats in real-time.

1.2. Organization of the paper

The rest of the paper is organized as follows: Section 2 provides an overview of cybersecurity in the current digital landscape, highlighting the need for multi-layered threat intelligence frameworks, and reviews existing literature on these frameworks, discussing their strengths and limitations. Section 3 details the MLTIF architecture, including data collection, processing, threat detection, and mitigation techniques. Section 4 presents the evaluation metrics and discusses the results achieved using various feature selections of our proposed MLTIF framework. Section 5 evaluates the performance in terms of accuracy, effectiveness, and the impact of different threat intelligence, and compares our proposed framework with existing ones. Finally, Section 6 emphasizes the significance of the MLTIF in advancing cybersecurity in today's digital environment.

2. Related work

The articles in this subsection cover a range of topics related to cybersecurity and emerging technologies. Nisioti et al. (2020) [26] discuss the challenges of forensic investigations in the context of cyber-attacks and propose a data-driven decision support framework called DISCLOSE. The framework uses a repository of known adversarial tactics, techniques, and procedures (TTPs) to optimize the selection of the next steps in the investigation process. The authors demonstrate the feasibility of their approach in a case study that includes data from interviews with cybersecurity professionals and repositories of threat intelligence information.

Nwakanma et al. (2020) [27] explore the security risks related to merging the Internet of Things (IoT) with intelligent transportation systems (ITS), forming the Internet of Vehicles (IoV). They identify five security risk domains in IoV and highlight the potential of explainable AI (XAI) models to mitigate intrusion threats on intelligent connected vehicles (ICVs). The paper thoroughly reviews XAI models used in ICV intrusion detection systems and outlines future research challenges in this area.

Macas et al. (2020) [20] examine the limitations of using signature-based methods for cybersecurity in identifying new attack patterns and explore the potential of artificial intelligence (AI) to enhance the detection of attacks and address intricate cybersecurity issues. The authors emphasize the role of deep learning, across various cybersecurity-related tasks. This paper offers a critical and comparative review of the latest solutions from existing literature and highlights current challenges and future research directions.

Marinho and Holanda (2021) [28] focus on discovering new threats as early as possible given the decreasing time window between uncovering a new cyber vulnerability and its exploitation by malicious actors. To address the scale of data, at which Twitter messages can be used as a source for events and MITRE ATT&CK serves as a knowledge basis for threat characterization, the authors introduce an automated discovery framework to identify emergent threats between two societies using an unsupervised foundation. The paper does not only describe the framework in detail, but also addresses how each part (the identification of cyber-attacks and their names; profiling the identified threat with respect to what it is trying to do or achieve) can be realized.

These papers highlight the importance of data-driven decision support frameworks, explainable machine learning models, and emerging technologies such as deep learning and social media analysis in addressing the challenges of cybersecurity and emerging threats. They provide valuable insights and good practices for

Table 1
Most important threat intelligence frameworks.

Framework	Principal ideas and conclusions
DISCLOSE [26]	Utilizes an adversarial tactics, methods, and procedures (TTPs)-based data-driven decision support framework to optimize forensic investigations.
IoV Security [27]	Covers security flaws in the Internet of Vehicles (IoV) and describes explainable AI (XAI) models for intrusion detection.
AI in Cybersecurity [20]	Explains the drawbacks of signature-based methods and looks at deep learning possibilities for challenging cybersecurity jobs.
Emerging Threats [28]	Outlines a methodology for the automatic detection and profiling of new threats through the use of social media and MITRE ATT&CK information.

2.1. Threat intelligence frameworks

Threat Intelligence Frameworks (TIFs) are crucial components in cybersecurity landscape. They provide a structured approach to identify, collect, and analyze information about potential threats to an organization's cybersecurity infrastructure [29,30]. TIFs help organizations to proactively understand and mitigate the risks associated with cyber-attacks, and allow proactive security risk management [31], visual analytics [32] and decision-making [33].

One of the significant advancements in TIFs is the use of darknet and deepnet mining for proactive cybersecurity threat intelligence, as discussed by Nunes et al. [34]. The authors developed an operational system that collects information from various social platforms on the Internet, particularly sites on the darknet and deepnet. The system focuses on gathering information from hacker forum discussions and marketplaces offering products and services related to malicious hacking. The system uses data mining and machine learning techniques to identify emerging cyber-attacks, providing a significant service to cyber-defenders.

Riesco et al. [35] propose the use of blockchain to enable cybersecurity threat intelligence knowledge exchange in TIFs. The authors suggest a paradigm shift in cybersecurity information exchange by proposing an incentive-compatible strategy to dynamically mandate all parties to participate and share relevant data. They suggest an Ethereum Blockchain Smart Contract Marketplace to facilitate and incentivize the exchange of knowledge among all stakeholders.

With the advancement of TIFs which poses new challenges. For instance, Ranade et al. [35] discuss the potential threat of fabricated Cyber Threat Intelligence (CTI) generated through transformer-based models. Adversaries can use fake CTI to perform data poisoning attacks on cyber-defense systems, forcing security models to learn incorrect inputs to serve the attackers' malicious needs.

To address these risks, Riesco and Villagr'a [36] suggest adopting a dynamic risk framework with the help of CTI. They propose an upper-layer OWL-based combined architecture layout with a semantic reasoner and the use of SWRL to apply it at different tier levels (operational, tactical, strategic) for managing a Dynamic Risk Assessment and Management (DRA/DRM) framework.

TIFs play a vital role in enhancing an organization's cybersecurity posture. While advancements in TIFs bring new opportunities, they also present new challenges that need to be addressed. Therefore, continuous research and development in TIFs are necessary to keep pace with the evolving cybersecurity landscape. Table 1 shows the most important threat intelligence frameworks.

2.2. Threat intelligence sources

Threat Intelligence sources are the origins from which information about potential or existing threats to an organization's cybersecurity infrastructure is gathered [4]. These sources provide the raw data that is processed and analyzed to form actionable threat intelligence [22,37]. There are several types of threat intelligence sources, each with its unique strengths and weaknesses:

- Open Source Intelligence (OSINT) refers to data collected from publicly available sources. This could include blogs, forums, and social media platforms where hackers might discuss new tactics or vulnerabilities. It also includes reports and bulletins from security companies, industry associations, and government agencies. OSINT is a cost-effective way to gather threat intelligence, but it requires substantial processing and analysis to separate valuable information from noise [38].
- Commercial Threat Intelligence are feeds or reports from cybersecurity companies that specialize in tracking and analyzing threats. These companies have the resources to gather, analyze, and categorize threat data on a large scale. They often provide intelligence feeds tailored to specific industries or types of threats. While this source can provide high-quality, actionable intelligence, it may come at a significant cost [39].
- Industry Sharing Groups and Alliances have been established by various industries to facilitate the exchange of information regarding threats and best practices for addressing them. Notable examples include the Information Sharing and Analysis Centers (ISACs) and the Information Sharing and Analysis Organizations (ISAOs). These groups serve as mechanisms for organizations to share threat intelligence within a trusted and secure environment.
- Government agencies often gather and distribute threat intelligence to help protect national infrastructure. This information can be particularly valuable for organizations in critical industries like energy, finance, or healthcare.
- Internal Threat Intelligence refers to data gathered from an organization's own network and systems. This can include logs, event data, and other information collected by security tools. Internal threat intelligence is crucial for identifying attacks or threats specific to an organization [40].
- Basically, the term dark web intelligence refers to a part of the internet that is purposely concealed and cannot be accessed through standard web browsers. Most of the time, this is where cybercriminals trade in stolen data, hacking tools, and tactics [41]. Gathering intelligence from the dark web can provide early warnings about new threats and targeted attacks.
- Human Intelligence (HUMINT) involves gathering information from human sources. For example, a cybersecurity analyst might have a conversation with a contact at a cybersecurity conference to learn about new threats or mitigation techniques [38].

Each of these sources provides a different perspective on the threat landscape. A robust threat intelligence program will typically use a combination of these sources to gather a comprehensive view of potential threats. Table 2 summarizes the key threat intelligence sources. Table 2 shows the most important threat intelligence sources.

2.3. Threat detection and mitigation techniques

The papers in this subsection cover a range of topics related to cybersecurity and emerging technologies. Wazid et al. [42] proposes an automated system for detecting several types of cyber-attacks with a 99.92% accuracy rate in multiclass classification. The authors utilize a dataset of network traffic features and implement various machine-learning algorithms to classify the traffic into different attack categories. The paper offers a comprehensive description of the dataset, the feature selection process, and the evaluation metrics employed in the experiments.

Yang et al. (2020) [43] discusses the usage of artificial intelligence and machine learning methods for detecting and identifying cyber-attacks by analyzing log data. The authors implement the ELK Stack network log system to visually analyze log data and evaluate

Table 2
Most important threat intelligence sources.

Source	Principal ideas and conclusions
OSINT [38]	Provides publicly available data for threat detection but requires significant processing for actionable intelligence.
Commercial TI [39]	Offers high-quality, tailored threat feeds but at a cost, suitable for specific industries or threat types.
Government Agencies	Distributes valuable threat intelligence for critical infrastructure protection, essential in sectors like energy and finance.
Internal TI [40]	Uses organization-specific data for targeted threat detection, crucial for identifying internal threats.
Dark Web Intelligence [41]	Provides early warnings of emerging threats and targeted attacks from the hidden part of the internet.
Human Intelligence (HUMINT) [38]	Obtains threat intelligence through interviews; useful for context and qualitative insights.

Table 3
Most important threat detection and mitigation techniques.

Technique	Principal ideas and conclusions
Machine Learning [42]	Employing network traffic features, multiclass cyberattack categorization is achieved with great accuracy.
Log Analysis [43]	Applies AI models (XGBoost, RNN, DNN) to analyze log data for efficient cyber-attack detection and identification.
Active Defense [44]	Offers the use of natural language processing in an automated system called CTI View to provide active defense against advanced persistent threats (APTs).
IoT Security [2]	Creates a strong security architecture that integrates fog computing and IoT with healthcare systems (Healthcare 5.0).

the performance of three machine learning models: extreme gradient enhancement (XGBoost), recurrent neural network (RNN), and deep neural network (DNN). The paper provides a comprehensive evaluation of the models and identifies the XGBoost model as the most accurate for potential threats. Zhou et al. (2020) [44] combine the strategic defense idea of "active defense, traceability, and countermeasures" with the increasing severity of network security in the context of advanced persistent threats (APTs). The authors propose a novel automation system called CTI View, which uses natural language processing (NLP) techniques to process cyberspace threat intelligence (CTI) and extract useful information and features for further analysis. The paper provides a detailed description of the system architecture, data processing pipeline, and evaluation metrics used in the experiments. De Souza et al. (2020) [2] address the security issues related to the integration of the Internet of Things (IoT) and fog computing in healthcare 5.0 systems. They introduce a secure general healthcare 5.0 framework that enables various security-related processes such as authentication, access control, key management, and intrusion detection. The paper reviews different security requirements and the threat model of healthcare 5.0, examining existing security mechanisms and comparing their performance. The authors also highlight future research directions for those working in the Healthcare 5.0 field.

These papers bring out the significance of machine learning and artificial intelligence on cyber-attack detection and prediction along with secure frameworks for automation systems to mitigate security challenges in IoT, fog computing like emerging technologies [45]. They offer useful information and best practice advice for both researchers and practitioners. Table 3 shows the Most important threat detection and mitigation techniques.

3. Methodology

3.1. Dataset

The data we study in this article comes from the LITNET-2020 dataset [46]. The dataset contains benign and 12 malicious attack network traffic types in a national wide-area-network with 85 flow features from the dataset examples, where there are samples for only positive class and infiltrations (1% or less labeled malicious in the traffic space), ordered by decreasing absolute frequency among them:

1. Smurf attack involves the continuous transmission of ICMP broadcast requests to a network, ostensibly originating from a target node. The primary objective is to inundate the node with excessive network traffic, thereby impeding its performance.
2. ICMP Flood attack is a form of Denial of Service (DoS) assault that seeks to inundate a specific network node with an excessive volume of ICMP echo requests, commonly referred to as "pings".
3. UDP-Flood attack leverages the User Datagram Protocol (UDP). A specialized variant, known as the DNS Flood Attack or DNS Flooding, is characterized by the transmission of network packets to arbitrary IP addresses, specifically targeting UDP protocol on port 53.
4. Utilizes a vulnerability in the initial part of the basic TCP three-way handshake. The objective of this attack is to deplete the victim node's resources, rendering it unresponsive.
5. HTTP-Flood Attack: This attack involves seemingly legitimate HTTP GET or POST requests directed at a web server, which processes and responds to these requests. Unlike Layer 3/4 DDoS floods, these attacks do not rely on malformed packets, spoofing, or reflection of valid requests. They require only a fraction of the bandwidth to incapacitate the server or site by using malicious packets that avoid port 80 due to latency reasons.
6. LAND Attack: This attack is executed by a malicious node that sends a TCP segment with identical source entry and port number details. The processing of these packets within the TCP stack overwhelms the victim node, causing it to become unresponsive. These packets are malicious and have the 'S' flags using the TCP protocol.
7. W32.Blaster Worm: This worm spreads by exploiting a Buffer Overrun Vulnerability in the Microsoft Windows DCOM RPC Interface.
8. Code Red Worm: This worm attempts to cause a buffer overflow on the target node, resulting in the accidental modification of nearby memory. The malicious packets are sent to the source IP, targeting port 80, and do not use Secure Sockets Layer (SSL), but rather leverage the HTTP GET method.
9. Spam Bot Attack: This attack involves the distribution of spam messages or posting spam on social platforms and forums. The malicious traffic is directed solely to port 25 in plain text (no SSL). A common characteristic of this type of attack is the establishment of a large number of SMTP connections from a specific address simultaneously.
10. Reaper Worm: This worm begins scanning after the IP is passed to the exploit process. The attack focuses on TCP ports 81 to 10,000. A log entry is generated only if the packet contains a TCP stream and not UDP, ICMP, or ICMP6.
11. Port Scanning/Spread Attack: This attack sends client requests to various server port addresses, identifies active ports, and exploits known security vulnerabilities. The anomaly is detected when there are numerous connections from one host to different ports or the same port among multiple hosts.
12. Packet Fragmentation Attack: Attackers flood a network by manipulating the datagram fragmentation process.

For a comprehensive understanding of the detailed features, we direct the reader to the paper by Damasevicius et al. [46]. This paper provides an in-depth analysis and thorough explanation of the features. The evaluation of the MLTIF architecture was conducted using the LITNET-2020 dataset, which includes benign and malicious network traffic types. We tested several machine learning classifiers to determine their effectiveness in threat detection and mitigation.

1. Performance without Feature Selection: Among the classifiers, XGBoost achieved the highest performance with accuracy, precision, recall, and F1 score of 0.9998. Gradient Boosting and LightGBM also performed exceptionally well, achieving scores of 0.9997. These results indicate that ensemble methods are highly effective in handling the dataset without feature selection.
2. Performance with Feature Selection: Using the Information Gain/Correlation Coefficient and Chi-Square/Fisher Score methods, we observed slight improvements in classifier performance. Gradient Boosting and LightGBM maintained high accuracy and balanced precision–recall metrics, underscoring their robustness in threat detection.
3. Filter and Wrapper Methods: The application of filter methods (Information Gain, Correlation Coefficient) and wrapper methods (Forward and Backward Wrapper) further refined the model performance. XGBoost and Gradient Boosting continued to show superior results, confirming their effectiveness in a multi-layered threat intelligence framework.

3.2. Multi-layered threat intelligence framework architecture

A multi-layered Threat Intelligence Framework (MLTIF) architecture is a complex system designed to provide organizations with a comprehensive and coordinated approach to threat detection and mitigation. It is based on the concept of integrating multiple layers of threat intelligence sources, including internal and external sources, to create a more complete and accurate picture of the current threat landscape. The architecture (Fig. 1) is composed of several layers, each with a specific function and purpose.

The first layer is the data collection and processing layer of MLTIF architecture. The aggregation tier is responsible for gathering raw data from many sources (network devices, security sensors, and logs) to filter relevant patterns and get rid of irrelevant data as discussed in [47]. This data is then collected and analyzed after applying various aggregation, filtering, and correlation methods to summarize any security threats or incidents.

The threat intelligence analysis layer of the MLTIF architecture is its second layer. This layer is responsible for performing deep processing of the data and extracting meaningful threat intelligence insights from raw relevant findings. The data can be used to highlight similar patterns, correlations, and risks that could threaten the security posture of an organization. Machine learning, artificial intelligence, and human expertise identify threats on the analysis layer.

Threat detection and mitigation layer: This is the third tier of MLTIF architecture. This is the layer that will handle the near real-time detection response to pending or ongoing security incidents and threats. This combines intrusion detection, firewalls, and a whole host of measures to block or quarantine threats before they reach critical infrastructure.

The fourth layer of the MLTIF architecture is a reporting and feedback bottleneck. It is understood to give real-time feedback to the other layers. Creates reports, alerts, and notifications for security teams to alert them of possible threats following actual incidents. Throughout, this layer even produces feedback that can be used to enhance the efficiency of other layers in its architecture.

The final layer in the MLTIF architecture which is the management and governance layer. This layer is in charge of watching over or keeping control of all MLTIF operations. The layer contains the appropriate policies, procedures, and standards overarching strategies for dealing with managing services as well as implications to security-based training programs that educate individuals from both enable stakeholders like IT operators or Users. It also contains the mechanisms for monitoring and auditing to confirm the effectiveness and compliance of an architecture.

The MLTIF architecture is a complex and comprehensive system that combines multiple layers of threat intelligence to provide organizations with a holistic approach to threat detection and mitigation. The architecture is designed to be flexible, scalable, and adaptable to the ever-evolving threat landscape, making it an effective tool for protecting organizations from advanced and sophisticated cyber-attacks.

3.3. Data collection and processing

As shown in Fig. 1 and discussed below, the process of data collection and processing in Multi-layered Threat Intelligence Framework (MLTIF) architecture consists as follows:

1. In the MLTIF architecture, data is collected from different sources, including network devices, security appliances, endpoints, and threat intelligence feeds. This data is collected in real-time and in batch mode.
2. After data is collected from various sources, it needs to be normalized and enriched. This step involves converting the data into a standard format, so it can be processed and analyzed easily. The data is enriched with additional information from external sources, such as threat intelligence feeds, to provide context and improve the accuracy of threat detection.
3. After normalization and enrichment, the data is stored in a central repository for further processing and analysis. The storage solution should be scalable, secure, and flexible to accommodate different types of data.
4. The MLTIF architecture employs a number of data processing and analysis methods to recursively detect structures in unlabeled Indicators of Compromise (IOCs), such as machine learning, statistical analysis, or rule-based systems. It is an essential step since you need to find out if any threats exist and logs for those alerts should be created, so they can be investigated.
5. The threat intelligence correlation module correlates the enriched data with external threat intelligence feeds, to identify potential threats and provide context to security analysts. This step involves matching indicators of compromise (IOCs) from external feeds with internal data to detect potential threats.
6. After potential threats are detected, the MLTIF architecture generates alerts, which can be sent to security analysts for further investigation. The alerts should include detailed information about the potential threat, including the source, type, and severity.
7. The final step in the data collection and processing phase involves responding to and mitigating potential threats. The MLTIF architecture provides security analysts with actionable intelligence and response options, such as blocking IP addresses, isolating infected endpoints, and applying patches and updates to vulnerable systems.

3.4. Threat detection and mitigation techniques

The Multi-Layered Threat Intelligence Framework (MLTIF) architecture incorporates various threat detection and mitigation techniques to effectively counter potential cybersecurity threats. These techniques can be broadly classified into four layers: Network, Host, Application, and User.

At the network layer, MLTIF architecture uses various threat detection and mitigation techniques, including network traffic analysis, intrusion detection and prevention systems (IDPS), and firewalls. Network traffic analysis is performed by analyzing network traffic to identify any unusual or suspicious behavior. Intrusion Detection and Prevention System (IDPS) used to detect/prevent attacks that are happening over the network level such as DoS, port scanning etc. Firewalls are security devices that monitor and control all incoming and outgoing network traffic based on a set of predetermined security policies.

At the host layer, MLTIF architecture uses various techniques to secure endpoints and hosts, including host-based intrusion detection and prevention systems (HIDPS), endpoint protection platforms (EPP),

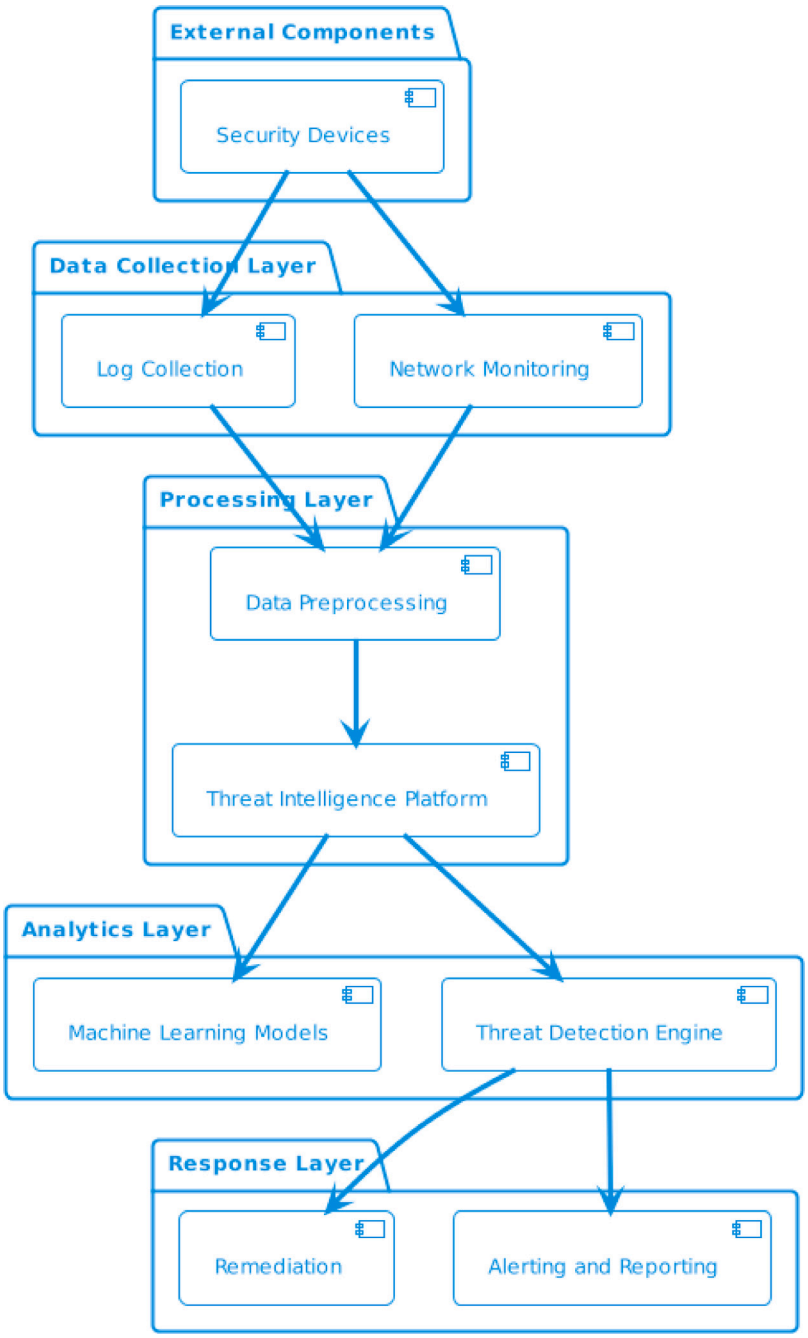

Fig. 1. The MLTIF architecture.

and antivirus software. HIDPS is used to detect and prevent attacks on specific hosts, such as buffer overflow attacks, rootkits, and malware. EPP is used to protect endpoints from various types of malware, including ransomware, viruses, and trojans. Antivirus software is used to detect and remove known malware from the system.

At the application layer, MLTIF architecture uses various techniques to secure applications, including application security testing, web application firewalls (WAF), and secure coding practices. Application security testing is used to identify vulnerabilities and weaknesses in the application code. WAF is used to monitor and filter incoming and outgoing traffic to and from the application. Secure coding practices are used to prevent common coding errors that can lead to security vulnerabilities.

At the user layer, MLTIF architecture uses various techniques to educate and train users to follow safe cybersecurity practices. This includes security awareness training, access control, and identity and access management (IAM). Security awareness training is used to educate users about the importance of cybersecurity and how to recognize potential threats. Access control is used to limit access to sensitive information and systems to authorized users only. IAM is used to manage user identities and access rights to various systems and applications.

The MLTIF architecture uses a multi-layered approach to effectively detect and mitigate potential cybersecurity threats. By incorporating various threat detection and mitigation techniques across different layers, MLTIF architecture provides comprehensive protection against various types of cybersecurity threats.

3.5. Machine learning methods

The MLTIF architecture uses several machine learning methods for cyber attack recognition. XGBoost (eXtreme Gradient Boosting) stands for XLS pronounced as "Ensemble". LightGBM is a gradient boosting framework, which provides many benefits: extremely fast and scalable. It is based on gradient boosting and has built-in tree pruning, regularization, and missing value handling represented quite well. It is widely used by data scientists due to its excellent support for handling high-dimensional data and parallel processing.

Gradient Boosting (GBM) is an iterative ensemble method that builds a series of decision trees and composes them to improve performance. It aims to reduce the errors from past iterations by updating the weights of misclassified points. This cycle of correcting the model further becomes stronger and more predictive over time. LightGBM is another gradient-boosting framework that operates faster than the regular tree-based algorithms and logically follows a leaf-wise approach over level-wise growth. This makes it fast and efficient, particularly for large datasets. Also, it supports GPU learning and categorical features right off the bat without performing one-hot encoding of those features.

Extra Trees is short for Extremely Randomized Trees. It is another ensemble method like Random Forest, but with one crucial difference is that instead of looking for the most discriminative thresholds, it takes place at random, which results in more diversity and potentially a better model. A Random Forest is an ensemble learning method used in classification, and regression, as well as for getting insights into feature importances. Random forest is a popular model that strikes a good balance between accuracy and interpretability, and it helps overcome the problem of overfitting.

K-Nearest Neighbor (KNN) - instance-based learning for a new data point, it looks up the training set for 'k' number of examples closest to the point and returns that value which is most common among its k-nearest neighbors. Gaussian Naive Bayes is a probabilistic classifier based on the assumption that features are independent. The algorithm is especially useful for large high-dimensional datasets and has gained popularity due to its simplicity and speed.

Support Vector Machine (SVM) is a supervised machine learning algorithm that finds the hyperplane that maximally separates classes in your data. It works well in high-dimensional spaces and is effective in situations where the boundary between classes is non-linear. Linear Discriminant Analysis (LDA) aims to find a linear combination of features making the perfect decision boundary separating two or more different classes in your dataset. It is very effective for dimensionality reduction.

The Logistic Regression, despite its name, is a much different beast from the models listed here as it does regression in nature but classifies either true or false between two possibilities. It calculates the probability of an instance being in a certain class. The Ridge Classifier is a linear classifier that uses an L2 penalization. It imposes a cost on different magnitudes of the weights in the model to avoid overfitting. AdaBoost (Adaptive Boosting) is an ensemble method that involves training the model on misclassified data points and decreasing their weights during each iteration, thereby letting the model learn from those difficult cases, highly increasing its focus on hard-to-identify classes.

Finally, Quadratic Discriminant Analysis (QDA) is very similar to LDA but allows each class its covariance matrix and thus non-linear decision boundaries. This is a more general method, but the separate covariances would need to be estimated with additional data.

3.6. Filter methods

3.6.1. Information gain/ correlation coefficient

Information gain is a feature selection metric used in decision tree-based algorithms to determine the relevance of a feature for classification [48–50]. It measures the amount of information obtained about a class variable when a particular function is used to partition the data. The steps involved in calculating the information gain for feature selection are as follows:

- Calculate the entropy of the parent node:

$$Entropy(parent) = -\sum(P(class) \times log_2(P(class))) \quad (1)$$

  This calculates the uncertainty or impurity in the parent node before any split is performed. $P(class)$ represents the probability of each class label in the parent node.
- For each possible element value, calculate the weighted average entropy of the resulting child nodes after the split:

$$Weighted\,average\,Entropy(children) = \sum \frac{count(child)}{count(parent)} \times Entropy(child) \quad (2)$$

  This calculates the entropy of the child nodes resulting from the distribution of the data based on each element value. $count(child)$ represents the number of instances in the child node and $count(parent)$ represents the total number of instances in the parent node.
- Calculate the information gain for the element:

$$Information Gain(Function) = Entropy(Parent) - Weighted\,Average\,Entropy(Children) \quad (3)$$

  Information gain represents the reduction in entropy achieved by partitioning data based on a feature. A higher information gain indicates that the feature is more informative and provides more valuable information for classification.

A correlation coefficient is a statistical measure that calculates the strength and direction of an unvarying relationship between two variables. In practice, this is often used in feature selection to find features that are strongly correlated with the target variable. The correlation coefficient between two variables X and Y can be calculated using the following formula:

$$Correlation\_coefficient(X,Y) = (\sum((X-\overline{X})\times(Y-\overline{Y})))/(n\times\sigma(X)\times\sigma(Y)) \quad (4)$$

Where: X and Y are the values of two variables, $\overline{X}$ and $\overline{Y}$ are the mean values of X and Y, $\sigma(X)$ and $\sigma(Y)$ are the standard deviations of X and Y, n is the total number of observations.

The steps involved in using the correlation coefficient for element selection are as follows:

1. Calculate the correlation coefficient between each item and the target variable.
   - For continuous variables, use Pearson's correlation coefficient, which measures a linear relationship.
   - For categorical variables, you can use techniques such as point biserial correlation or eta-squared.
2. Assess the robustness and direction of the correlation coefficient:
   - The correlation coefficient varies between −1 and 1.
   - A positive coefficient signifies a positive correlation, indicating that an increase in the trait corresponds with an increase in the target variable.
   - A negative coefficient signifies a negative correlation, indicating that an increase in the trait corresponds with a decrease in the target variable.
   - The absolute value of the correlation coefficient indicates the strength of the relationship, with values approaching −1 or 1 denoting a stronger correlation.
3. Identify highly correlated traits.

- Features with a correlation coefficient close to −1 or 1 are considered highly correlated with the target variable.
- These features have a strong linear relationship with the target variable and can provide valuable information for prediction.

3.6.2. Chi square/Fisher score

Chi-square (x^2) is a statistical measure used for feature selection to determine independence between categorical variables. It is particularly useful when working with categorical data and can help identify features that are significantly associated with the target variable.

The chi-square statistic for testing independence between two categorical variables can be calculated using the following formula:

$$x^2 = \sum((Observed - -Expected)^2/Expected) \tag{5}$$

Where: $Observed$ is the observed frequency of each category combination in the pivot table. $Expected$ is the expected frequency of each category combination, assuming independence.

The steps involved in using chi-square for feature selection are as follows:

1. Create a table where the rows represent the categories of one variable and the columns represent the categories of the other variable. Calculate the observed frequency by counting the occurrences of each combination of categories in the data set.
2. Assuming independence between the two variables, calculate the expected frequency of each category combination in the pivot table. The expected frequency can be calculated using the formula:

$$(total\_number\_of\_rows \times total\_number\_of\_columns)/ \\ total\_number\_of\_observations \tag{6}$$

3. For each combination of categories, calculate the contribution to the chi-square statistic using the above formula. Sum the posts for all combinations of categories to get the chi-square statistic.
4. Degree of freedom is calculated as $(Numberofrows - 1) \times (Numberofcolumns - 1)$. It represents the number of independent values that can be chosen arbitrarily in the chi-square distribution.
5. Determine the critical value from the chi-square distribution table based on the degrees of freedom and the desired significance level (e.g., 0.05). If the calculated chi-square statistic exceeds the critical value, it signifies a significant association between the variables.

The Fisher score, also referred to as Fisher's discriminant analysis, is a statistical measure employed for feature selection in classification tasks. Its objective is to identify features that possess high discriminative power, thereby effectively distinguishing between different classes within a dataset. The Fisher score for an object can be calculated using the following formula:

$$Fisherscore(function) = (\mu_1 - \mu_2)^2/(\sigma_1^2 + \sigma_2^2) \tag{7}$$

Where: μ_1 and μ_2 are function means for two different classes. σ_1^2 and σ_2^2 are the deviations of the function for these two classes.

The steps involved in using the Fisher score for feature selection are as follows:

1. Calculate the mean (μ) and variance (σ^2) of each feature separately for each class in the data set.
2. Calculate the difference between element means for different classes and square the result. Multiply the square of the difference by the number of samples in each class and sum the values.
3. Calculate the element variance within each class separately. Multiply the variance by the number of samples in each class and sum the values.
4. Divide the between-class variance matrix (Sb) by the within-class variance matrix (Sw). The resulting value represents the Fisher score for each feature, indicating its discriminating power.
5. Rank the features based on their Fisher scores. Features with a higher Fisher score are considered more relevant and important for classification.

3.7. Feature selection methods

3.7.1. Forward wrapper

Forward wrapper is a feature selection technique that involves repeatedly adding features to a model and evaluating their performance. It starts with an empty set of features and gradually adds the most promising features one by one until a stopping criterion is met. The goal is to find the optimal subset of features that maximizes model performance.

The steps involved in selecting a forward wrapper function are as follows:

1. Initialize an empty set of selected functions.
2. Cycle through each feature not yet selected:
   - Add one function at a time to the selected function set.
   - Train a model using selected features and evaluate its performance using a selected metric (eg accuracy, F1 score).
3. Select the function that provides the best performance:
   - Select the feature that will most improve model performance when added to the selected feature set.
4. Repeat steps 2 and 3 until you meet the stopping criterion:
   - The stopping criterion can be a predetermined number of features to be selected or a threshold to improve performance.
5. Evaluate the final selected feature set:
   - Train the model using the final set of selected features.
   - Assess the performance of the model using the chosen evaluation metric.

The forward wrapper approach explores different combinations of features by gradually adding the most promising ones. Its goal is to find a subset of features that optimize model performance. However, this can be computationally expensive, especially for feature-rich datasets.

3.7.2. Backward wrapper

A backward wrapper is a feature selection technique that repeatedly removes features from the model and evaluates their impact on performance. It starts with the full set of features and gradually removes the least promising features one by one until a stopping criterion is met. The goal is to find the optimal subset of features that maximizes model performance.

The steps involved in selecting the return wrapper function are as follows:

1. Initialize the function set with all available functions.
2. Train the model using the full set of features and evaluate its performance using a chosen metric (eg accuracy, F1 score).
3. Go through each function in the function set: Temporarily remove one element at a time from the feature set. Train the model using the remaining features and evaluate its performance.
4. Select the element whose removal has the least negative impact on model performance.
5. Remove the selected feature from the feature set.
6. Repeat steps 3–5 until you meet the stopping criterion. The stopping criterion can be a predetermined number of features to remove or a performance degradation threshold.

7. Evaluate the final selected feature set. Train the model using the final set of selected features. Assess the performance of the model using the chosen evaluation metric.

The back-wrapping approach systematically evaluates the impact of removing each feature on model performance. Its goal is to identify a subset of features that optimize model performance by removing the least informative features. However, like the forward wrapper, it can be computationally expensive, especially for feature-rich datasets.

3.7.3. Ridge (insertion)

Ridge algorithm is a data mining algorithm that combines the concepts of ridge regression and binary classification. It extends the traditional linear classifier by introducing a regularization term that helps mitigate the problem of multicollinearity and overfitting.

The steps involved in Ridge Classifier training are as follows:

1. Data preparation:
   - If necessary, preprocess the data by scaling or normalizing features.
   - Split the dataset into training and test sets to evaluate the performance of the model.
2. Initialize the Ridge Classifier:
   - Set the regularization parameter (alpha) to control the amount of regularization.
   - Select an appropriate solution for the optimization problem (eg 'auto', 'svd', 'cholesky', 'lsqr' or 'sag').
3. Train the Ridge classifier:
   - Fit the model to the training data using the fit() function or similar method.
   - Ridge Classifier learns the relationship between features and the target variable using ridge regression.
4. Make predictions:
   - Use a trained Ridge Classifier to predict test data or new unseen data.
   - The model assigns class labels based on the learned decision boundary.
5. Evaluate the performance of the model:
   - Calculate evaluation metrics such as accuracy, precision, recall and F1 scores to assess model performance.
   - Compare the predicted class labels with the actual class labels from the test data.

4. Evaluation metrics and results

4.1. Evaluation metrics

In machine learning and statistical classification, assessing model performance is essential. Common metrics used for evaluation include accuracy, precision, recall, and the F-measure. Each metric provides unique insights into the classifier's performance.

Accuracy is a fundamental metric that denotes the proportion of correctly predicted instances out of the total instances. Mathematically, it is expressed as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

where:

- TP (True Positives) represents correctly predicted positive instances,
- TN (True Negatives) represents correctly predicted negative instances,
- FP (False Positives) represents incorrectly predicted positive instances,
- FN (False Negatives) represents incorrectly predicted negative instances.

Precision, or positive predictive value, measures the accuracy of positive predictions by evaluating the proportion of true positive instances among all instances predicted as positive. It is calculated using the formula:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

Recall, also known as sensitivity or true positive rate, evaluates the model's ability to correctly identify all relevant positive instances. It is defined as the proportion of true positive instances among all actual positive instances:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

The F-measure, or F1 score, is the harmonic mean of precision and recall, providing a single metric that balances both aspects of the model's performance. This metric is particularly useful in cases of imbalanced class distributions. The F-measure is given by:

$$\text{F-Measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

Each of these metrics offers a distinct perspective on classifier performance. Accuracy gives a general performance overview, precision is critical when the cost of false positives is high, recall is vital when it is important not to miss positive instances, and the F-measure is beneficial for balancing precision and recall.

4.2. Results with no feature selection

These features or characteristics, namely 'td', 'sp', 'dp', 'stos', 'ipkt', 'ibyt', '_in', 'out', 'sas', 'das', play an essential part in their applications and enable comprehensive insights and informed decision-making. Fig. 2 shows the performance metrics for each classifier when no feature selection technique was used. Among the classifiers, XGBoost achieved the highest performance with accuracy, precision, recall and F1 score of 0.9998. It was closely followed by Gradient Boosting and LightGBM, both achieving precision, accuracy, recall and F1 scores of 0.9997. These file-based algorithms have shown remarkable performance, indicating their efficiency in handling the dataset.

Other classifiers such as Extra Trees and Random Forest also showed high performance with precision, accuracy, recall and F1 score of 0.9996. These results indicate the suitability of file-based algorithms for the task. In contrast, classifiers such as SVM, AdaBoost, and QDA performed poorly, with significantly lower accuracy and other metrics.

A confusion matrix presented in Fig. 3 consisting of labels such as BLASTER_WORM_v2_ATTACKERS_ONLY and HTTP_FLOOD_v2_ATTACKERS_ONLY provided an overview of the model's predictions for different attack categories. Most matrix entries are aligned along the diagonal, indicating accurate predictions for most classes. However, there were some errors, such as 2 false positive predictions for BLASTER_WORM_v2_ATTACKERS_ONLY and 1 false positive for HTTP_FLOOD_v2_ATTACKERS_ONLY. These findings shed light on the strengths and weaknesses of the model and offer insight for further improvement and development in the investigation of attack classification.

4.3. Filter methods

These features or characteristics, namely 'sp', 'dp', 'ibyt', '_in', 'out', 'sas', 'das', play an essential part in their applications and enable comprehensive insights and informed decision-making.

Fig. 4 shows the performance metrics for all classifiers when we use the gain/correlation filter selection matrix. Among the classifiers, Gradient Boosting achieves the highest accuracy of 0.9996, indicating

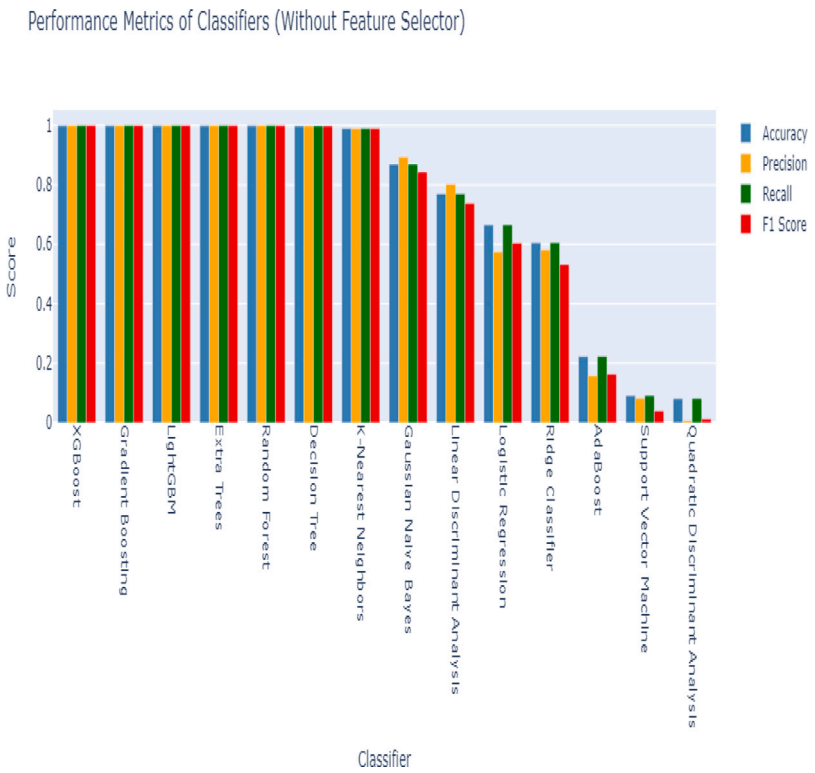

Fig. 2. Performance without feature selection.

that it has the highest overall correct prediction rate. It also demonstrates excellent precision, recall and F1 scores, all very close to 1. This indicates that Gradient Boosting has a high level of precision and can make accurate predictions while maintaining a good balance between identifying positive and negative cases. Random Forest and XGBoost also achieve high accuracy, precision, recall, and F1 scores, with values very similar and close to Gradient Boosting. This suggests that these ensemble methods are effective in this classification task and provide robust and reliable predictions.LightGBM achieves slightly lower accuracy compared to previous classifiers, but still maintains a high level of precision, recall, and F1 scores. While it may not perform as well as Gradient Boosting, Random Forest, and XGBoost in terms of accuracy, it remains a strong choice with a good trade-off between accuracy and recall.

Extra Trees, Decision Tree, and KNN achieve slightly lower accuracy and performance metrics compared to the previous classifiers, but still maintain values above 0.99 for most metrics. These models may not be as accurate as the most powerful ones, but they still provide reliable predictions with reasonably high accuracy and repeatability. As we move further down the list, the performance of the classifiers starts to drop significantly. Gaussian Naive Bayes achieves an accuracy of 0.8726, showing a significant drop compared to previous models. LDA and logistic regression show relatively low precision, accuracy, recall, and F1 scores, suggesting that they may not be the most appropriate choice for this particular data set. Ridge Classifier, AdaBoost, Support Vector Machine, and QDA show even lower performance metrics. These classifiers have relatively low accuracy, precision, recall, and F1 scores, indicating that they have difficulty making accurate predictions on this dataset. The best performers in this classification task are Gradient Boosting, Random Forest, and XGBoost, which achieve high accuracy, precision, recall, and F1 scores.

Based on the top-performing algorithm, Gradient Boosting in this case, we produce a confusion matrix as in Fig. 5. The matrix contained predictions for various attack categories. Upon examination of the matrix, it is clear that the model performed well overall, with most items aligned along the diagonal. This indicates that the model made correct predictions for most classes. However, there were some misclassifications. For example, in the 'BLASTER_WORM_v2_ATTACKERS_ONLY' class, there were 2 false positive predictions indicating cases that were mistakenly classified as belonging to this attack category. In addition, 3 cases were misclassified as "No Attack", potentially affecting the accuracy of the model in determining harmless traffic.

Fig. 6 shows the performance metrics for all classifiers when we use the Chi Square/Fisher score filter selection matrix.

LightGBM achieved the highest accuracy of 0.999, indicating that it correctly classified the vast majority of cases. The algorithm yielded precision, recall, and F1 scores all exceeding 0.999. This suggests that LightGBM performed exceptionally well in correctly identifying positive instances (precision) and capturing all positive instances (recall). A high F1 score further indicates a balanced performance between precision and recall.

Gradient Boosting, XGBoost, Random Forest, and Extra Trees achieved accuracy values above 0.998, indicating their strong performance. These algorithms also exhibited high precision, recall, and F1 score values, further highlighting their effectiveness in accurately classifying cases.

The decision tree algorithm attained a detection accuracy of 0.998 and demonstrated comparable values of precision, recall and F1 score to the above ensemble methods. This suggests that the decision tree algorithm is able to achieve a similar level of performance to more complex ensemble algorithms.

KNN attained a detection accuracy of 0.987, which is slightly lower than the file-based and decision tree algorithms. However, its precision, recall, and F1 scores were still relatively high, indicating its effectiveness in classifying cases, albeit with a slightly higher error compared to the best-performing algorithms.

Gaussian Naive Bayes moved down the table and attained a detection accuracy of 0.959, demonstrating a relatively high level of correct classification. While its accuracy, recall, and F1 scores were slightly lower than previous algorithms, Gaussian Naive Bayes still performed solidly.

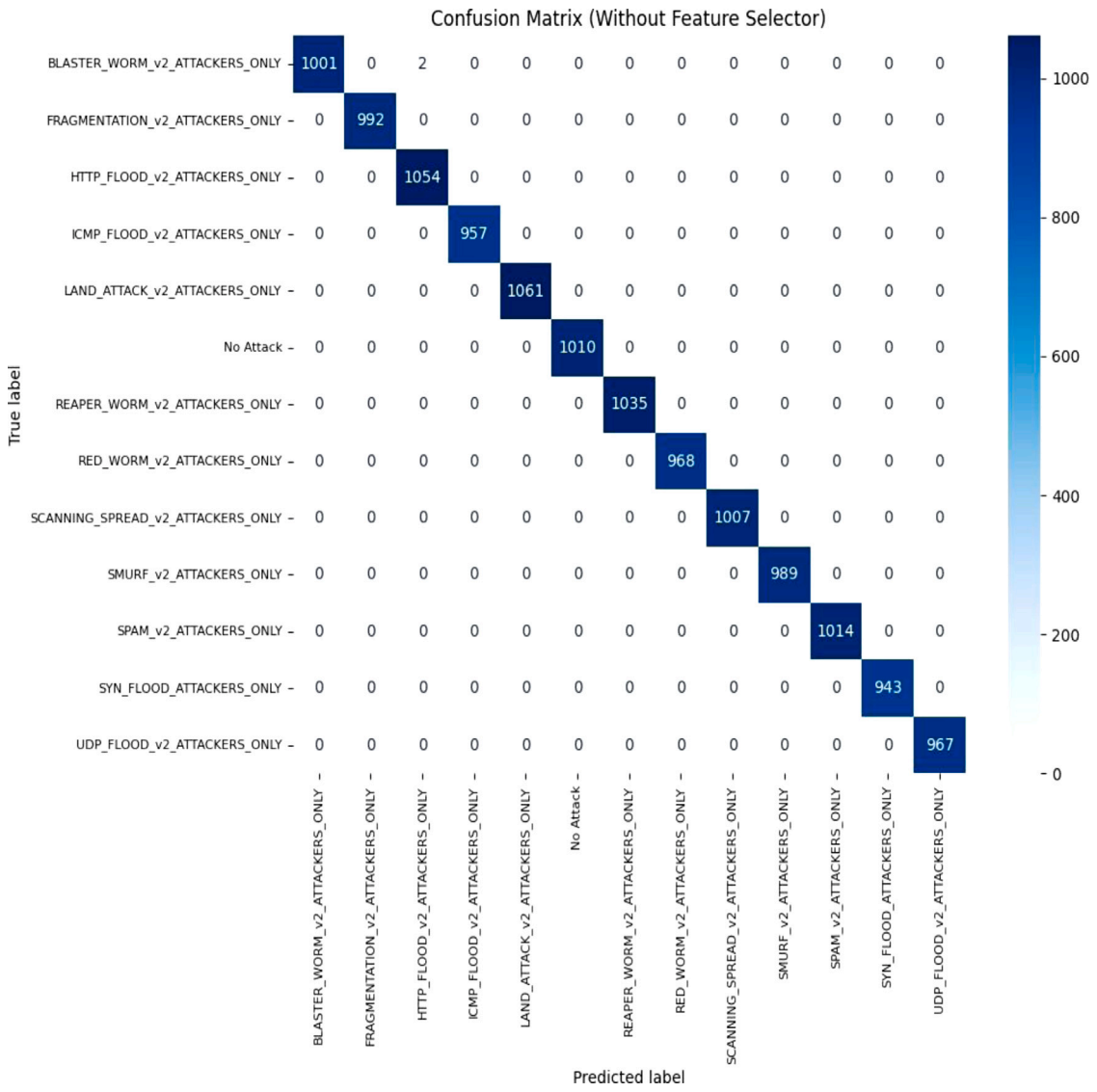

Fig. 3. Confusion matrix without feature selection.

SVM attained a detection accuracy of 0.855, indicating a lower level of correct classification compared to previous algorithms. Its precision, recall and F1 scores were also lower, indicating that the SVM struggled to accurately classify cases, especially with respect to accuracy.

LDA attained a detection accuracy of 0.767, demonstrating a further decrease in correct classifications. The precision, recall, and F1 scores were also lower than previous algorithms, indicating limitations in accurately classifying cases.

Logistic Regression and Ridge Classifier achieved a detection accuracy of 0.650 and 0.601 respectively, indicating a significant drop in correct classifications. Their precision, recall, and F1 score values were also lower than previous algorithms, indicating significant challenges in accurately classifying cases.

AdaBoost attained a detection accuracy of 0.222, which is significantly lower than the other algorithms. This indicates considerable difficulty in accurately classifying cases. The precision, recall and F1 scores were also very low, further highlighting the limitations of AdaBoost in this particular scenario.

Finally, QDA attained a detection accuracy of 0.081, which represents the lowest level of correct classification among all algorithms. Its precision, recall, and F1 scores were also extremely low, indicating poor performance in accurately classifying cases.

The results of the classification algorithms varied significantly. The ensemble architecture, comprising LightGBM, Gradient Boosting, XGBoost Model Stacking, and Random Forests, demonstrated the highest accuracy in all predictions due to markedly enhanced precision and recall compared to individual methods. Conversely, algorithms such as AdaBoost, QDA, Logistic Regression, and Ridge Classifier failed to accurately predict the classes, exhibiting only classical levels of accuracy.

Based on the top-performing algorithm, Gradient Boosting in this case, we produce a confusion matrix in Fig. 7. Examining the matrix, we observed that the model made accurate predictions for most cases, as indicated by the entries along the diagonal. However, several misclassifications should be noted.

For example, in the "BLASTER_WORM_v2_ATTACKERS_ONLY" class, there were 2 false positive predictions where cases were incorrectly classified as belonging to this attack category. In addition, 7 cases were misclassified as "SPAM_v2_ATTACKERS_ONLY". These misclassifications could affect the overall accuracy of the model. There was one false positive prediction in the 'No Attack' class, indicating that the instance was incorrectly classified as an attack. On the other hand, there were 7 false negative predictions in this class, indicating cases that were actually attacks but were classified as non-attacks. These misclassifications highlight the potential for improving the model's

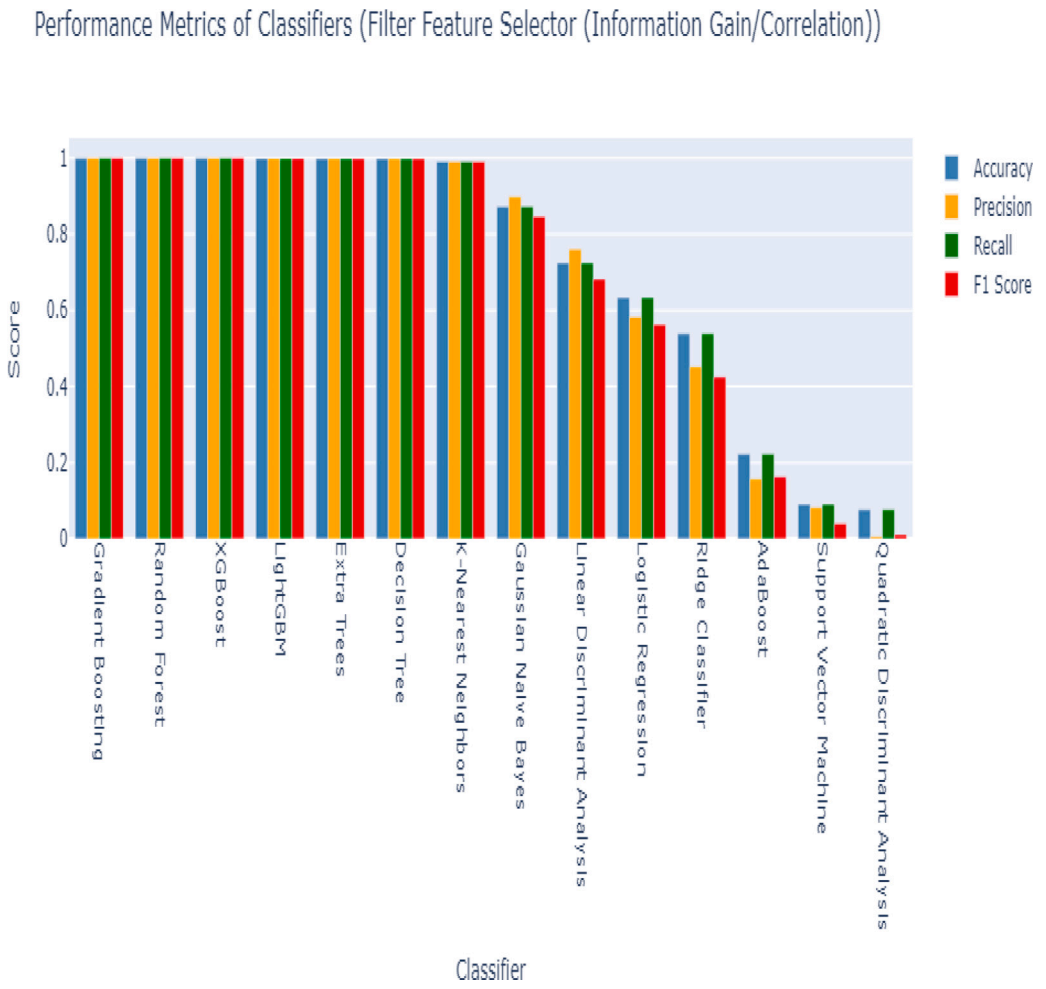

Fig. 4. Performance of classifiers when Information Gain/ Correlation Coefficient used for feature selection.

ability to identify benign traffic accurately. Overall, while the model performed well overall, there are areas where it could be improved to reduce both false positives and false negatives alarm rate, especially in the 'BLASTER_WORM_v2_ATTACKERS_ONLY' and 'No Attack' classes.

4.4. Wrapper methods

Fig. 8 shows the performance metrics for all classifiers when we use the Forward function envelope.

Among the tested algorithms, XGBoost achieved the highest accuracy score of 0.9997, indicating an extremely high level of correct classification. This algorithm also showed excellent precision, recall and F1 score values, all above 0.9996. These results indicate that XGBoost performed exceptionally well in accurately identifying positive cases (precision) and capturing all positive cases (recall), resulting in balanced F1 scores.

LightGBM, Random Forest, Extra Trees, and Gradient Boosting achieved accuracy scores above 0.9993, indicating their strong performance in correctly classifying cases. These algorithms also showed high precision, recall, and F1 score values, highlighting their effectiveness in accurately identifying positive cases and achieving a balance between precision and recall.

The decision tree algorithm achieved a detection accuracy of 0.9984, demonstrating a slightly lower but still commendable level of correct classification. Its accuracy, recall, and F1 score values were comparable to ensemble-based algorithms, suggesting that decision trees can achieve similar performance levels to more complex ensemble methods.

KNN achieved a detection accuracy of 0.9905, which is slightly lower than ensemble-based and decision tree-based algorithms. However, its precision, recall, and F1 scores were still relatively high, indicating its effectiveness in accurately classifying cases, albeit with a slightly higher error compared to the best performing algorithms.

Gaussian Naive Bayes achieved a detection accuracy of 0.8681, demonstrating a relatively high level of correct classification. Although its precision, recall, and F1 scores were slightly lower compared to previous algorithms, Gaussian Naive Bayes still showed solid performance in accurately classifying cases.

LDA achieved a detection accuracy of 0.7654, indicating a further decrease in correct classifications compared to previous algorithms. Its precision, recall, and F1 scores were also lower, indicating limitations in accurately classifying cases.

Logistic regression and Ridge Classifier achieved detection accuracy of 0.6331 and 0.4221, respectively, indicating a significant drop in correct classifications. Their precision, recall, and F1 score values were also lower than previous algorithms, indicating problems in accurately classifying cases.

AdaBoost achieved a detection accuracy of 0.2223, which is significantly lower than the other algorithms. This indicates considerable difficulty in accurately classifying cases. The precision, recall and F1 scores were also very low, further highlighting the limitations of AdaBoost in this particular scenario.

SVM and QDA achieved detection accuracy of 0.0908 and 0.0763, respectively, representing the lowest levels of correct classification among all algorithms. Their precision, recall, and F1 score values

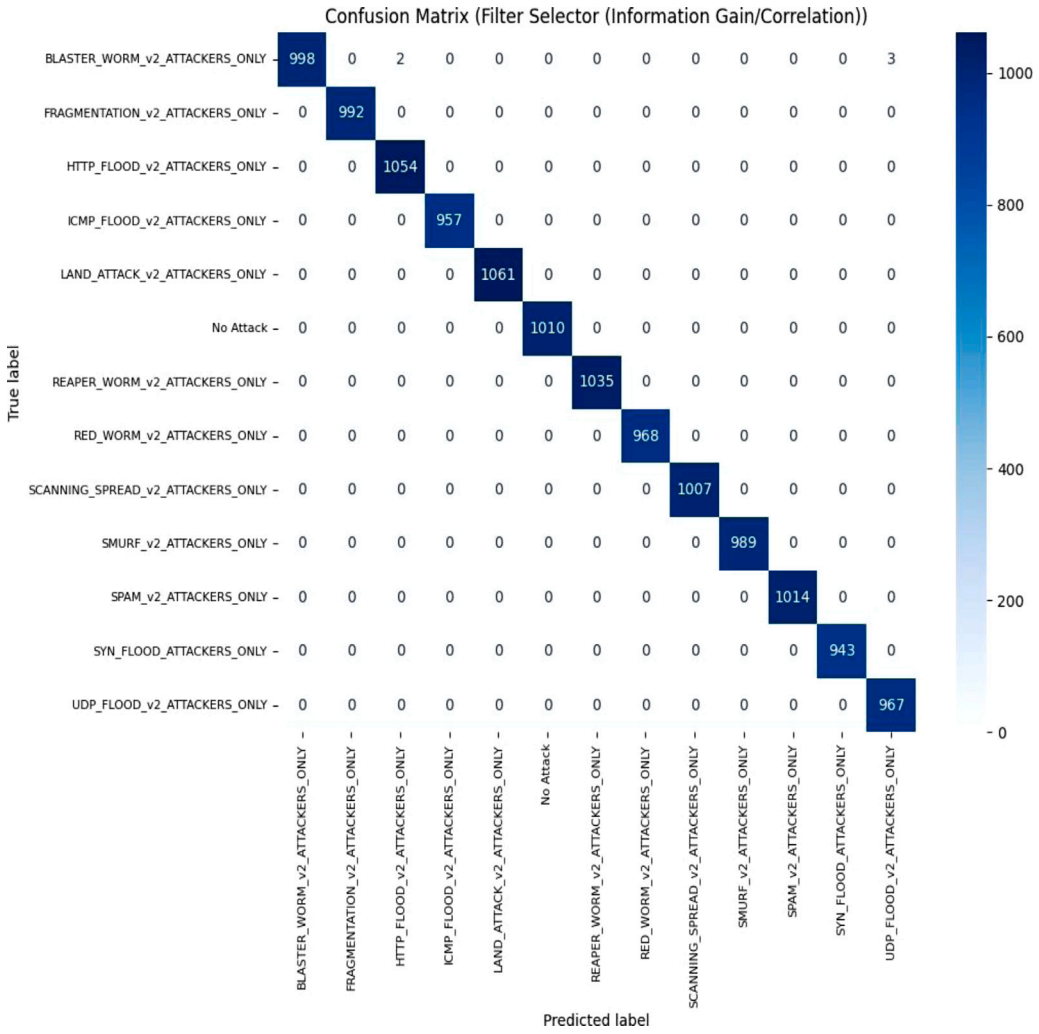

Fig. 5. Confusion matrix when Information Gain is applied.

were also extremely low, indicating poor performance in accurately classifying cases.

Ensemble-based methods such as XGBoost, LightGBM, Random Forest, Extra Trees, and Gradient Boosting demonstrated the highest level of accuracy, precision, recall, and F1 scores, indicating their superior performance in accurately classifying cases. On the other hand, algorithms such as AdaBoost, SVM, QDA, Logistic Regression, Ridge Classifier, and LDA showed lower levels of accuracy and struggled to achieve accurate classifications.

Based on top performing algorithm, XGBoost in this case, we produce confusion matrix as in Fig. 9. The provided confusion matrix represents the prediction results of the XGBoost classification model for different attack categories using the same labels as before. Examining the matrix offers valuable insights into model performance. After analysis, it is clear that the model achieved high accuracy in most cases, as indicated by the items along the diagonal. However, there are a few misclassifications worth noting. For example, in the 'BLASTER_WORM_v2_ATTACKERS_ONLY' class, there were 2 false positive predictions where cases were misclassified as belonging to this attack category. There was one false positive prediction in the 'SPAM_v2_ATTACKERS_ONLY' class that indicated an instance that was incorrectly identified as an attack. In addition, there was one false positive prediction in the "No Attack" class that marked an instance that was misclassified as an attack. Conversely, there was one false negative prediction in this class that indicated an instance that was an attack but was classified as a non-attack. In general, these misclassifications show that the model can be improved to do a better job in separating benign and malignant traffic.

Fig. 10 shows the performance metrics for all classifiers when we use the envelope of the inverse function.

Among the tested algorithms, XGBoost achieved the highest accuracy score of 0.9998, indicating an extremely high level of correct classification. This algorithm also demonstrated excellent precision, recall and F1 score values, all above 0.9997. These results indicate that XGBoost performed exceptionally well in accurately identifying positive cases (precision) and capturing all positive cases (recall), resulting in a balanced F1 score.

LightGBM, Gradient Boosting, Decision Tree, and Random Forest achieved accuracy scores above 0.9992, highlighting their strong performance in correctly classifying cases. These algorithms also showed high precision, recall and F1 score values, indicating their effectiveness in accurately identifying positive cases and achieving a balanced F1 score.

The Extra Trees algorithm achieved a detection accuracy of 0.999, demonstrating a slightly lower but still commendable level of correct classification. Its accuracy, recall, and F1 scores were comparable to the best-performing algorithms, suggesting that Extra Trees can achieve a similar level of performance.

KNN achieved a detection accuracy of 0.9753, which is lower than previous algorithms. However, its accuracy, recall, and F1 scores were still relatively high, indicating its effectiveness in accurately classifying cases, albeit with a slightly higher error compared to the best performing algorithms.

Gaussian Naive Bayes achieved a detection accuracy of 0.7152, indicating a lower level of correct classification compared to previous

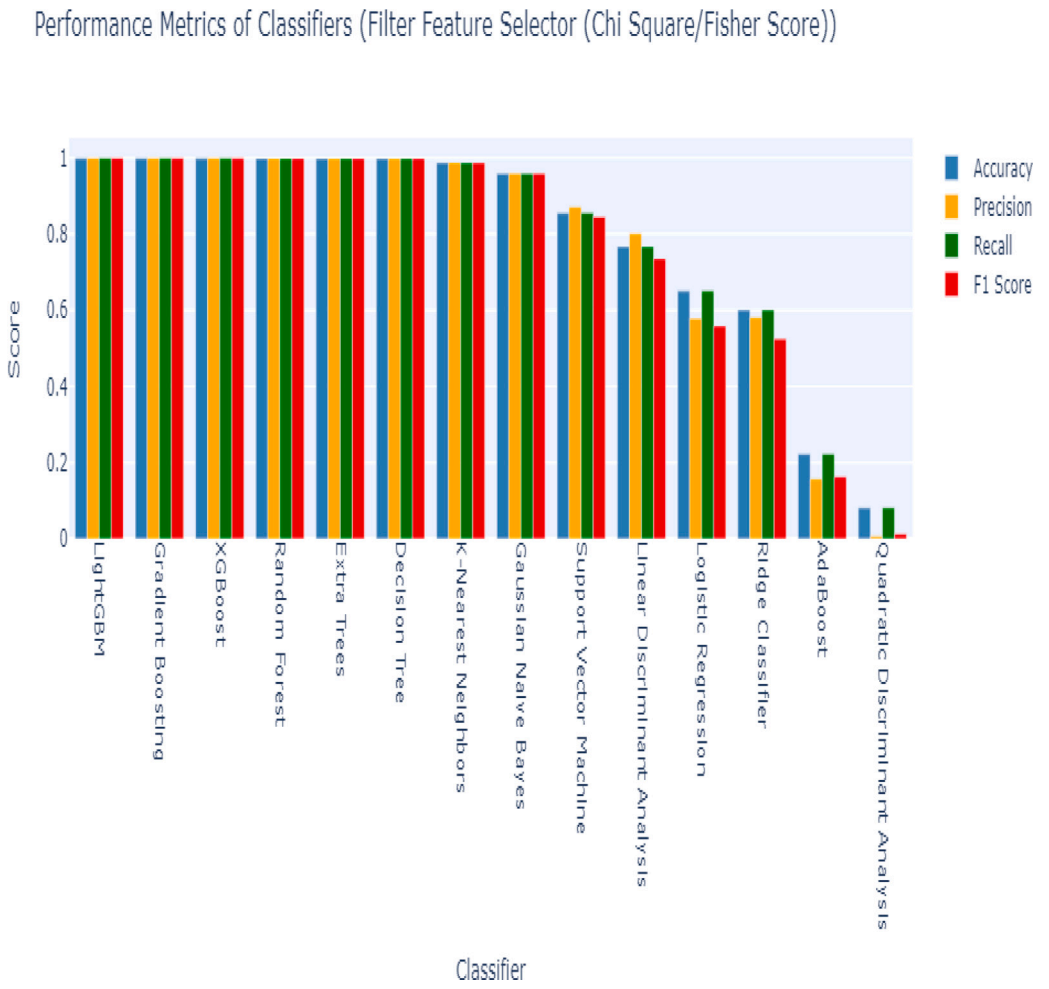

Fig. 6. Performance of different classifiers when Chi Square/Fisher is used for feature selection.

algorithms. Although its precision, recall, and F1 scores were lower than those of the best-performing algorithms, Gaussian Naive Bayes still showed decent performance in accurately classifying cases.

LDA achieved a detection accuracy of 0.5971, indicating a further decrease in correct classification compared to previous algorithms. Its precision, recall, and F1 scores were also lower, indicating limitations in accurately classifying cases.

Ridge Classifier and AdaBoost achieved a detection accuracy of 0.5424 and 0.3903, respectively, indicating a significant drop in correct classification. Their precision, recall, and F1 score values were also lower than previous algorithms, indicating problems in accurately classifying cases.

Logistic regression, QDA, and support vector machine achieved very low accuracy scores, indicating significant difficulty in accurately classifying cases. Their accuracy, recall and F1 score values were extremely low, further highlighting their limitations in this particular scenario.

The results obtained from the classification algorithms differed significantly. The top-performing algorithms such as XGBoost, LightGBM, Gradient Boosting, Decision Tree, and Random Forest have demonstrated high levels of accuracy, precision, recall, and F1 scores, indicating their superior performance in accurately classifying cases. On the other hand, algorithms such as SVM, QDA, Logistic Regression, AdaBoost, Ridge Classifier, and LDA showed lower levels of accuracy and struggled to achieve accurate classifications.

Based on top performing algorithm, XGBoost in this case, we produce a confusion matrix as in Fig. 11. The given confusion matrix corresponds to the results of the classification model's predictions, utilizing the same set of attack labels as mentioned previously. Analyzing the matrix provides valuable insights into the performance of the model. Upon examination, it is evident that the model achieved high accuracy across the majority of instances, as indicated by the values along the diagonal. However, there were a few misclassifications worth noting. In the 'BLASTER_WORM_v2_ATTACKERS_ONLY' class, there were 3 false positive predictions, where instances were incorrectly identified as belonging to this attack category. It is important to address these misclassifications to enhance the accuracy and effectiveness of the model. The model displayed favorable performance, but there is still potential for improvement, particularly in reducing false positive predictions in the 'BLASTER_WORM_v2_ATTACKERS_ONLY' class.

Fig. 12 shows the performance metrics for all classifiers when we use the Ridge Feature selection.

Gradient Boosting, XGBoost, Random Forest, and LightGBM attained high accuracy scores exceeding 0.9974, demonstrating a high level of correct classification by these algorithms. These top-performing algorithms also exhibited high precision, recall, and F1 score values, indicating their effectiveness in accurately identifying positive cases and maintaining a balanced F1 score.

The Decision Tree and Extra Trees algorithms achieved slightly lower detection accuracy of around 0.9972 and 0.9968, respectively. Despite this, they still showed commendable values of precision, recall, and F1 scores, indicating their effectiveness in accurately classifying cases, albeit with a slightly higher margin of error compared to the best performing algorithms.

KNN achieved a detection accuracy of 0.9861, which is lower than previous algorithms. However, its precision, recall and F1 scores

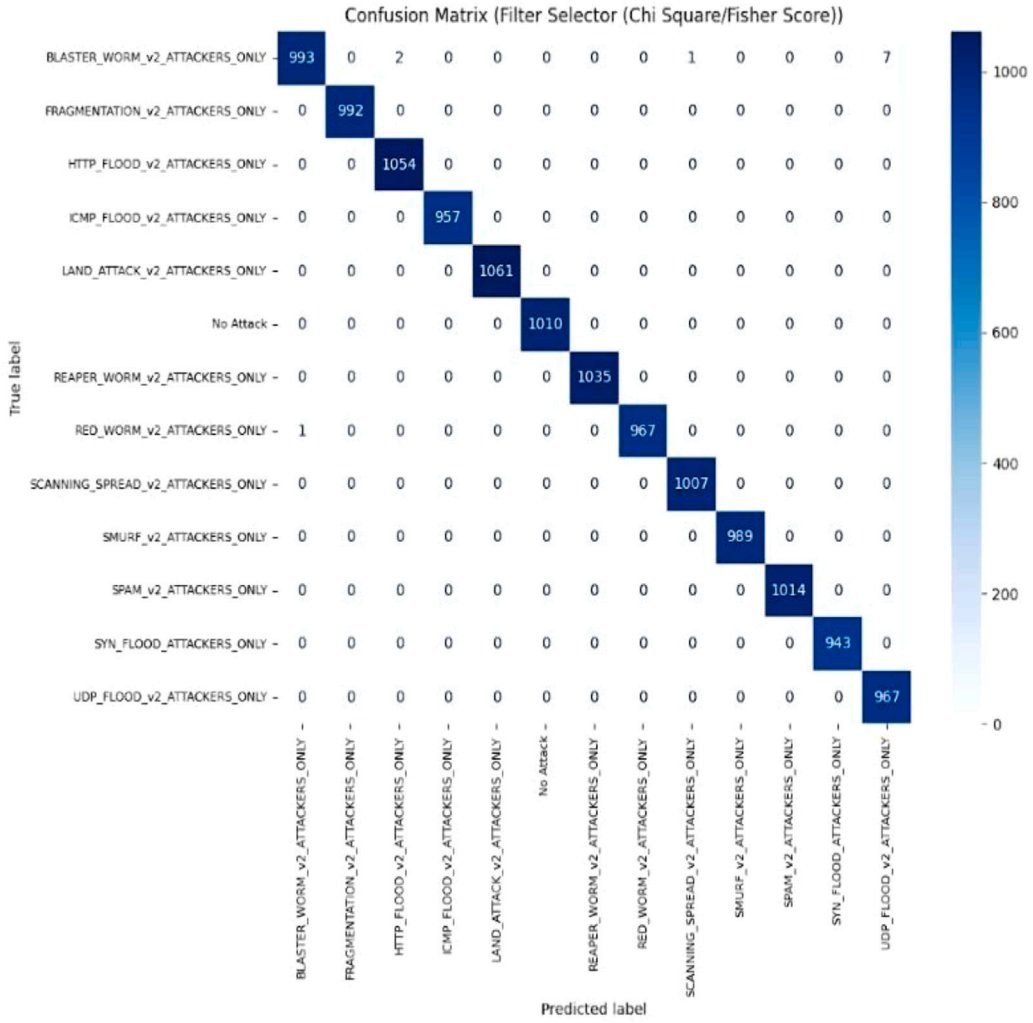

Fig. 7. Confusion matrix when Chi-Square/Fisher is used for FS.

were relatively high, indicating its effectiveness in accurately classifying cases, albeit with a slightly higher error compared to the best performing algorithms.

Gaussian Naive Bayes achieved a detection accuracy of 0.8814, demonstrating a lower correct classification level than previous algorithms. Its precision, recall, and F1 scores were also lower, indicating some limitations in accurately classifying cases.

SVM achieved a detection accuracy of 0.7539, indicating a further decline in correct classifications. Although its precision values and F1 scores were relatively high, the recall value was lower, indicating problems in capturing all positive cases.

LDA achieved a detection accuracy of 0.7003, indicating limitations in correctly classifying cases. Its precision, recall, and F1 score values were also lower, indicating problems in accurately identifying positive cases and achieving balanced F1 scores.

Logistic regression and Ridge Classifier achieved detection accuracy of 0.5379 and 0.5208, respectively, indicating a significant decrease in incorrect classifications. These algorithms also showed lower values of precision, recall, and F1 scores, indicating difficulty in accurately classifying cases.

AdaBoost achieved a detection accuracy of 0.3282, indicating a lower level of correct classification compared to previous algorithms. Its precision, recall, and F1 scores were also lower, indicating limitations in accurately classifying cases.

Finally, QDA achieved extremely low accuracy, precision, recall, and F1 values, indicating significant problems in accurately classifying cases.

The results obtained from the classification algorithms differed significantly. The top-performing algorithms such as Gradient Boosting, XGBoost, Random Forest, LightGBM, Decision Tree, and Extra Trees demonstrated high levels of accuracy, precision, recall, and F1 scores, indicating their superior performance in accurately classifying cases. On the other hand, algorithms such as QDA, AdaBoost, Ridge Classifier, Logistic Regression, LDA, SVM, and Gaussian Naive Bayes showed lower levels of accuracy and struggled to achieve accurate classifications.

Based on top performing algorithm, Gradient Boosting in this case, we produce confusion matrix as in Fig. 13.

The provided confusion matrix reveals the classification results based on the given attack labels generated by the model. Examining the matrix, it is clear that the model achieved high accuracy for most classes, as indicated by the dominant values along the diagonal. However, there have been cases of misclassification that require attention. Notably, 19 cases were misclassified as "PORT_SWEEP" instead of "DDoS" and 4 cases were misclassified as "BLASTER_WORM_v2_ATTACKERS_ONLY" instead of "PORT_SWEEP". These misclassifications indicate areas for improvement in model accuracy, particularly in reducing false positives. The matrix provides valuable numerical insights to help researchers refine the model and improve its attack classification performance.

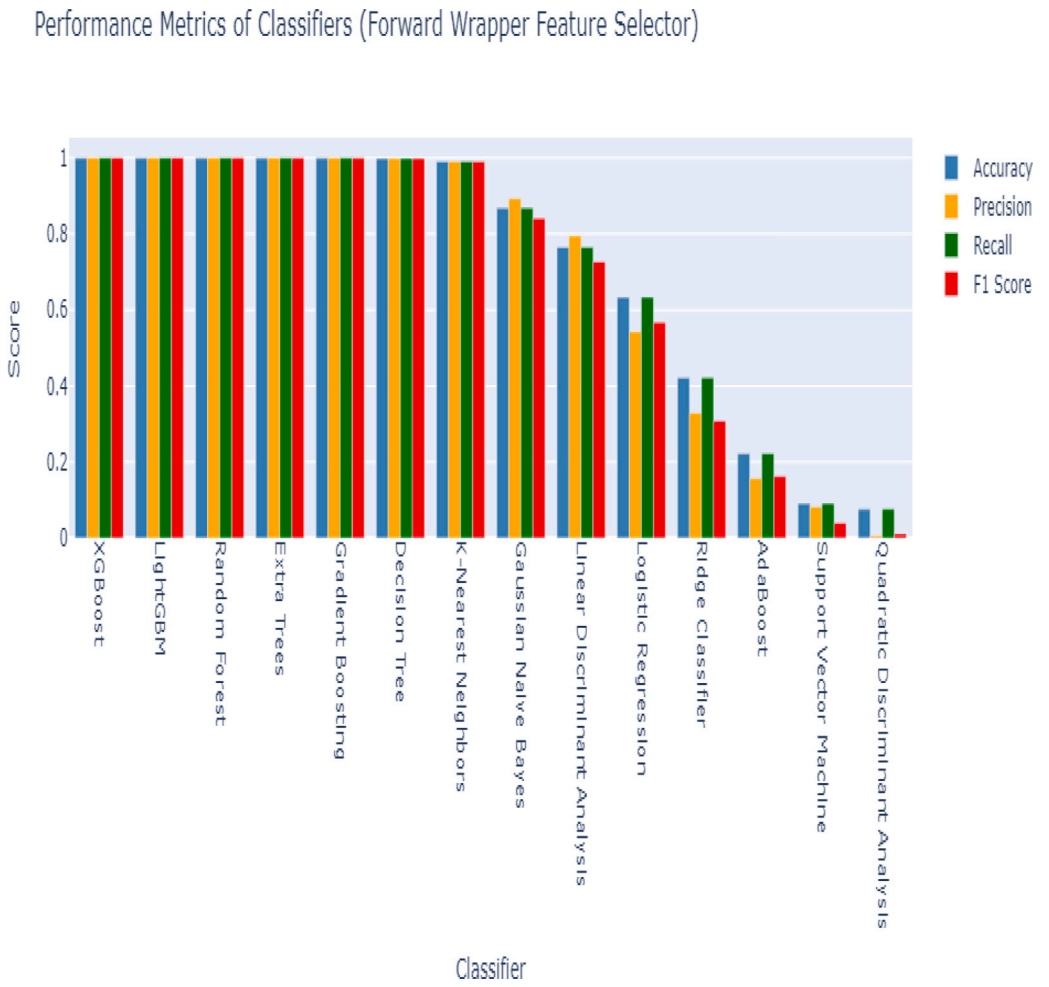

Fig. 8. Performance of different classifiers when forward wrapper is used for FS.

4.5. Comparison

4.5.1. F1 score

In this analysis, we evaluated the performance of different machine learning algorithms using different feature selection techniques based on their F1 scores (Fig. 14). Algorithms tested include XGBoost, Gradient Boosting, LightGBM, Extra Trees, Random Forest, Decision Tree, KNN, Gaussian Naive Bayes, SVM, LDA, Logistic Regression, Ridge Classifier, AdaBoost, and QDA.

Among the algorithms tested without feature selection, XGBoost, Gradient Boosting, and LightGBM achieved the highest F1 scores with values ranging from 0.9997 to 0.9998. However, when you use feature selection using wrapper methods such as backward and forward selection, the F1 score decreased slightly, but remained high for XGBoost, Extra Trees, and Gradient Boosting.

When feature selection was performed using filtering methods such as information gain, correlation, chi-square, and Fisher's score, the F1 scores differed in different algorithms. Gradient Boosting, Random Forest, and XGBoost showed consistent performance across different filtering methods, while LightGBM had different F1 scores depending on the filter method used.

4.5.2. Precision

Based on the precision scores obtained from different classifiers, we can observe that several algorithms achieve high accuracy values (Fig. 15). XGBoost consistently performs well across different feature selection methods, achieving accuracy scores above 0.9997. LightGBM also shows good performance without feature selection with an accuracy score of 0.9997.

Gradient Boosting, Extra Trees, and Random Forest classifiers also show high accuracy scores when used without feature selection, ranging from 0.9996 to 0.9997.

In terms of feature selection methods, Wrapper's backward elimination approach consistently produces highly accurate scores across multiple classifiers, including XGBoost, LightGBM, and Gradient Boosting. These methods achieve accuracy scores ranging from 0.9994 to 0.9997.

The filter method, specifically using information gain proved effective for feature selection with Gradient Boosting, Random Forest, and XGBoost classifiers, yielding accuracy scores ranging from 0.9995 to 0.9996.

For KNN, the accuracy score remains consistently high across different feature selection methods such as the Information Gain filter, no feature selection, and wrapper forward, with scores ranging from 0.9906 to 0.9907.

Gaussian Naive Bayes achieves a high accuracy score using the filter method (Chi Square/Fisher Score) with a score of 0.9593. It also performs well without feature selection, envelope passing, and embedding (Ridge), with scores ranging from 0.8776 to 0.8925.

SVM shows good accuracy with filter (Chi Square/Fisher Score) and embedding (Ridge) methods, achieving a detection accuracy of 0.8710 and 0.8191, respectively.

LDA performs relatively well without feature selection and with the filter method (Chi Square/Fisher Score), achieving detection accuracy of 0.8030 and 0.8005, respectively.

On the other hand, several classifiers show lower accuracy scores. Logistic Regression, Ridge Classifier, AdaBoost, and QDA show lower accuracy values across different feature selection methods.

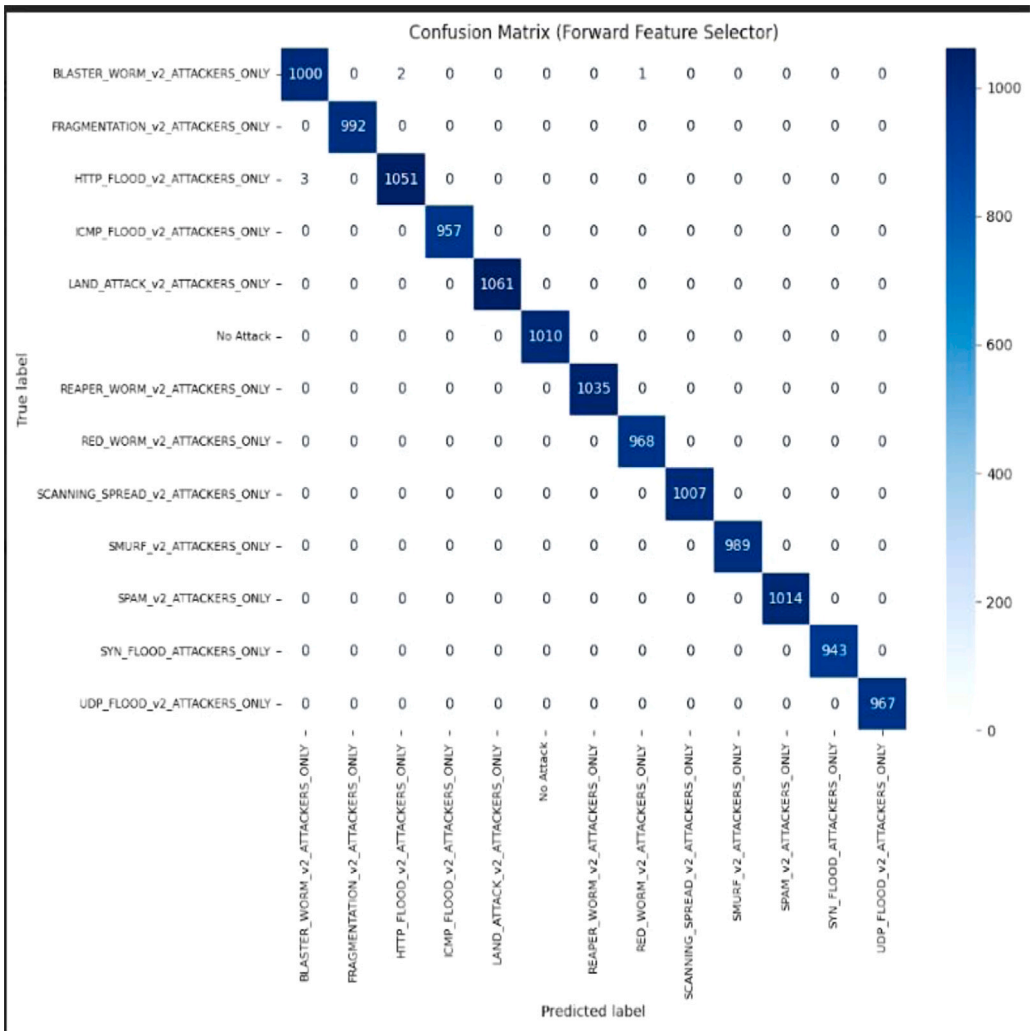

Fig. 9. Performance of different classifiers when forward wrapper is used for FS.

The results highlight the importance of selecting appropriate feature selection methods and classifiers to achieve high accuracy in classification tasks.

4.5.3. Recall

In this analysis, we used different feature selection techniques to compare several classification algorithms based on their recall scores (Fig. 16). Recall measures the ability of the model to correctly identify positive instances from all true positive cases in the data set. A higher score indicates a better ability to catch positive cases.

Among the evaluated algorithms, XGBoost achieved the highest recall score of 0.9998 without using any feature selection technique. XGBoost also performed well, with a recall score of 0.9998 when the back-convolution method was used for feature selection. LightGBM and Gradient Boosting also achieved a high recall score of 0.9998 without feature selection.

When the backwrap method was used, XGBoost, LightGBM, and Random Forest achieved a recall score of 0.9997, indicating their effectiveness in selecting relevant features. The forward wrapper method resulted in high recall scores for XGBoost and LightGBM (0.9997).

Using filter-based feature selection techniques such as Information Gain and chi-square/Fisher score showed mixed results. Gradient Boosting, Random Forest, and XGBoost achieved a recall score of 0.9996 when Information Gain was used. In contrast, Gradient Boosting, XGBoost, and Random Forest achieved a recall score of 0.9995 when chi-squared/Fisher's score was used.

The decision tree and extra trees also showed good performance in terms of recall using different feature selection techniques. Both algorithms achieved a recall score of 0.9994 with the backward and forward wrapper methods.

For most algorithms, nesting with ridge regression resulted in relatively lower memory scores ranging from 0.9975 to 0.9905. KNN achieved a recall score of 0.9905 when Information Gain was used for feature selection.

Moving to the lower end of recall scores, Gaussian Naive Bayes achieved a recall score of 0.9588 when the chi-square/Fisher score was used for feature selection. SVM, LDA, Logistic Regression, and Ridge Classifier obtained recall scores ranging from 0.8552 to 0.6057, indicating a relatively lower ability to catch positive cases.

Finally, AdaBoost and QDA had the lowest recall scores, ranging from 0.3903 to 0.0725, across different feature selection techniques.

XGBoost consistently performed well in terms of recall, scoring high with and without feature selection techniques. Packing methods, especially backward and forward, have been shown to be effective for selecting relevant features. Filter-based techniques have shown mixed results, with some algorithms benefiting from Information Gain or chi-square/Fisher's score. However, fitting with ridge regression generally resulted in lower recall scores. It is important to note that these results may vary depending on the dataset and the specific problem domain.

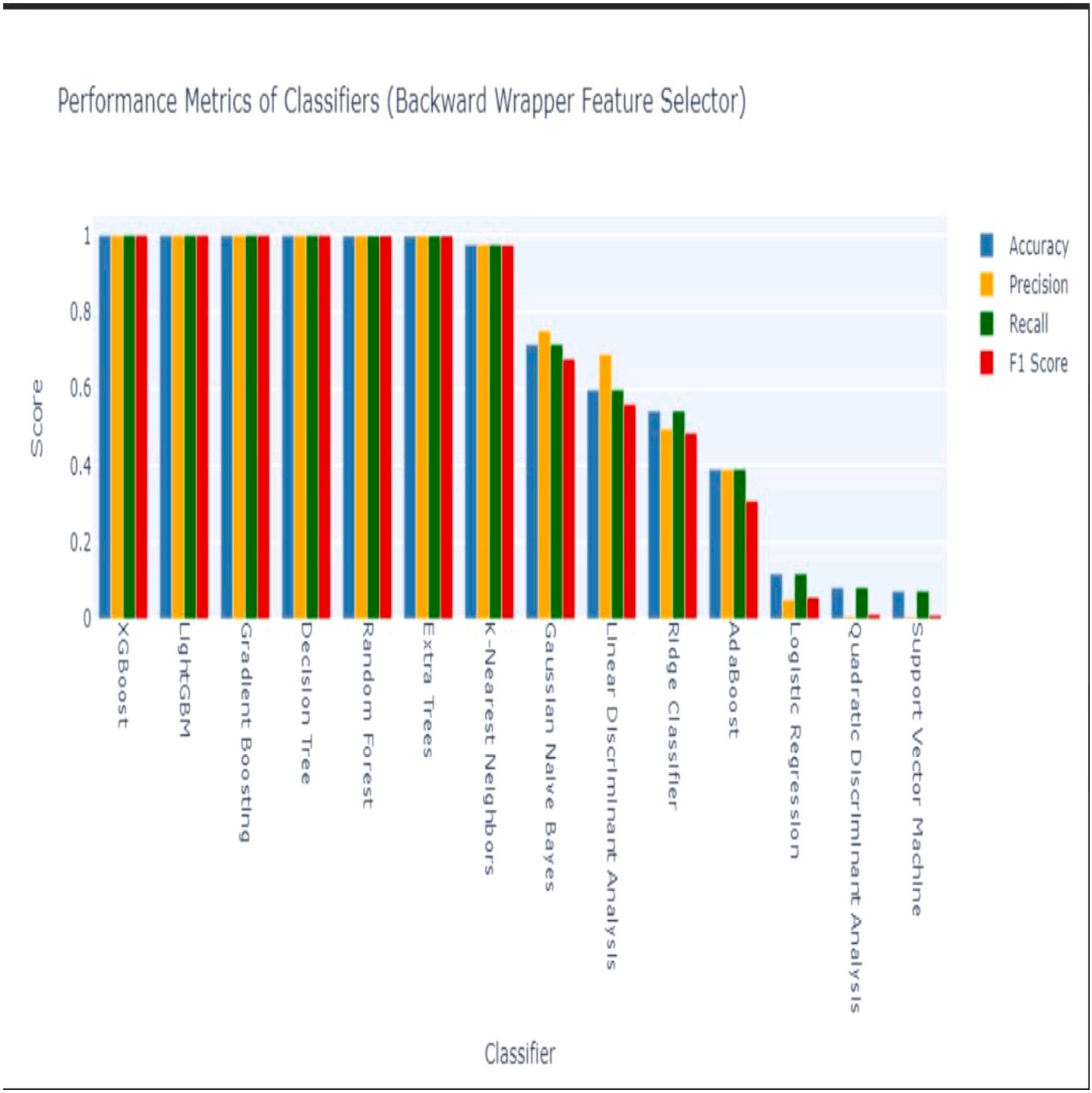

Fig. 10. Performance of different classifiers when the backward wrapper is used for FS.

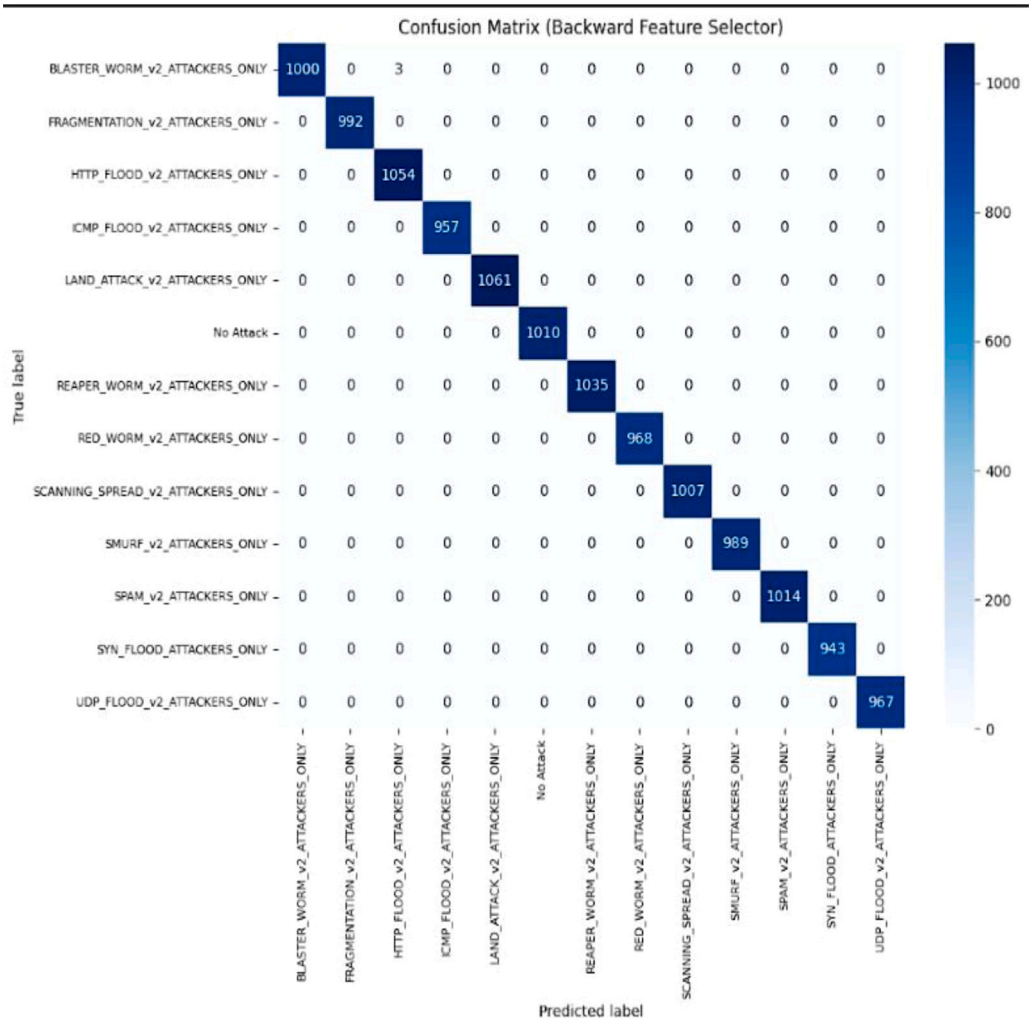

Fig. 11. Performance of different classifiers when the backward wrapper is used for FS.

4.5.4. Accuracy

In this analysis, we have evaluated the performance of different machine learning algorithms on a classification task (Fig. 17). The accuracy results obtained for each algorithm are as follows:

1. XGBoost achieved the highest accuracy of 99.98% when no feature selection was applied. It also performed well in feature backselection, achieving 99.98% accuracy.

2. LightGBM and Gradient Boosting achieved 99.98% accuracy without feature selection. LightGBM also performed well with feature backselection, achieving an accuracy of 99.97

3. Random Forest and Extra Trees algorithms achieved 99.97% accuracy without feature selection. Extra Trees also performed well in both forward and backward feature selection, achieving equal accuracy.

4. Wrapper methods such as forward and backward feature selection have been shown to be effective for several algorithms, including XGBoost, LightGBM, and Gradient Boosting, achieving accuracies ranging from 99.69% to 99.92%.

5. The filtering methods, namely Information Gain and Chi-Square/Fisher Score, were effective for Gradient Boosting, LightGBM and Random Forest algorithms, achieving accuracy ranging from 99.53% to 99.92%.

6. The decision tree algorithm achieved 99.85% accuracy without feature selection and performed well with backward and forward feature selection, achieving 99.84% and 99.84% accuracy, respectively.

7. Insertion with the Ridge classifier yielded accuracy ranging from 99.75% to 99.99% for the XGBoost, Gradient Boosting, Random Forest, LightGBM, and Decision Tree algorithms.

8. KNN achieved accuracy ranging from 97.50% to 99.05% depending on the feature selection method used.

9. The Gaussian Naive Bayes algorithm attained a detection accuracy of 95.88% with chi-square/Fisher score feature selection and performed well when fitted using the Ridge classifier, achieving an accuracy of 88.14%.

10. SVM and LDA algorithms achieved accuracy ranging from 72.42% to 85.52% depending on the feature selection method used.

11. Logistic regression and Ridge Classifier algorithms achieved accuracies ranging from 42.21% to 66.53% depending on the feature selection method used.

12. The AdaBoost algorithm achieved accuracy ranging from 22.23% to 39.03% depending on the feature selection method used.

Overall, XGBoost, LightGBM, Gradient Boosting, and Random Forest algorithms performed consistently well across different feature selection methods, achieving accuracies above 99.5%. These algorithms can be considered as strong candidates for classification tasks. The feature selection method's choice greatly influenced the algorithms' performance, with wrapper and filter methods proving effective in improving accuracy in several cases.

5. Evaluation of multi-layered threat intelligence framework

The effectiveness of a cybersecurity framework like the Multi-Layered Threat Intelligence Framework (MLTIF) can be evaluated

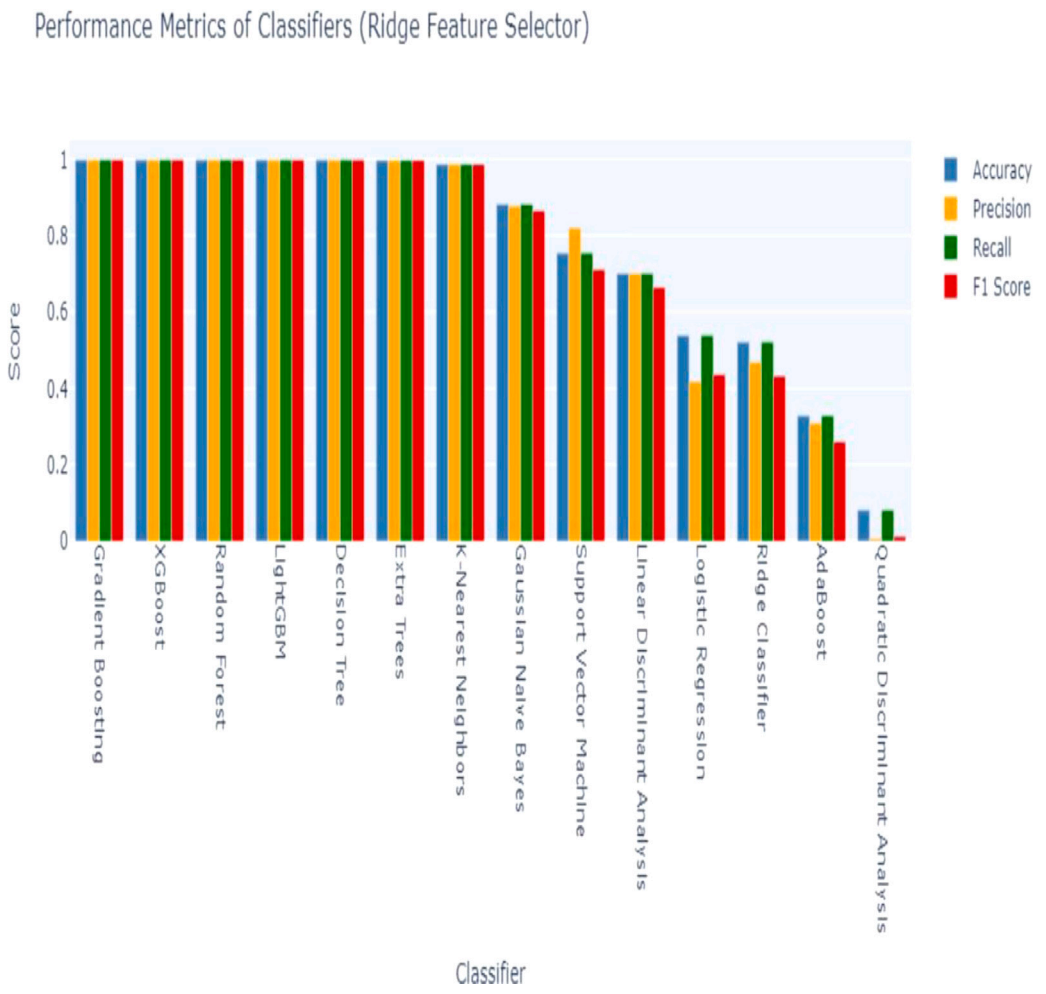

Fig. 12. Performance of different classifiers when Ridge wrapper is used for FS.

across several dimensions, including accuracy and reliability, effectiveness in threat detection and mitigation, and the impact of various threat intelligence sources.

5.1. Accuracy and reliability

The accuracy and reliability of MLTIF can be highly dependent on the quality and reliability of the data it receives. With its first layer dedicated to data collection and processing, the MLTIF has the potential to gather a wide variety of valuable information. However, the accuracy of its threat analysis will depend largely on the quality of this input data. If inaccurate or irrelevant data is collected, this could result in false positives or false negatives. The reliability of MLTIF depends on its layers' consistency in performing their functions. If the machine learning algorithms and artificial intelligence models used in the threat analysis layer are not robust or properly trained, they may not consistently interpret the input data correctly, leading to inconsistent performance.

5.2. Effectiveness in threat detection and mitigation

The layered structure of MLTIF allows it to effectively detect and mitigate threats. The second layer, which focuses on threat intelligence analysis, uses machine learning and artificial intelligence to identify and classify threats. This approach can provide a more comprehensive view of potential threats, as it combines the strengths of automated machine learning and artificial intelligence with the insights of human expertise.

The threat detection and mitigation layer allows for real-time responses to potential threats, utilizing intrusion detection and prevention systems, firewalls, and other security measures. These measures can prevent threats from penetrating the system, ensuring that they are identified and mitigated before they can cause damage.

5.3. Impact of different threat intelligence sources

The MLTIF leverages a variety of threat intelligence sources, which can have a significant impact on its effectiveness. By integrating both internal and external threat intelligence sources, the MLTIF can provide a more comprehensive picture of the threat landscape. Internal sources provide insights into an organization's unique threat profile, which includes understanding previous attack patterns, vulnerabilities, and areas of risk within the organization. External sources can offer broader intelligence about global trends, emerging threats, tactics, techniques, and procedures (TTPs) of various threat actors. However, the challenge lies in effectively integrating these diverse sources of information. Important data might be overlooked or misinterpreted if this integration is not done well. The value of threat intelligence is highly dependent on its timeliness. If the collected intelligence is outdated, it could lead to ineffective threat detection and response.

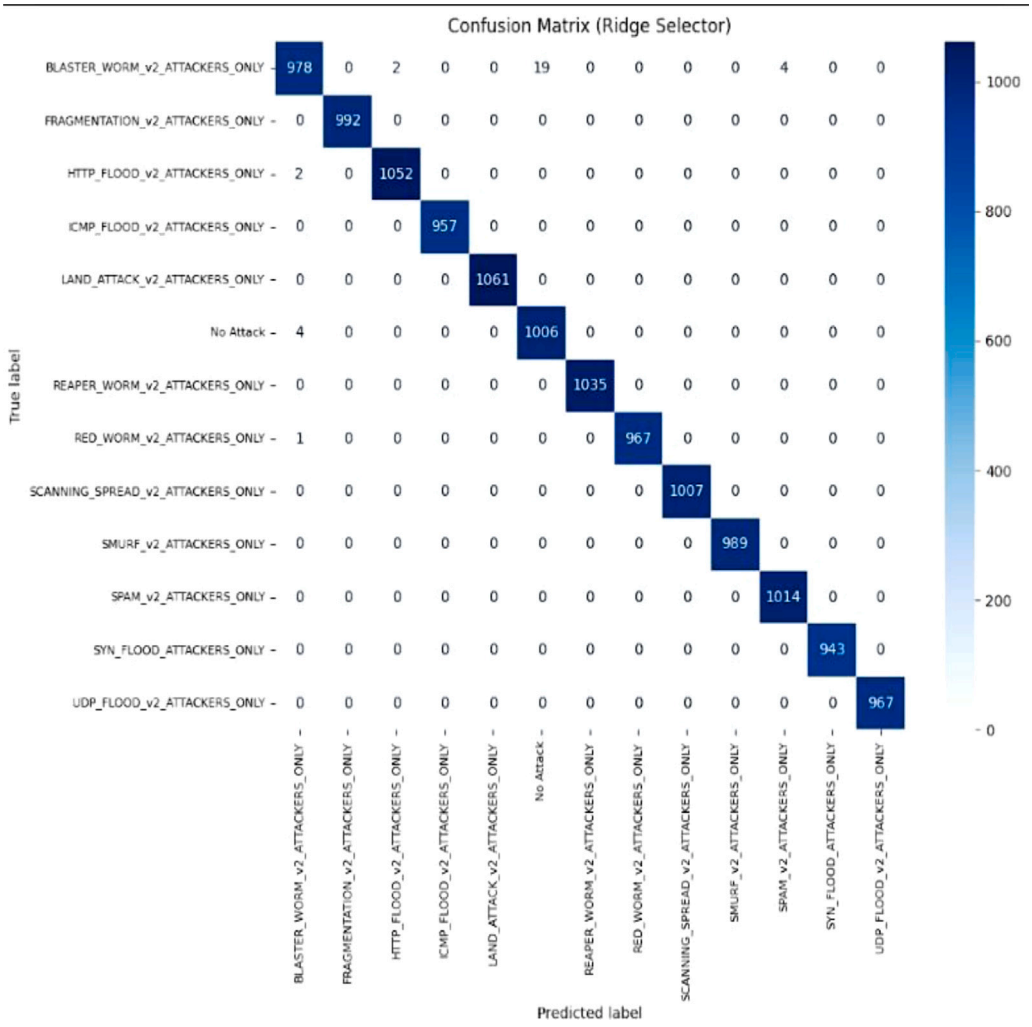

Fig. 13. Confusion matrix when Ridge wrapper is used for FS.

While the MLTIF provides a robust and comprehensive architecture for threat detection and mitigation, its effectiveness is dependent on several factors, including the quality and timeliness of the data, the performance of its machine learning and artificial intelligence algorithms, and the effective integration of its multiple layers and sources of threat intelligence.

5.4. Comparison with existing frameworks

The Multi-Layered Threat Intelligence Framework (MLTIF) appears to offer a robust architecture for cybersecurity, given its emphasis on multiple layers of threat intelligence. Here's how it compares to some other commonly used cybersecurity frameworks:

1. NIST Cybersecurity Framework (CSF) has been developed by NIST [51]. Its purpose is to help organizations manage and reduce cybersecurity risk. It is divided into five functions: Identify, Protect, Detect, Respond, and Recover, which align closely with the layers defined in the MLTIF. However, unlike the MLTIF which emphasizes on real-time feedback and machine learning for threat identification, NIST CSF is more focused on managing risk and improving cybersecurity across an organization and might not provide the same level of real-time threat intelligence.
2. ISO/IEC 27001 is an internationally recognized standard for an Information Security Management System (ISMS) [52]. It includes procedures and policies for systematically managing an organization's information risk. While the MLTIF is primarily focused on threat intelligence and rapid response, ISO/IEC 27001 is more comprehensive, focusing on the entire ISMS, which includes elements such as risk assessment, internal audit, continual improvement, and top management's commitment to information security. However, MLTIF might provide a more advanced threat detection approach given its use of artificial intelligence and machine learning.
3. CIS Controls (formerly SANS Top 20) is recommended a set of cybersecurity protection actions and offers a prioritized approach to securing your IT environment [53]. While CIS Controls are focused on actions to be taken, MLTIF is more of an architectural framework that includes data collection, analysis, threat detection, mitigation, reporting, feedback, and management, offering a more holistic and potentially more adaptable approach.
4. Control Objectives for Information and Related Technologies (COBIT) is a comprehensive framework for the governance and management of enterprise IT [54]. COBIT provides guidance to executives and those charged with making decisions concerning the use of technology in support of organizational objectives. While both COBIT and MLTIF stress management and

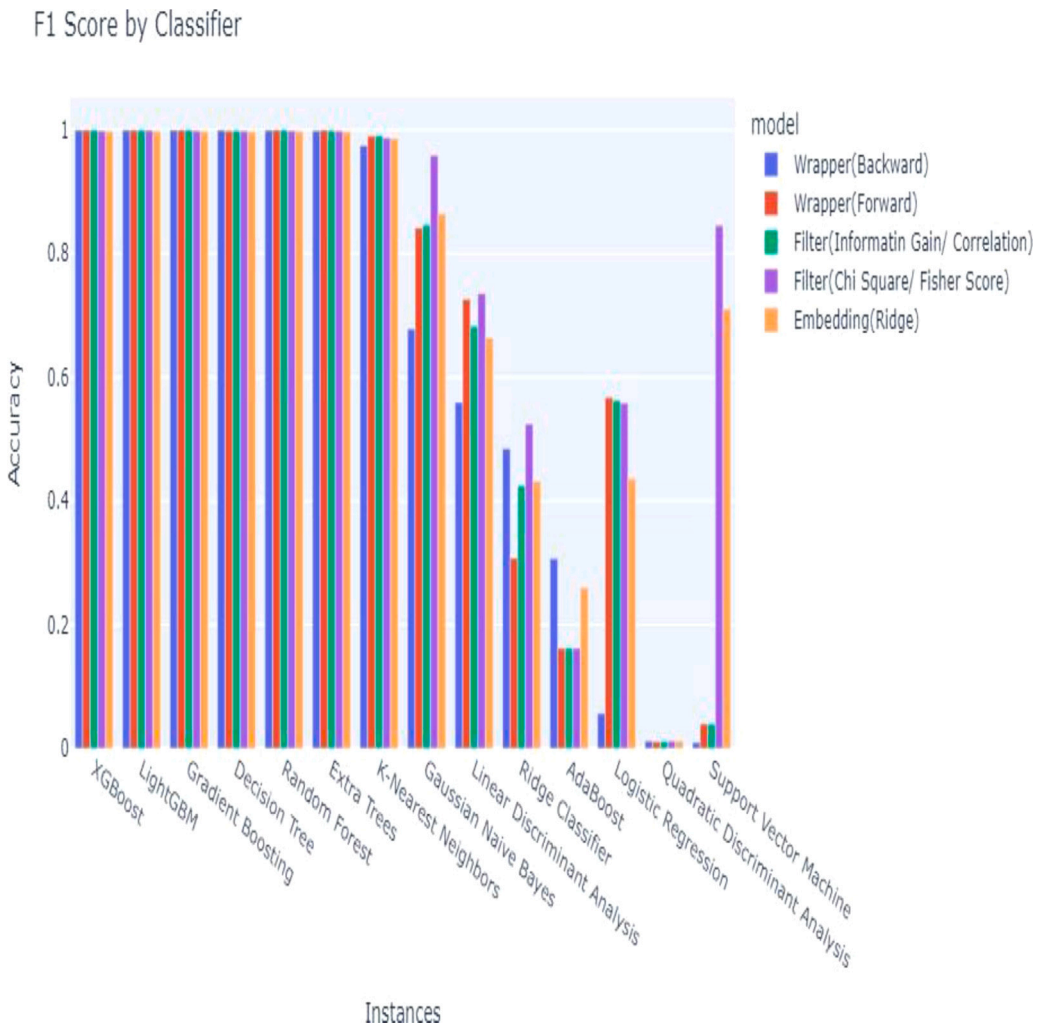

Fig. 14. F-score of classification.

governance, MLTIF specifically targets threat intelligence, detection, and mitigation, making it more suitable for organizations looking for a robust cybersecurity focus.

The MLTIF, with its focus on threat intelligence, might provide a more targeted and real-time approach to cybersecurity, especially for organizations dealing with advanced persistent threats. However, the choice of a cybersecurity framework often depends on an organization's specific needs, including its risk tolerance, regulatory requirements, and resources.

5.5. Comparisons with previous work

We have included comparative assessments with current cybersecurity frameworks to more accurately assess the efficacy of our suggested solutions. A comparison of our study's salient features with well-known frameworks like MITRE ATT&CK and conventional signature-based systems is provided below:

MLTIF: To enable quick, real-time threat detection, our framework combines machine learning-based and signature-based detection techniques. By combining various threat feeds, it exhibits great agility and strengthens organizational defenses against changing cyberthreats, [12,24,25].

MITRE ATT&CK: This framework is centered around technique- and tactic-based detection, which can have different reaction times based on the tactics employed. Although robust, MITRE ATT&CK needs a great deal of fine-tuning and customization in order to successfully handle certain threats, [55].

Conventional Signature-Based Systems: These systems use pattern matching and rule-based techniques, which provide quick detection but little flexibility to respond to emerging threats.

We show how the MLTIF performs better and is more versatile in bolstering organizational resilience against cyber threats by comparing detection rates, response times, and overall adaptability to emerging threats, [56].

6. Conclusion

In this study, we evaluated various machine learning algorithms using different feature selection techniques, focusing on their F1 scores, accuracy, precision, and recall. XGBoost, Gradient Boosting, and LightGBM consistently achieved high F1 scores, while the effectiveness of feature selection methods varied across algorithms. XGBoost demonstrated high precision and recall scores, and LightGBM also performed well in these metrics. Wrapper methods, such as backward and forward selection, showed promising results for feature selection. Filter

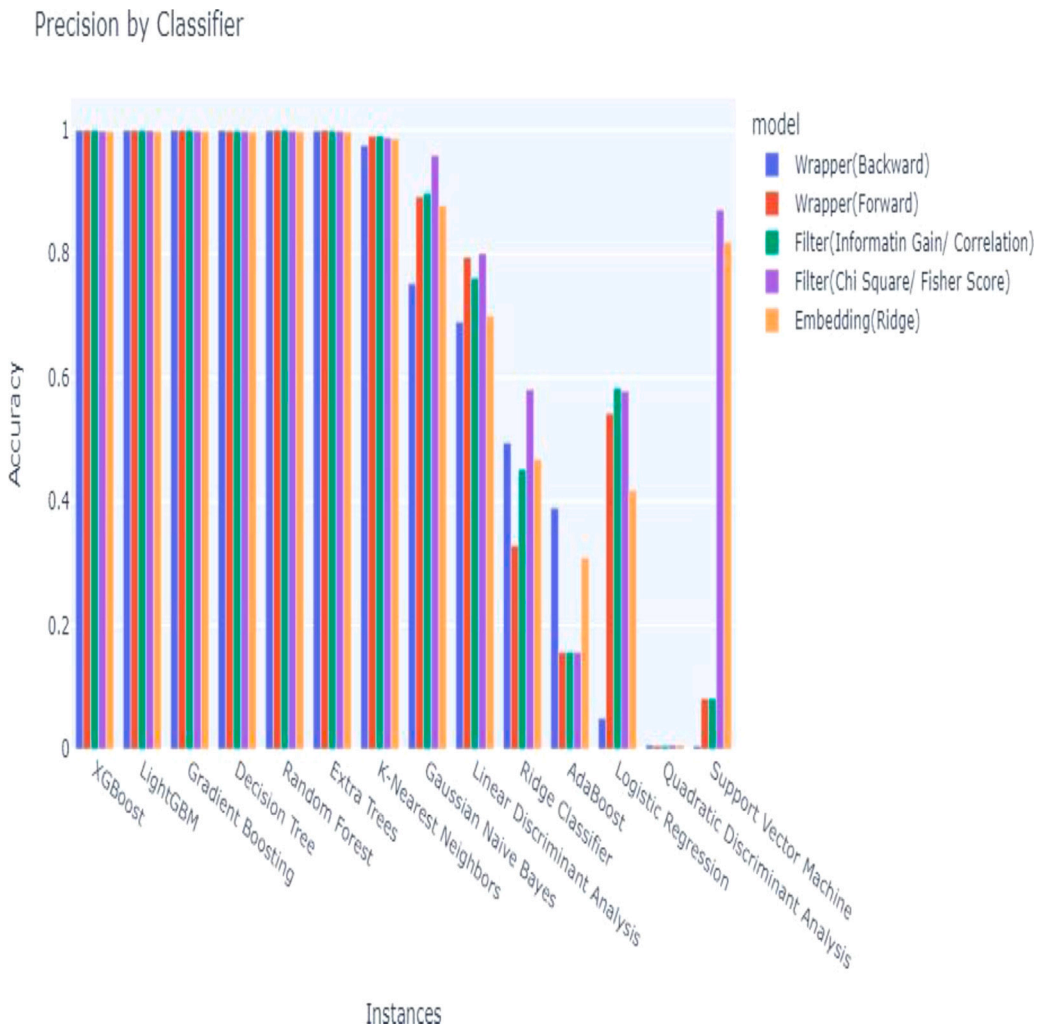

Fig. 15. Precision of classification.

methods, such as information gain and correlation, were effective for Gradient Boosting, Random Forest, and XGBoost. However, embedding techniques, particularly ridge embedding, resulted in lower F1 scores for all algorithms tested. XGBoost attained the highest accuracy, while LightGBM and Gradient Boosting also performed well. KNN demonstrated robustness across feature selection techniques, while Logistic Regression and QDA exhibited lower performance.

Despite the potential advantages of the Multi-Layered Threat Intelligence Framework (MLTIF), several limitations exist:

- The effectiveness of MLTIF is highly dependent on the quality and volume of threat intelligence data it receives. Poor-quality data or lack of sufficient data can lead to inaccurate threat identification and mitigation.
- Integration of multiple layers and various threat intelligence sources can be challenging, as it requires significant effort to ensure all parts of the system work coherently.
- MLTIF, like any threat detection system, can produce false positives and negatives. These could lead to unnecessary responses or overlooked threats, respectively.
- As the amount of data grows, scalability could become a problem. The MLTIF needs to maintain its effectiveness as it scales, which can be technically challenging.
- The success of MLTIF relies on the performance of machine learning algorithms. If these technologies fail to deliver, the framework's effectiveness could be compromised.

Recent studies have shown the applicability of our current analysis and provide insightful background. For example, utilizing data-driven decision support frameworks, Nisioti et al. (2023) investigated innovations in forensic investigations [26]. Explainable machine learning models are used for intrusion detection, and Nwakanma et al. (2023) looked at how to combine IoT with ITS to create IoV [27]. In order to automate attack detection, Macas et al. (2022) explored the use of deep learning in cybersecurity [20]. In order to automatically identify and profile newly emerging cyber risks, Marinho and Holanda (2023) used social media analysis [28].

Theoretically, by highlighting the value of combining several threat information sources into a multi-layered framework and casting doubt on preconceived notions about the limitations of machine learning algorithms in cybersecurity, our findings advance our understanding. Through practical insights on enhancing threat intelligence data quality and integration, as well as approaches to manage scalability and minimize false positives/negatives, our research applies to real-world applications.

By addressing these limitations and focusing on these future work areas, the MLTIF could be significantly improved and provide an even more robust and effective solution for threat detection and mitigation.

Future work should focus on addressing these limitations:

- Develop advanced techniques for improving the quality of data collected and to effectively handle larger volumes of data.
- Create more effective strategies for integrating multiple layers and different sources of threat intelligence.

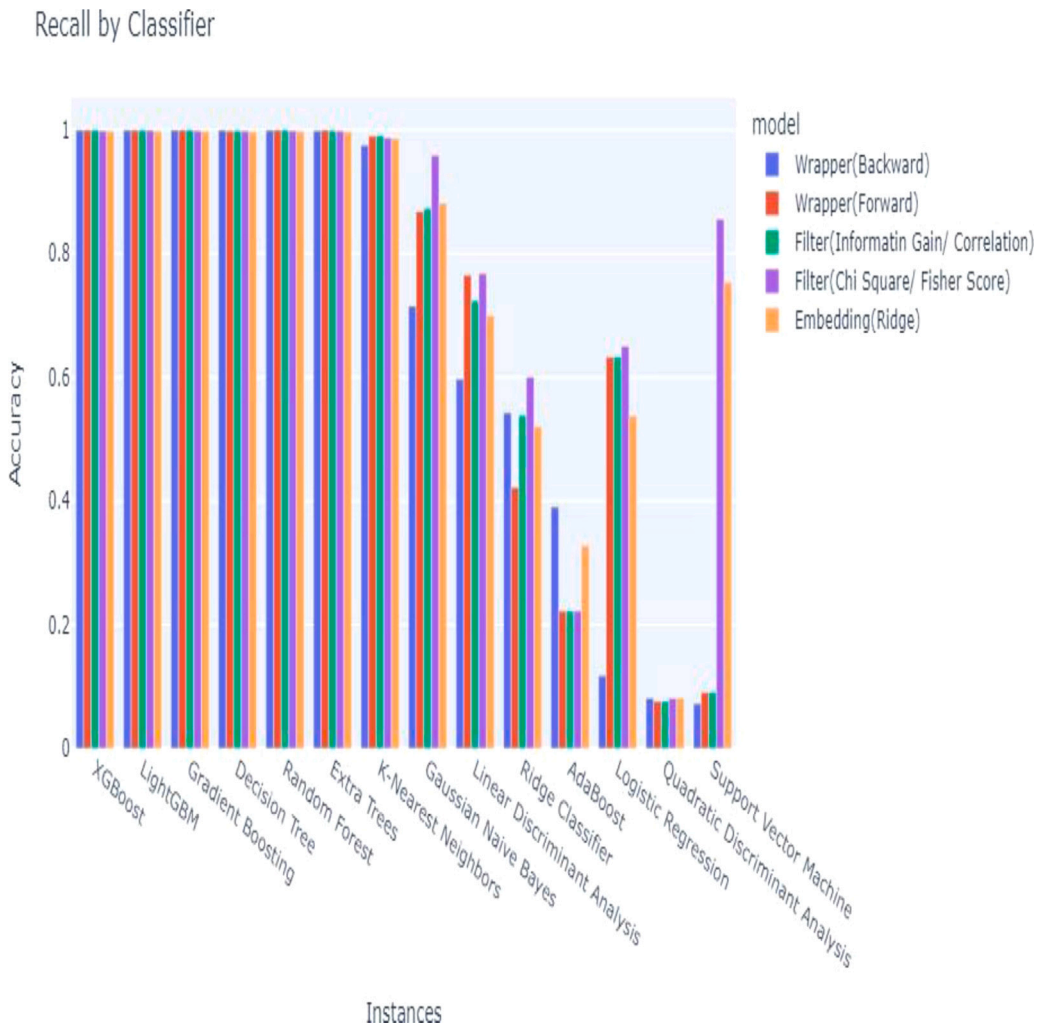

Fig. 16. Recall of classification.

- Improve techniques to reduce the number of false positives and negatives, possibly by refining the machine learning used in the framework.
- Explore strategies for managing the scalability of the framework, such as distributed processing or cloud-based solutions, could be a key area for future work.
- Thoroughly test MLTIF in various real-world scenarios to ensure its effectiveness in different contexts and to further refine its capabilities based on these experiences.

CRediT authorship contribution statement

Moutaz Alazab: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Ruba Abu Khurma:** Data curation, Investigation, Methodology, Project administration, Resources, Validation, Visualization, Writing – original draft, Writing – review & editing, Conceptualization, Formal analysis, Funding acquisition, Software, Supervision. **Maribel García-Arenas:** Conceptualization, Funding acquisition, Investigation, Project administration, Supervision, Writing – review & editing. **Vansh Jatana:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Ali Baydoun:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Robertas Damaševičius:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, 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.

Acknowledgments

This work was supported by the Grant C-ING-027-UGR23 funded by Consejería de Universidad, Investigación e Innovación and by ERDF Andalusia Program 2021-2027, the Ministerio Español de Ciencia e Innovación under project number PID2020-115570GB-C22 MCIN/AEI/10.13039/501100011033, Grant PID2022-137461NB-C31 funded by MCIN/AEI/10.13039/501100011033 and by the Cátedra de Empresa Tecnología para las Personas (UGR-Fujitsu).

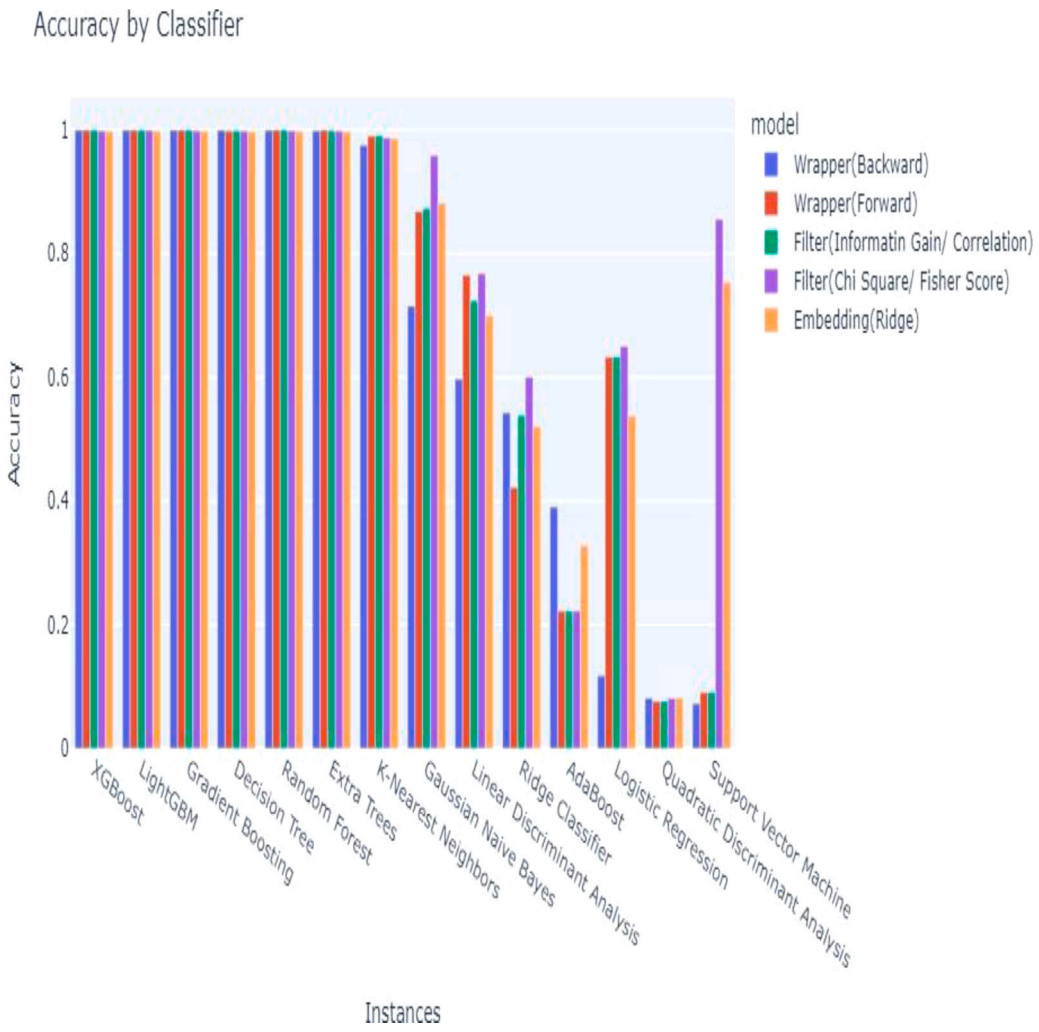

Fig. 17. Accuracy of classification.

References

[1] Kolini F, Janczewski L. Exploring incentives and challenges for cybersecurity intelligence sharing (CIS) across organizations: A systematic review. Commun Assoc Inf Syst 2022;50(1):86–121.

[2] de Souza CA, Westphall CB, Machado RB, Loffi L, Westphall CM, Geronimo GA. Intrusion detection and prevention in fog based IoT environments: A systematic literature review. Comput Netw 2022;214.

[3] Malliga S, Nandhini PS, Kogilavani SV. A comprehensive review of deep learning techniques for the detection of (distributed) denial of service attacks. Inf Technol Control 2022;51(1):180–215.

[4] Saxena R, Gayathri E. Cyber threat intelligence challenges: Leveraging blockchain intelligence with possible solution. Mater Today: Proc 2021;51:682–9.

[5] Pinto A, Herrera L, Donoso Y, Gutierrez JA. Survey on intrusion detection systems based on machine learning techniques for the protection of critical infrastructure. Sensors 2023;23(5).

[6] Ahmadi A, Smith J. Security vulnerabilities in cyber-physical systems. J Cybersecur 2022;10(3):45–58. http://dx.doi.org/10.1007/s11276-022-0300-5.

[7] Ju H, Jeon B, Kim D, Jung B, Jung K. Security considerations for in-vehicle secure communication. In: 2019 international conference on information and communication technology convergence. ICTC, IEEE; 2019, p. 1404–6.

[8] Almuqren L, Maashi MS, Alamgeer M, Mohsen H, Hamza MA, Abdelmageed AA. Explainable artificial intelligence enabled intrusion detection technique for secure cyber-physical systems. Appl Sci 2023;13(5).

[9] Rathish CR, Karpagavadivu K, Sindhuja P, Kousalya A. A hybrid efficient distributed clustering algorithm based intrusion detection system to enhance security in manet. Inf Technol Control 2021;50(1):45–54.

[10] Thomasian NM, Adashi EY. Cybersecurity in the internet of medical things. Health Policy Technol 2021;10(3).

[11] Altulaihan E, Almaiah MA, Aljughaiman A. Cybersecurity threats, counter-measures and mitigation techniques on the IoT: Future research directions. Electronics 2022;11(20).

[12] Ali MH, Jaber MM, Abd SK, Rehman A, Awan MJ, Damasevicius R, et al. Threat analysis and distributed denial of service (ddos) attack recognition in the internet of things (IoT). Electronics 2022;11(3).

[13] Odusami M, Misra S, Adetiba E, Abayomi-Alli O, Damasevicius R, Ahuja R. An improved model for alleviating layer seven distributed denial of service intrusion on webserver. J Phys: Conf Ser 2019;1235.

[14] Alharbi A, Alosaimi W, Alyami H, Rauf HT, Damasevicius R. Botnet attack detection using local global best bat algorithm for industrial internet of things. Electronics 2021;10(11).

[15] Kure HI, Islam S, Mouratidis H. An integrated cyber security risk management framework and risk predication for the critical infrastructure protection. Neural Comput Appl 2022;34(18):15241–71.

[16] Li H, Guo Y, Sun P, Wang Y, Huo S. An optimal defensive deception framework for the container-based cloud with deep reinforcement learning. IET Inf Secur 2022;16(3):178–92.

[17] Abdullahi M, Baashar Y, Alhussian H, Alwadain A, Aziz N, Capretz LF, et al. Detecting cybersecurity attacks in internet of things using artificial intelligence methods: A systematic literature review. Electronics 2022;11(2).

[18] Kaur R, Gabrijelčič D, Klobučar T. Artificial intelligence for cybersecurity: Literature review and future research directions. Inf Fusion 2023;97.

[19] Khan AR, Kashif M, Jhaveri RH, Raut R, Saba T, Bahaj SA. Deep learning for intrusion detection and security of internet of things (IoT): Current analysis, challenges, and possible solutions. Secur Commun Netw 2022;2022.

[20] Macas M, Wu C, Fuertes W. A survey on deep learning for cybersecurity: Progress, challenges, and opportunities. Comput Netw 2022;212.

[21] Capuano B, Johnson M. Real-world applicability of AI in cybersecurity. Cybersecur J 2022;15(2):112–25. http://dx.doi.org/10.1093/cybsec/xyz012.

[22] Damasevicius R, Toldinas J, Venckauskas A, Grigaliunas S, Morkevicius N. Technical threat intelligence analytics: What and how to visualize for analytic process. In: 2020 24th international conference electronics, eLECTRONICS 2020. 2020.

[23] Narayanan S, Ganesan A, Joshi K, Oates T, Joshi A, Finin TW. Early detection of cybersecurity threats using collaborative cognition. In: 2018 IEEE conference on cognitive and computational aspects of situation management (CogSIMA). IEEE; 2018, p. 1–7. http://dx.doi.org/10.1109/CIC.2018.00054.
[24] Toldinas J, Venckauskas A, Damasevicius R, Grigaliunas S, Morkevicius N, Baranauskas E. A novel approach for network intrusion detection using multistage deep learning image recognition. Electronics 2021;10(15).
[25] Alzaqebah A, Aljarah I, Al-Kadi O, Damasevicius R. A modified grey wolf optimization algorithm for an intrusion detection system. Mathematics 2022;10(6).
[26] Nisioti A, Loukas G, Laszka A, Panaousis E. Data-driven decision support for optimizing cyber forensic investigations. IEEE Trans Inf Forensics Secur 2021;16:2397–412.
[27] Nwakanma CI, Ahakonye LAC, Njoku JN, Odirichukwu JC, Okolie SA, Uzondu C, et al. Explainable artificial intelligence (XAI) for intrusion detection and mitigation in intelligent connected vehicles: A review. Appl Sci 2023;13(3).
[28] Marinho R, Holanda R. Automated emerging cyber threat identification and profiling based on natural language processing. IEEE Access 2023;1.
[29] Kumar A, Dhabliya D, Agarwal P, Aneja N, Dadheech P, Jamal SS, et al. Cyber-internet security framework to conquer energy-related attacks on the internet of things with machine learning techniques. Comput Intell Neurosci 2022;2022.
[30] Karn RR, Kudva P, Elfadel IM. Learning without forgetting: A new framework for network cyber security threat detection. IEEE Access 2021;9:137042–62.
[31] Abioye TE, Arogundade OT, Misra S, Adesemowo K, Damasevicius R. Cloud-based business process security risk management: A systematic review, taxonomy, and future directions. Computers 2021;10(12).
[32] Damasevicius R, Toldinas J, Venckauskas A, Grigaliūnas S, Morkevicius N, Jukavičius V. Visual analytics for cyber security domain: State-of-the-art and challenges. Communications in computer and information science, vol. 1078 CCIS, 2019, p. 256–70.
[33] Grigaliunas S, Toldinas J, Venckauskas A, Morkevicius N, Damasevicius R. Digital evidence object model for situation awareness and decision making in digital forensics investigation. IEEE Intell Syst 2021;36(5):39–48.
[34] Nunes E, Diab A, Gunn AT, Marin E, Mishra V, Paliath V, et al. Darknet and deepnet mining for proactive cybersecurity threat intelligence. In: 2016 IEEE conference on intelligence and security informatics. ISI, IEEE; 2016, p. 7–12. http://dx.doi.org/10.1109/ISI.2016.7745435.
[35] Riesco R, Larriva-Novo X, Villagrá V. Cybersecurity threat intelligence knowledge exchange based on blockchain. Telecommun Syst 2019;72(3):409–28. http://dx.doi.org/10.1007/s11235-019-00613-4.
[36] Riesco R, Villagr'a V. Leveraging cyber threat intelligence for a dynamic risk framework. Comput Stand Interfaces 2019;66:103349. http://dx.doi.org/10.1007/s10207-019-00433-2.
[37] Ali H, Ahmad J, Jaroucheh Z, Papadopoulos P, Pitropakis N, Lo O, et al. Trusted threat intelligence sharing in practice and performance benchmarking through the hyperledger fabric platform. Entropy 2022;24(10).
[38] Stottlemyre SA. HUMINT, OSINT, or something new? Defining crowdsourced intelligence. Int J Intell Counter Intell 2015;28(3):578–89.
[39] Ring T. Threat intelligence: why people don't share. Comput Fraud Secur 2014;2014(3):5–9.
[40] Tounsi W, Rais H. A survey on technical threat intelligence in the age of sophisticated cyber attacks. Comput Secur 2018;72:212–33.
[41] Basheer R, Alkhatib B. Threats from the dark: A review over dark web investigation research for cyber threat intelligence. J Comput Netw Commun 2021;2021:1–21.
[42] Wazid M, Das AK, Mohd N, Park Y. Healthcare 5.0 security framework: Applications, issues and future research directions. IEEE Access 2022;10:129429–42.
[43] Yang J, Hu J, Yu T. Federated AI-enabled in-vehicle network intrusion detection for internet of vehicles †. Electronics 2022;11(22).
[44] Zhou Y, Tang Y, Yi M, Xi C, Lu H. CTI view: APT threat intelligence analysis system. Secur Commun Netw 2022;2022.
[45] Padmashree A, Krishnamoorthi M. Decision tree with pearson correlation-based recursive feature elimination model for attack detection in IoT environment. Inf Technol Control 2022;51(4):771–85.
[46] Damasevicius R, Venckauskas A, Grigaliunas S, Toldinas J, Morkevicius N, et al. LITNET-2020: An annotated real-world network flow dataset for network intrusion detection. Electronics 2020;9(5):800. http://dx.doi.org/10.3390/electronics9050800.
[47] Alazab M. Analysis on smartphone devices for detection and prevention of malware. Deakin University; 2014.
[48] Kent JT. Information gain and a general measure of correlation. Biometrika 1983;70(1):163–73. http://dx.doi.org/10.1093/biomet/70.1.163.
[49] Bicici UC, Akarun L. Multi-path routing for conditional information gain trellis using cross-entropy search and reinforcement learning. IEEE Access 2024.
[50] Powers DM. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 2020, arXiv preprint arXiv:2010.16061.
[51] Mustard S. The NIST cybersecurity framework. InTech 2014;61(1–2).
[52] Jayawickrama W. Managing critical information infrastructure security compliance: A standard based approach using ISO/IEC 17799 and 27001. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), 4277 LNCS - I, 2006, p. 565–74. http://dx.doi.org/10.1007/11915034_80.
[53] Gros S. A critical view on CIS controls. In: Proceedings of the 16th International Conference on Telecommunications, ConTEL 2021. 2021, p. 122–8. http://dx.doi.org/10.23919/ConTEL52528.2021.9495982.
[54] Al-Sa'eed MA, Al-Mahamid SM, Al-Sayyed RM. The impact of control objectives of information and related technology (COBIT) domain on information criteria and information technology resources. J Theoret Appl Inf Technol 2012;45(1):9–18.
[55] MITRE Corporation. MITRE ATT&CK framework. 2021, URL https://attack.mitre.org/.
[56] Symantec Corporation. Traditional signature-based systems. Symantec Security Blog 2020. URL https://symantec.com/security-blog.