الجـــمـهـوريـــة الجـزائـريـة الديمـقـراطيـة الـشـعـبـيـة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التـــعـلـيـــم العـــــالي والبـــحث العـــــلـــمـــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جـــــــــــامعة سعيدة – د. الطاهر مولاي –

Université Saïda – Dr. Tahar Moulay –

Faculté des Mathématiques, Informatique et Télécommunications

MEMOIRE

Présenté pour l'obtention du **Diplôme** de **MASTER en Télécommunications**

Spécialité : Réseaux et Télécommunications

Par : M. SEHMI Abdelhak et M. YAGOUBI Oussama

A Bio-Inspired Feature Selection Method for Optimized Intrusion Detection in Smart Home IoT Networks

---

Soutenu, le 17 /06/ 2025, devant le jury composé de :

M. BOUYEDDOU Benamar	M.C.A	Président
M. GUENDOUZ Mohamed	M.C.A	Rapporteur
M. MOKADEM Djelloul	M.C.B	Examinateur

2024 / 2025

Acknowledgement

Praise be to Allah, by whose grace good deeds we are completed, and by whose guidance this work has been accomplished.

We would like to express our sincere gratitude to our supervisor, Dr. GUENDOUZ Mohamed, for his valuable guidance, support and encouragement throughout the preparation of this thesis.

We would also like to thank our colleagues and friends who have been a source of support and motivation.

Finally, my deepest gratitude goes to my dear family for their constant love, prayers, and encouragement. After Allah, they are the reason behind every achievement in my life.(YAGOUBI)

Abdelhak ***SEHMI &*** Oussama **YAGOUBI**

🎓 مرت وكأنها بدات بالامس

اكتملت مسيرتنا الدراسيه

" اللهم اجعلها نهاية خير لبدايه طريق جديد"

الحمد لله الذي هيأ البدايه ويسر الطريق وطيب المنتهي

الحمد لله علي لذه الوصول الحمد لله الذي ماتم جهد ولا ختم سعي الا بفضله

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

{ رب أوزعني أن أشكر نعمتك الَّتِي أَنْعَمْتَ عَلَيَّ وَعَلَى وَالِدَيَّ وَأَنْ أَعْمَلَ صَالِحًا تَرْضَاهُ وَأَصْلِحْ لِي فِي ذُرِّيَّتِي إِنِّي تُبْتُ إِلَيْكَ وَإِنِّي مِنَ الْمُسْلِمِينَ }

صدق الله العظيم

🤍 الحمد لله حمدا كثيرا طيبا مباركا

Dedication

To myself

To my mom and dad so much more

My brothers and sisters

My friends

Everyone who helped me in any way

I express my gratitude…

Abdelhak Sehmi

Dedication

To those who were the light that guided my path, and whose prayers accompanied me every step of the way, I dedicate this work:

- **To my beloved mother**, the source of comfort, prayers, and unwavering support.
- **To my dear father**, who taught me patience and responsibility, and supported me with love.
- **To my brothers and my sister**, who shared this journey with their love, encouragement, and prayers.
- **To everyone who put a smile on my face** throughout my university years.
- **To my dear friends**, who were always there with support and companionship.

To all of you, I dedicate this humble effort with love, appreciation, and gratitude.

YAGOUBI Oussama

Abbreviations List

IoT	**I**nternet **o**f **T**hings
RFID	**R**adio **F**requency **I**dentification
AI	**A**rtificial **I**ntelligence
ML	**M**achine **L**earning
MQTT	**M**essage **Q**ueueing **T**elemetry **T**ransport
M2M	**M**achine **T**o **M**achine
HTTP	**H**ypertext **T**ransfer **P**rotocol
CoAP	**C**onstrained **A**pplication **P**rotocol
UDP	**U**ser **D**atagram **P**rotocol
TCP	**T**ransmission **C**ontrol **P**rotocol
IP	**I**nternet **P**rotocol
6LoWPAN	**IP**v6 **o**ver **L**ow-**P**ower **W**ireless **P**ersonal **A**rea **N**etworks
WSN	**W**ireless **S**ensors **N**etwork
GUI	**G**raphical **U**ser **I**nterface
HDMI	**H**igh **D**efinition **M**ultimedia **I**nterface
SSH	**S**ecure **S**hell
FTP	**F**ile **T**ransfer **P**rotocol
IDS	**I**ntrusion **D**etection **S**ystems
PCI-DSS	**P**ayment **C**ard **I**ndustry **D**ata **S**ecurity **S**tandard
IPS	**I**ntrusion **P**revention **S**ystems
HIDS	**H**ost-based **I**ntrusion **D**etection **S**ystem
NIDS	**N**etwork-based **I**ntrusion **D**etection **S**ystem
SPAN	**S**witch **S**panning **P**ort
VLAN	**V**irtual **L**ocal **A**rea **N**etworks
RSPAN	**R**emote **SPAN**
DoS	**D**enial-of-**S**ervice
SYN	**S**ynchronize
ICMP	**I**nternet **C**ontrol **M**essage **P**rotocol
SIEM	**S**ecurity **I**nformation and **E**vent **M**anagement
SVM	**S**upport **V**ector **M**achines
SVR	**S**upport **V**ector **R**egression
PCA	**P**rincipal **C**omponent **A**nalysis

GAN	**G**enerative **A**dversarial **N**etwork
ANN	**A**rtificial **N**eural **N**etwork
GA	**G**enetic **A**lgorithm
PSO	**P**article **S**warm **O**ptimization
Pbest	**P**ersonal **B**est
Gbest	**G**lobal **B**est
ACO	**A**nt **C**olony **O**ptimization
FA	**F**irefly **A**lgorithm
BFO	**B**acterial **F**oraging **O**ptimization
BA	**B**at **A**lgorithm
CNN	**C**onvolutional **N**eural **N**etwork
SI	**S**warm **I**ntelligence
NSL	**N**ational **S**cience **L**ab
KDD	**K**nowledge **D**iscovery in **D**atabases
MQTT	**M**essage **Q**ueuing **T**elemetry **T**ransport
CO-Gas	carbon monoxide gas
BPSO	**B**inary **P**article **S**warm **O**ptimization
SHA-256	**S**ecure **H**ash **A**lgorithm 256-bit
VS Code	**V**isual **S**tudio **C**ode
I/O	**I**nput/**O**utput
KNN	**K**-**N**earest **N**eighbor
LR	**L**inear **R**egression
SVM	**S**upport **V**ector **M**achine
NB	**N**aive **B**ayes
MLP	**M**ulti **L**ayer **P**erceptron
DT	**D**ecision **T**ree
RF	**R**andom **F**orest

List of figures

List of Tables

CHAPTER I: INTERNET OF THINGS (IoT)

CHAPTER II: Intrusion detection systems (IDSs)

CHAPTER III: Machine Learning (ML)

CHAPTER IV: Particle Swarm Optimization (PSO)

General Introduction

General Introduction

The Internet of Things (IoT) has rapidly expanded, connecting a wide range of devices-from everyday home appliances to complex industrial machinery-through embedded sensors, software, and network connectivity. This interconnected ecosystem generates massive volumes of real-time data, enabling enhanced operational efficiency, automation, and data-driven decision-making across various sectors such as manufacturing, smart cities, and healthcare. For example, IoT sensors facilitate predictive maintenance by monitoring equipment health and alerting operators before failures occur, significantly reducing downtime and costs.

However, the vast scale and heterogeneity of IoT devices introduce significant security challenges. Many devices have limited built-in security, and the sheer volume of data and network traffic makes manual monitoring infeasible. Intrusion Detection Systems (IDS) are therefore critical for continuously monitoring IoT networks to identify suspicious or malicious activities that could threaten system integrity. Traditional IDS approaches often struggle to handle the dynamic, large-scale, and resource-constrained nature of IoT environments.

Machine Learning (ML) offers a powerful solution by enabling IDS to automatically learn from vast amounts of IoT-generated data and detect complex attack patterns or anomalies without relying solely on predefined signatures. ML models can adapt to evolving threats and improve detection accuracy, making them well-suited for IoT security applications. However, ML models require careful tuning and optimization to perform effectively in diverse IoT contexts.

Particle Swarm Optimization (PSO), a nature-inspired algorithm modeled on the social behavior of bird flocks or fish schools, is frequently used to optimize ML model parameters and feature selection in IDS frameworks. PSO efficiently explores high-dimensional search spaces to find optimal or near-optimal configurations, enhancing the accuracy and efficiency of ML-based intrusion detection in IoT networks.

In summary, the explosive growth of IoT demands advanced security mechanisms. Integrating IDS with ML techniques, further optimized by algorithms like PSO, creates intelligent, adaptive defense systems capable of protecting complex IoT ecosystems while harnessing their data for operational excellence.

To provide a clear understanding of the research topic, the thesis is organized into four main chapters. The first chapter introduces the concept of the Internet of Things (IoT), its architecture, applications, IoT characteristics, and IoT communication protocols. The second chapter discusses Intrusion Detection Systems (IDS), focusing on IDS architectures, their characteristics, and various types of intrusion attacks. The third chapter focuses on the use of Machine Learning (ML) techniques in the context of IDS, highlighting common algorithms and their effectiveness in detecting intrusions. Finally, the fourth chapter presents the Particle Swarm Optimization (PSO) algorithm, explaining its application in feature selection to enhance the performance and accuracy of ML-based intrusion detection models.

Chapter I

IoT (Internet of Things)

I.1 Introduction

The next wave in the erea of computing will be outside the realm of the traditional desktop. In the internet of things (IoT) paradigm, many of the objects that surround us will be on the network in one form or another. Radio Frequency Identification (RFID) and sensor network technologies will rise to meet this new challenge, in which information and communication systems are invisibly embedded in the environment around us. This result in the generation of enormous amounts of data which have to be stored, processed and presented in a seamless, effcient and easily interpretable form. This model will consist of services that are commodities and delivred in a manner similar to traditional commodities [1].

I.2 Definition of IoT

The internet of things (IoT) is the inter-networking of physical device, vehicles, buildings, and other items embedded with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data. The (IoT) allows objects to be sensed or controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy and economic benefit in addition to reduced human intervention [2].

Internet of Things is a concept and a paradigm that considers pervasive presence in the environment of a variety of things that through wireless and wired connection and unique addressing schemes are able to interact with each other and cooperate with other things to create new applications and reach common goals [3].

The goal of the internet of tings is to enable things to be connected every time, everywhere, with anything and anyone ideally using any path/network and any service.

I.3 IoT network characteristics

A network in the Internet of Things (IoT) has specific characteristics that make it different from traditional networks. Here’s a detailed breakdown:

I.3.1 Interconnectivity

Interconnectivity is a fundamental characteristic of IoT networks, allowing devices to connect with each other and the global internet. This connectivity enables devices to share data and coordinate actions across different locations. For instance, smart home devices can communicate

with each other to optimize energy consumption and security. Interconnectivity is facilitated by various wireless technologies such as Wi-Fi, Bluetooth, and cellular networks [4] [5].

I.3.2 Heterogeneity

IoT networks are heterogeneous, meaning they consist of devices with different hardware and communication protocols. This diversity includes devices ranging from simple sensors to complex industrial equipment, each with its own set of capabilities and limitations. Heterogeneity poses challenges in terms of interoperability but also allows for a wide range of applications across different industries [4] [5].

I.3.3 Dynamic Changes

IoT devices can change their state dynamically, such as switching between active and sleep modes to conserve energy. This dynamic behavior is crucial for managing power consumption, especially in battery-powered devices. Dynamic changes also include adapting to environmental conditions or responding to commands from central systems [4] [6].

I.3.4 Thing-related Services

Thing-related services are designed to provide functionalities that respect the constraints and capabilities of physical devices. These services consider factors such as device memory, processing power, and energy availability. By tailoring services to the specific needs of IoT devices, they can operate efficiently and effectively within their operational limits [4].

I.4 Types of IoT Technologies

IoT technologies include:

- Sensors and Actuators: Sensors collect data, while actuators perform actions based on that data [1][4].
- Communication Protocols: Such as Bluetooth, Zigbee, Wi-Fi, and cellular networks [1][3].
- Artificial Intelligence (AI) and Machine Learning (ML): Enhance data analysis and decision-making [1].
- Edge Computing: Enables local data processing to reduce latency and bandwidth usage [1].

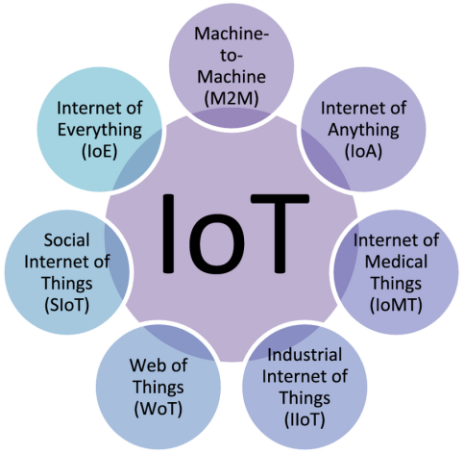

Fig I.1: Different types of IoT technologies

I.5 IoT Architecture

IoT architecture is a multi-layered framework that facilitates the interaction between physical devices and digital systems. It typically includes several layers, each with distinct responsibilities. Here's an overview of the common layers and some variations:

I.5.1 Common Layers of IoT Architecture

I.5.1.1 Perception/Sensor Layer (Device Layer)

- This layer consists of devices, sensors, and actuators that collect data from the environment and control physical objects [7][8][9].
- Technologies used include RFID tags, cameras, and various sensors [7].

I.5.1.2 Network Layer (Communications Layer)

- Responsible for transmitting data from devices to the cloud or other parts of the IoT system [7][8][9].
- Utilizes protocols like Wi-Fi, Bluetooth, Zigbee, and cellular networks [8][10].

I.5.1.3 Middleware/Processing Layer

- Processes and analyzes the collected data, often using cloud computing or big data processing [9].
- This layer transforms raw data into useful information and manages devices [11].

I.5.1.4 Application Layer

- Provides services and applications based on the processed data, such as smart home automation or industrial monitoring [7] [9] [11].
- Acts as the interface between the IoT system and users [11].

I.5.1.5 Business Layer

- Focuses on managing the IoT system, including business models and user privacy.

I.5.2 Simplified Three-Tier Architecture

I.5.2.1 Devices (Sensors and Actuators)

- Collects data and interacts with the physical environment.

I.5.2.2 Edge Gateway

- Aggregates and preprocesses data before sending it to the cloud.

I.5.2.3 Cloud

- Stores and analyzes data using advanced analytics tools.

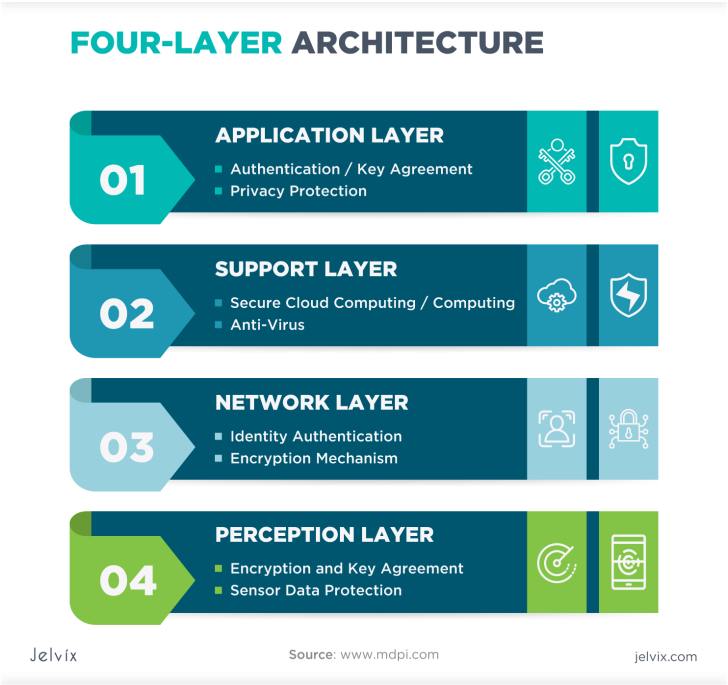

Fig I.2: IoT Architecture

I.6 IoT protocols

IoT protocols are essential for enabling communication between devices in the Internet of Things ecosystem. These protocols ensure that data is transmitted efficiently and securely across different layers of the IoT architecture. Here's a detailed overview of some key IoT protocols:

I.6.1 Application Layer Protocols

I.6.1.1 MQTT (Message Queueing Telemetry Transport)

- Description: MQTT is a lightweight messaging protocol designed for IoT and M2M applications. It uses a publish-subscribe architecture, making it ideal for remote environments with limited bandwidth.
- Use Cases: Predictive maintenance, smart home automation.

I.6.1.2 HTTP (Hypertext Transfer Protocol)

- Description: Widely used for web applications, HTTP is also used in IoT for data transfer via REST APIs. However, it requires more bandwidth and energy compared to MQTT.
- Use Cases: Applications with fewer data and battery constraints.

I.6.1.3 CoAP (Constrained Application Protocol)

- Description: Designed for constrained networks and devices, CoAP is similar to HTTP but uses UDP for faster communication. It supports RESTful architecture and is suitable for low-power devices.
- Use Cases: Resource-constrained IoT environments.

I.6.2 Transport Layer Protocols

I.6.2.1 TCP (Transmission Control Protocol)

- Description: Ensures reliable data transfer by reassembling packets and resending lost data. It prioritizes accuracy over speed.
- Use Cases: Applications requiring guaranteed data delivery.

I.6.2.2 UDP (User Datagram Protocol)

- Description: Prioritizes speed over reliability, making it suitable for real-time applications like video streaming.
- Use Cases: Time-sensitive applications.

I.6.3 Network Layer Protocols

I.6.3.1 IP (Internet Protocol)

- Description: Essential for routing data packets across networks. Both IPv4 and IPv6 are used in IoT.
- Use Cases: General internet connectivity.

I.6.3.2 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks)

- Description: Optimized for low-power devices, enabling IPv6 communication over low-power networks.
- Use Cases: Low-power IoT devices.

I.6.4 Wireless Communication Protocols

I.6.4.1 Bluetooth

- Description: Suitable for short-range, low-power applications. Often used in personal area networks.
- Use Cases: Wearables, smart home devices.

I.6.4.2 Zigbee

- Description: Used for low-power, low-data-rate wireless communication. Common in-home automation.
- Use Cases: Smart lighting, thermostats.

I.6.4.3 Wi-Fi

- Description: Provides high-speed internet connectivity. Widely used in IoT applications requiring fast data transfer.
- Use Cases: Smart home devices, industrial automation.

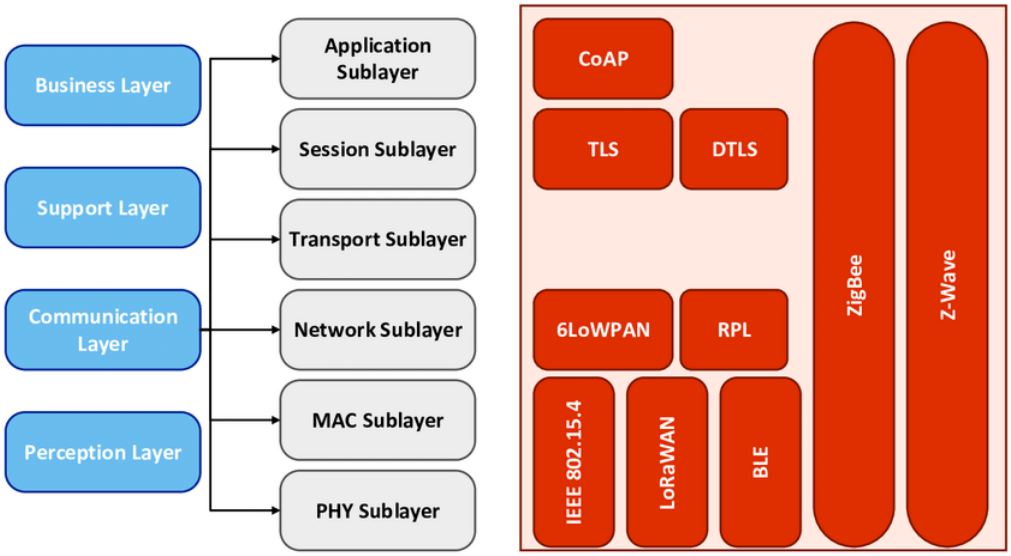

Fig I.3: IoT Communication Protocols

I.7 IoT applications

The Internet of things applications are addressing the societal needs and the advancements to enabling technologies such as nanoelectronics and cyber-physical systems continue to be challenged by a variety of technical.

I.7.1 Smart Cities

By 2020 we will see the development of Mega city corridors and networked integrated and branded cities. With more than 60 percent of the world population expected to live in urban cities by 2025. This will lead to the evolution of smart cities with eight smart features, including smart economy, smart buildings, smart mobility, smart energy, smart information communication and technology, smart planning and smart citizen. There will be about 40 smart cities globally by 2025.

The role of the smart cities will be crucial for Internet of Things (IoT) deployment, running of the day-to-day city operations and creation of city development strategies will drive the use of the (IoT). Therefore, cities and their service represent an almost ideal platform for (IoT) research, taking into account city requirements and transferring them to solutions enabled by (IoT) technology.

The figure depicts several commons actions that may take place in the smart day, highlighting in each occasion which domain applies [3].

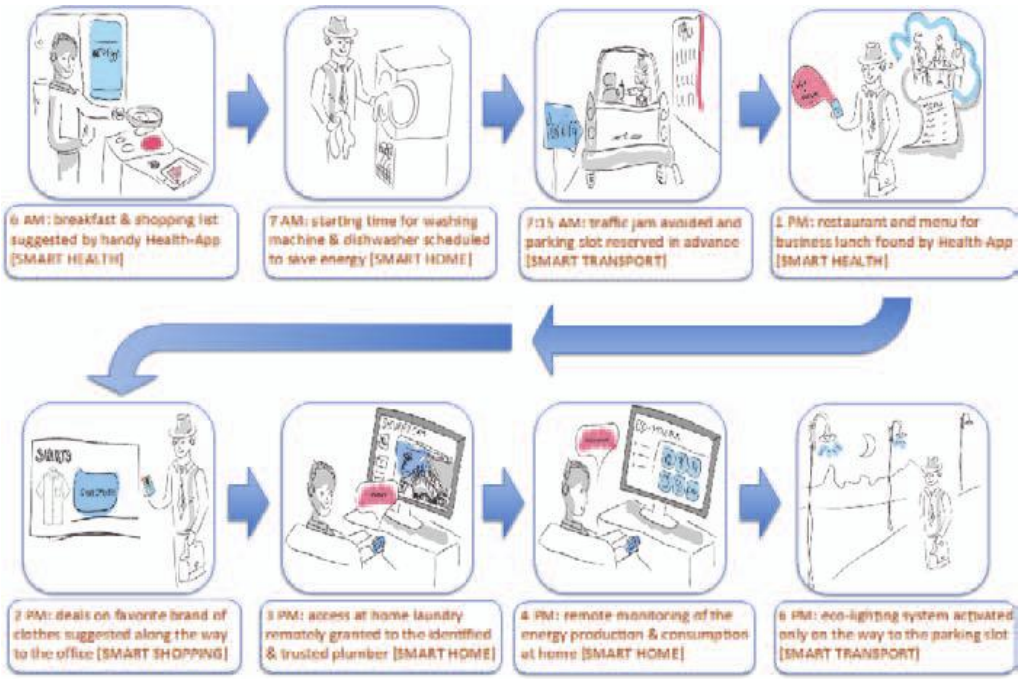

Fig I.4: A day in the life of a typical European citizen of a smart city

I.7.2 Smart Energy and the Smart Grid

There is increasing public awareness about the changing paradigm of our pol-icy in energy supply, consumption and infrastructure. For several reasons our future energy supply should no longer be based on fossil resources.

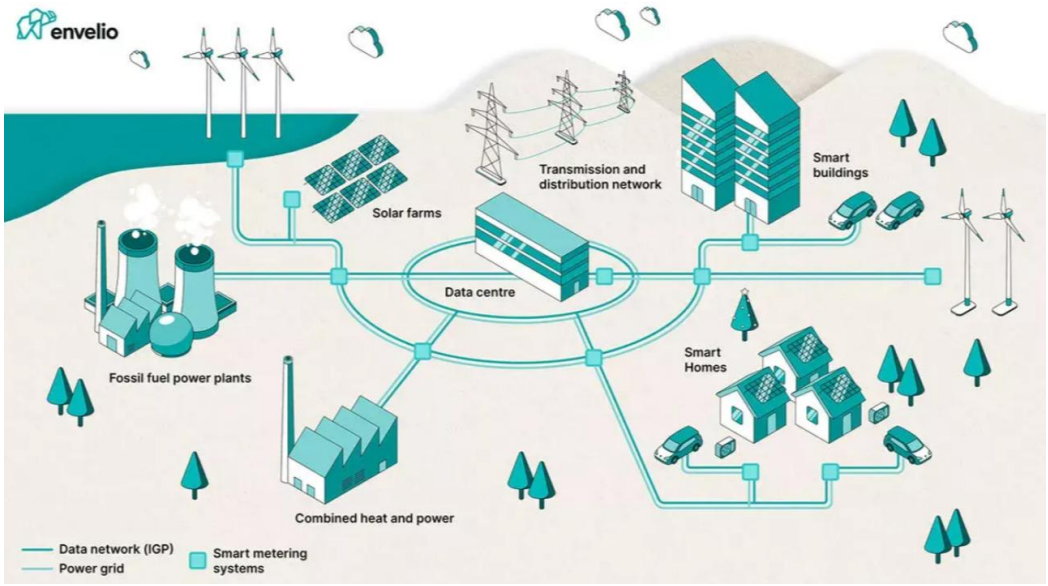

Fig I.5: Smart grid representation

I.7.3 Smart Home

The rise of Wi-Fi is role in home automation has primarily come about due to the networked nature of deployed electronics where electronic devices (mobile device, AV receivers ...etc).

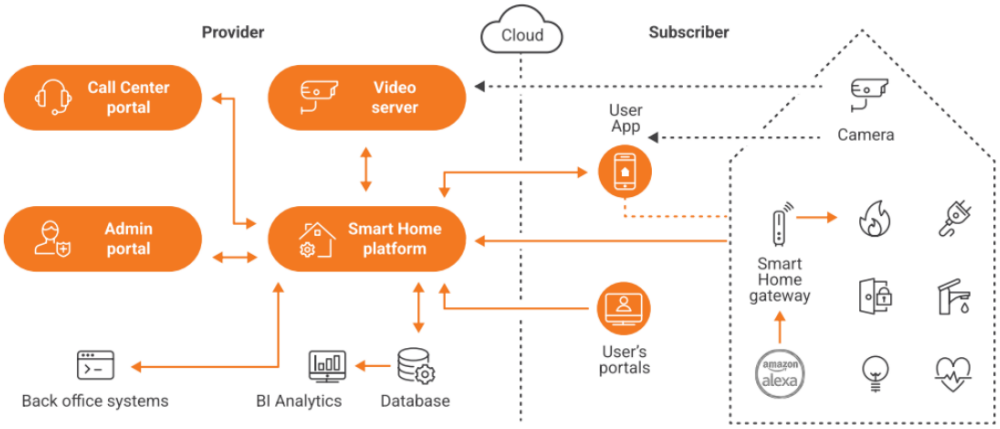

Fig I.6: Smart home platform

I.8 Networks and Communication

Present communication technologies span the globe in wireless and wired networks and support global communication by globally-accepted communication standards. The evolution and pervasiveness of present communication technologies has the potential to grow to unprecedented levels in the near future by including the world of things into the developing Internet of Things.

I.8.1 Wireless Networks

Wireless networks especially will grow largely by adding vast amounts of small Internet of Things devices with minimum hardware, software and intelligence, limiting their resilience to any imperfections in all their functions. Based on the research of the growing network complexity, caused by the Internet of Things, predictions of traffic and load models will have to guide further research on unfolding the predicted complexity to real networks, their standards and on-going implementations. The idea of internet of things (IoT) was developed in parallel to WSNs.

I.8.2 Wireless Sensors Networks (WSN)

A wireless sensor network (WSN) is a network formed by a large number of sensor nodes where each node is equipped with a sensor to detect physical phenomena such as light, heat, pressure, etc. WSNs are regarded as a revolutionary information gathering method to build the information

and communication system which will greatly improve the reliability and efficiency of infrastructure systems. Compared with the wired solution, WSNs feature easier deployment and better flexibility of devices. With the rapid technological development of sensors, WSNs will become the key technology for IoT. (Book: Internet of things: wireless sensor networks) [4].

I.9 Advantages of IoT

- Execute multiple tasks at a time like a computer.
- Easiest internet connectivity
- Works on GUI (Graphical User Interface) mode because of HDMI port.
- Best suited for server-based applications i.e., can be connected via SSH–Secure Shell-to access the Rpi command line remotely and file sharing via FTP–File Transfer Protocol.
- More reliable for software applications.

I.10 Disadvantages of IoT

- Security concerns and potential for hacking or data breaches.
- Privacy issues related to the collection and use of personal data.
- Dependence on technology and potential for system failures.
- Limited standardization and interoperability among devices.
- Complexity and increased maintenance requirements.
- High initial investment costs.
- Limited battery life on some devices.
- Concerns about job displacement due to automation.
- Limited regulation and legal framework for IoT, which can lead to confusion and uncertainty.

I.11 Conclusion

The Internet of Things represents a profound technological shift that continues to transform industries, cities, homes, and daily life. By connecting the physical and digital worlds through networks of sensors, processors, and interfaces, IoT enables unprecedented levels of automation, insight, and efficiency.

While the potential benefits are immense ranging from energy conservation and improved urban living to enhanced industrial productivity and innovative consumer experiences significant challenges remain. Security vulnerabilities, interoperability obstacles, data management complexities, and other implementation hurdles must be addressed to realize the full potential of IoT technologies.

As we look toward the future, the continuing evolution of edge computing, AI integration, and standardization efforts promises to overcome many of these challenges. With projected market growth remaining strong through 2032, IoT will likely become increasingly embedded in our technological infrastructure, economic systems, and daily routines creating a more connected, efficient, and responsive world in the process.

Chapter II

Intrusion Detection Systems (IDSs)

II.1 Introduction

Intrusion detection systems (IDSs) are software or hardware systems that automate the process of monitoring the events occurring in a computer system or network, analysing them for signs of security problems. As network attacks have increased in number and severity over the past few years, intrusion detection systems have become a necessary addition to the security infrastructure of most organizations. This guidance document is intended as a primer in intrusion detection, developed for those who need to understand what security goals intrusion detection mechanisms serve, how to select and configure intrusion detection systems for their specific system and network environments, how to manage the output of intrusion detection systems, and how to integrate intrusion detection functions with the rest of the organizational security infrastructure. References to other information sources are also provided for the reader who requires specialized or more detailed advice on specific intrusion detection issues. [12] [13]

II.2 Definition of Intrusion Detection Systems (IDS)

An intrusion detection system (IDS) is a network security tool that monitors network traffic and devices for known malicious activity, suspicious activity or security policy violations.

An IDS can help accelerate and automate network threat detection by alerting security administrators to known or potential threats, or by sending alerts to a centralized security tool. A centralized security tool such as a security information and event management (SIEM) system can combine data from other sources to help security teams identify and respond to cyberthreats that might slip by other security measures.

IDSs can also support compliance efforts. Certain regulations, such as the Payment Card Industry Data Security Standard (PCI-DSS), require organizations to implement intrusion detection measures.

An IDS cannot stop security threats on its own. Today IDS capabilities are typically integrated with—or incorporated into—intrusion prevention systems (IPSs), which can detect security threats and automatically act to prevent them. [13] [14]

II.3 Types of Intrusion Detection Systems

II.3.1 Types of IDS by Deployment

II.3.1.1 Host-based IDS (HIDS)

Host-based Intrusion Detection System [HIDS] is a security software designed to monitor & analyse the activities on an individual host or endpoint to detect & respond to potential security breaches. It works by examining system logs, file integrity, user activities & network connections, aiming to identify suspicious behaviour or signs of unauthorised access or tampering.

HIDS works by deploying agents or sensors on individual hosts, continuously monitoring system events & activities. It compares these events with a database of known attack patterns & abnormal behaviours. If any anomalous activity is detected, the system generates alerts or notifications to administrators, allowing them to investigate & take necessary action. [15]

II.3.1.2 Network-based IDS (NIDS)

A Network-based Intrusion Detection System [NIDS] is a security solution designed to monitor & analyse network traffic for potential security breaches or malicious activities. It operates as a passive monitoring system, observing data packets passing through the network in real-time. NIDS helps identify & respond to various cyber threats, such as malware, unauthorised access attempts & suspicious patterns, to enhance overall network security.

NIDS inspects network packets using various techniques like signature-based & anomaly-based detection. It examines packet headers & payloads, comparing them against a database of known attack signatures. If a match is found, it raises an alert. Anomaly-based detection identifies deviations from normal network behaviour, flagging any unusual activities that might indicate a potential intrusion. NIDS can also prevent attacks by blocking malicious traffic or sending alerts to administrators. [16]

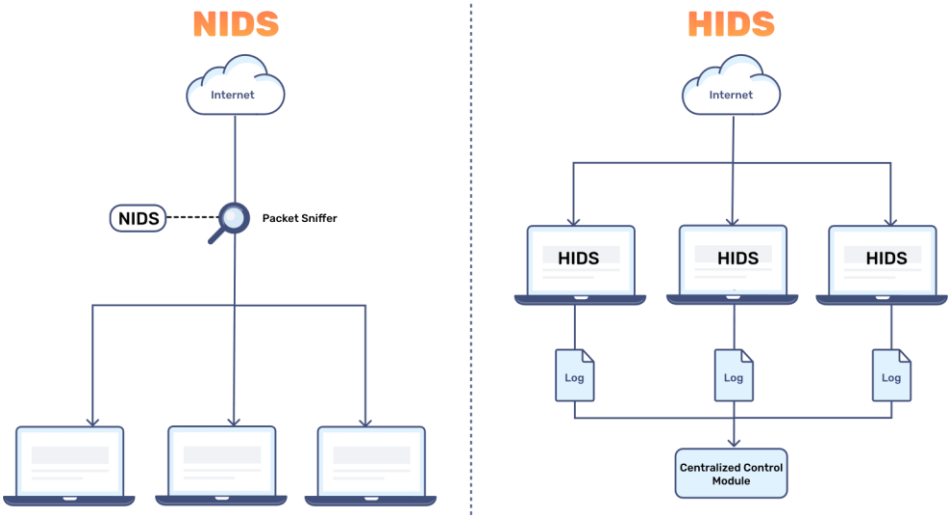

Fig II.1: Network-based IDS vs Host-based IDS

II.3.1.3 Hybrid IDS

A hybrid intrusion detection system combines both anomaly-based and signature-based detection methods to address the limitations of each approach. A hybrid system leverages signature-based detection for known threats and anomaly-based detection for novel attacks. This enhances the overall effectiveness of intrusion detection. [17]

II.3.2 Types of IDS by Detection Method

II.3.2.1 Anomaly-Based IDS

Anomaly-based IDS focuses on identifying deviations from normal behavior within a network or system. It works by establishing a baseline for normal activity by statistically analyzing network traffic or system activity over time. This baseline becomes a reference for identifying anomalies. The IDS then continuously monitors network traffic or system activity and compares the real-time data to the established baselines. Significant deviations from these baselines are flagged as potential intrusions. [17]

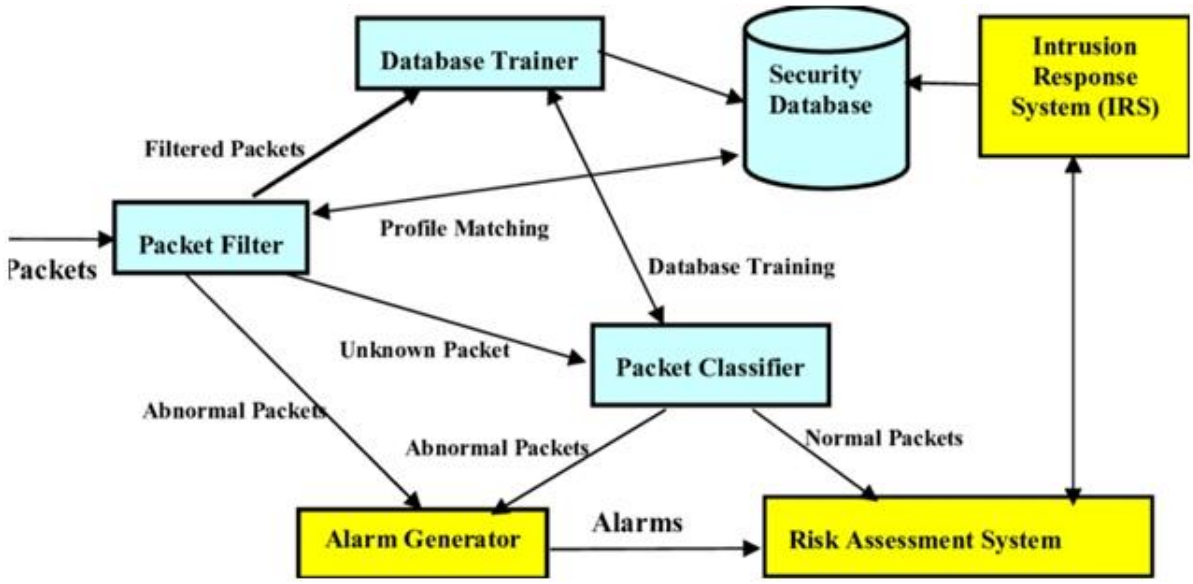

Fig II.2: Anomaly-Based IDS

II.3.2.2 Signature-Based IDS

A signature-based intrusion detection system relies on a predefined database of attack signatures to identify malicious activity. These signatures represent known patterns or fingerprints of network attacks or suspicious system behavior. The IDS continuously monitors network traffic or system activity and compares this data against the database of attack signatures. Any matches trigger an alert, indicating a potential intrusion attempt. [17]

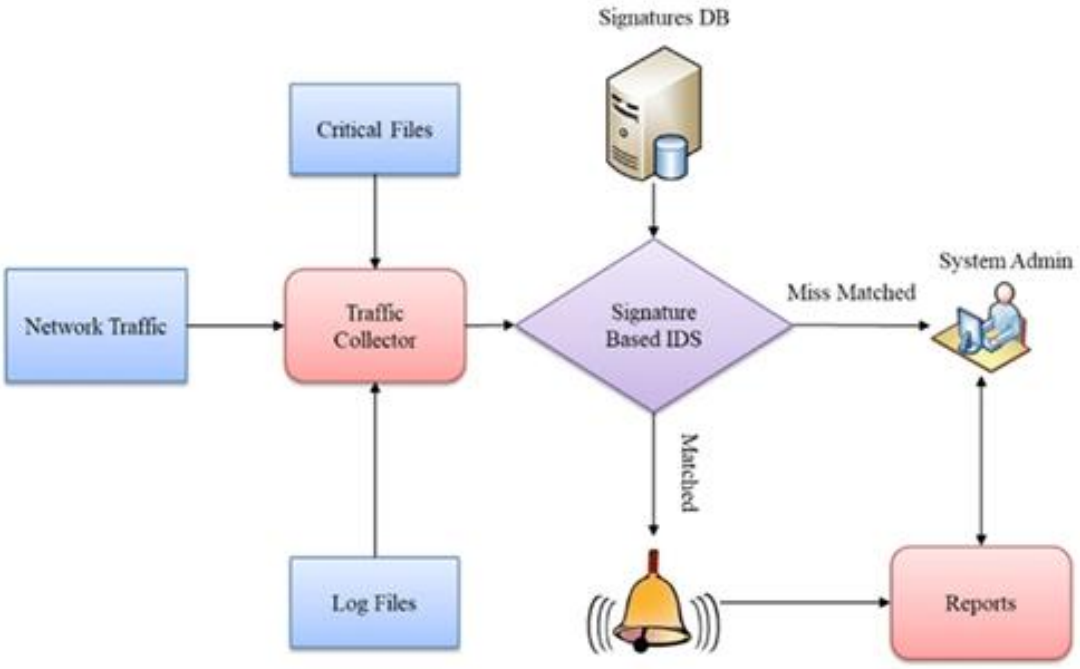

Fig II.3: Signature-Based IDS

II.4 Characteristics of an intrusion detection system

An intrusion detection system (IDS) is a tool that monitors network traffic for suspicious activity and known threats [18]. Here's a more detailed look at the characteristics of an intrusion detection system:

- Accuracy An IDS should be accurate, minimizing false positives (identifying harmless behavior as an attack) and false negatives (failing to detect actual attacks). An IDS that is accurate detects genuine attacks, while one that is precise does not report legitimate behavior as an attack. Accuracy is a main parameter in determining the performance of the algorithm used to analyze and predict intrusions.
- Timeliness An IDS must detect and report intrusions quickly. The faster an intrusion is detected, the quicker it can be addressed, thereby minimizing potential damage.
- Scalability An IDS needs to handle growth in traffic or nodes without a drop in performance. It should adapt to increasing demands, accommodate new technologies, adjust to network infrastructure changes, and expand coverage areas.
- Robustness An IDS should be resistant to attackers trying to disable or deceive it. A novel IDS architecture can improve robustness against adversarial attacks by combining conventional machine learning (ML) models and deep learning models.

- Configurability An IDS should be configurable to adapt to specific environments or requirements.
- Real-time Monitoring an IDS should monitor system and network activities in real-time to provide instant alerts.
- Logging and Audit an IDS should keep detailed logs for forensic analysis and compliance.
- Low Overhead an IDS should not significantly degrade the performance of the system it protects.
- Adaptability Advanced IDSs can learn and adapt to new threats, especially anomaly-based systems.

II.5 IDS Architectures

Intrusion Detection System (IDS) architectures define how IDS devices or software are connected within a network to capture and monitor traffic effectively. The fundamental requirement is that the IDS must have access to network traffic to analyze it for suspicious activity. There are three primary IDS architecture models:

II.5.1 Switch Spanning Port (SPAN)

- A SPAN port is a special port on a network switch configured to mirror traffic from one or more source ports or VLANs to a designated destination port where the IDS sensor is connected.
- This allows the IDS to passively monitor traffic without interfering with the flow, as it receives a copy of the data.
- SPAN ports support one-way traffic capture, meaning the IDS can observe traffic but cannot alter or block it.
- Limitations include a restricted number of SPAN sessions per switch (e.g., six sessions on Cisco Catalyst 6000 switches) and potential packet duplication when capturing both transmit and receive traffic, which can affect IDS signature processing.
- Over-subscription can occur if the mirrored traffic exceeds the destination port's capacity, potentially leading to packet loss on the IDS but not affecting the source ports.
- To handle high traffic volumes, multiple SPANs or Remote SPAN (RSPAN) can be used, and EtherChannel configurations may help aggregate traffic to the IDS, though this requires careful planning to avoid missing threats or delayed detection.

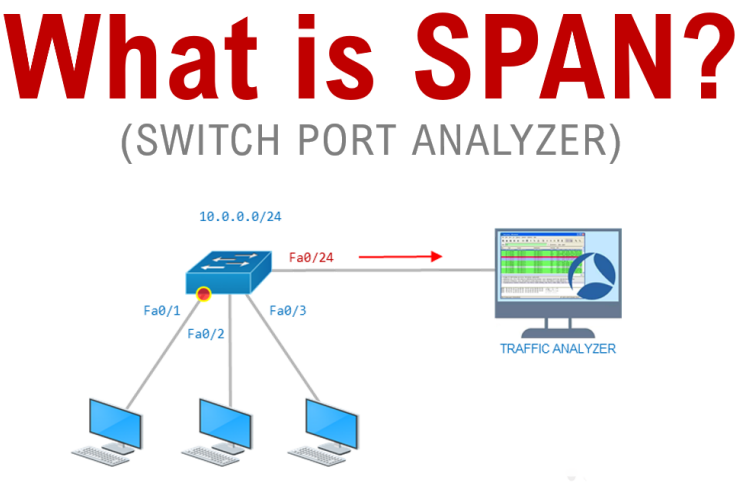

Fig II.4: Switch Spanning Port (SPAN)

II.5.2 Network Tap

- A network tap is a dedicated physical device inserted at a network bottleneck or uplink point to passively capture all traffic flowing through that segment.
- It provides a reliable and complete copy of network data in full duplex mode, allowing the IDS to see both sides of conversations without packet loss.
- Taps are passive and do not have an IP address, which enhances security by making the IDS invisible to attackers and preventing direct attacks on the IDS itself.
- They simplify deployment in switched networks where promiscuous monitoring is difficult.
- Network taps can be deployed at multiple points and aggregated at a central monitoring rack, often requiring load balancing among several IDS sensors to handle high-speed traffic.
- Taps reduce IDS implementation costs and improve security posture by ensuring comprehensive traffic visibility without impacting network performance.

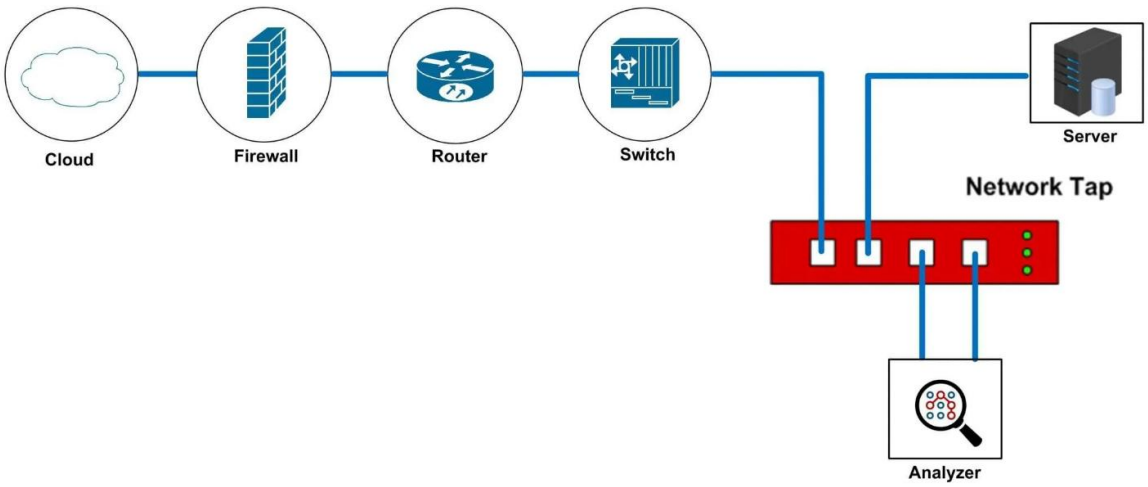

Fig II.5: Network TAPs

II.5.3 Inline Architecture

- In this model, the IDS (or Intrusion Prevention System, IPS) is placed physically inline between two network segments.
- All traffic passes through the IDS device, which can analyze and potentially block malicious traffic in real time.
- This allows active intervention, such as dropping malicious packets or terminating connections, unlike passive monitoring in SPAN or tap setups.
- Inline IDS/IPS devices must be highly robust and capable of processing large volumes of traffic with minimal latency to avoid network disruption.
- Inline placement is often used at network perimeters or critical chokepoints where immediate threat mitigation is required.
- Inline IDS differs from passive IDS in that it can control traffic flow, while passive IDS only monitors and alerts.

II.6 Intrusion attacks

Intrusion attacks represent a broad spectrum of malicious activities that an Intrusion Detection System (IDS) is designed to identify by analyzing network traffic or system behavior for suspicious patterns. Below are detailed descriptions of common types of intrusion attacks that IDS typically detects:

II.6.1 Scanning Attacks

- Attackers perform reconnaissance by probing networks to find vulnerabilities.
- This includes port scans, vulnerability scans, and network mapping attempts.
- IDS detects these by identifying unusual patterns of connection attempts or probes across multiple ports or hosts, which deviate from normal traffic behavior.

II.6.2 Denial-of-Service (DoS) Attacks

- DoS attacks aim to overwhelm a system with excessive traffic, making it unavailable to legitimate users.
- Common forms include SYN floods, UDP floods, and ICMP floods.
- IDS detects these attacks by recognizing traffic spikes, repeated connection attempts, or abnormal packet patterns that indicate flooding or resource exhaustion.

II.6.3 Social Engineering Attacks

- Although social engineering primarily targets human factors, IDS can detect indirect signs such as unusual access attempts, phishing emails triggering suspicious downloads, or anomalous data transfers.
- For example, IDS may flag abnormal login patterns or unexpected file downloads that could be linked to social engineering exploits.

II.6.4 Malware Attacks

- IDS identifies attempts to download, install, or communicate with malware by detecting known malware signatures or suspicious file transfers.
- Signature-based IDS matches traffic against databases of known malware patterns, while anomaly-based IDS detects unusual behaviors indicative of malware activity.

II.6.5 Exploit Attacks

- Exploits leverage software vulnerabilities to gain unauthorized access or control.
- IDS monitors network traffic for patterns associated with known exploits, such as attempts to execute buffer overflows, code injections, or remote code execution.
- Signature-based detection is effective for known exploits, while anomaly-based methods help detect zero-day or novel exploits.

II.6.6 Privilege Escalation

- Attackers attempt to gain higher privileges within a system to access sensitive resources.
- IDS can flag suspicious user activities, such as unauthorized access attempts to critical files or system areas, or unusual process executions that may indicate privilege escalation attempts.

II.6.7 Insider Threats

- These are attacks originating from trusted users misusing their access.
- IDS detects unusual activity patterns like unauthorized access attempts from legitimate accounts, abnormal data exfiltration, or deviations from typical user behavior.
- Detecting insider threats is challenging but critical, often relying on anomaly-based detection and behavioral analysis.

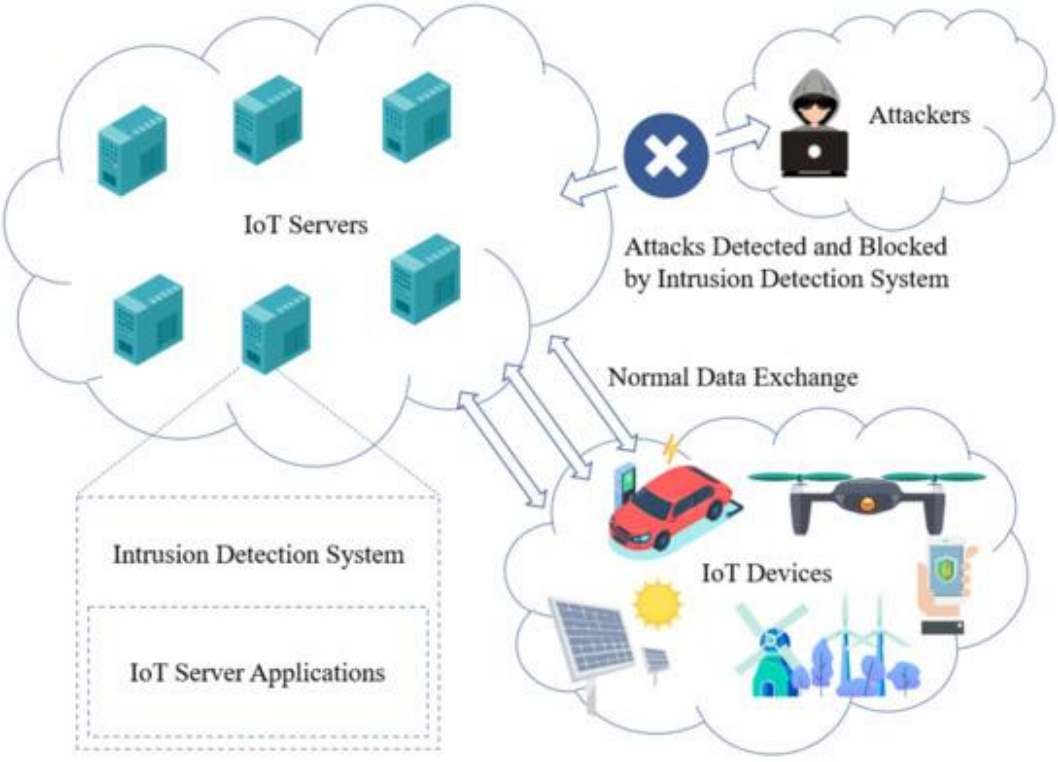

Fig II.6: Intrusion attacks

II.7 Conclusion

Intrusion Detection Systems (IDS) remain a cornerstone of modern cybersecurity, providing critical visibility into network and system activities to identify potential threats and suspicious behaviors. Despite inherent challenges such as false positives and evolving attack techniques, IDS technologies have matured to incorporate sophisticated detection methods—including signature-based and anomaly-based approaches—that enable timely alerts and support rapid incident response. By continuously monitoring traffic and system events, IDS helps organizations detect a wide range of attacks, from scanning and malware to insider threats, thereby enhancing overall security posture and aiding compliance efforts. However, IDS effectiveness depends heavily on proper configuration, ongoing tuning, and integration with complementary security tools like Intrusion Prevention Systems (IPS) and Security Information and Event Management (SIEM) platforms. As cyber threats grow more complex, IDS remains an indispensable, dynamic component of layered defense strategies.

With the increasing complexity and volume of cyber threats, traditional IDS methods face limitations in detecting novel or sophisticated attacks. This has led to the growing adoption of machine learning (ML) techniques in intrusion detection. ML enables systems to learn from historical data, recognize patterns, and identify anomalies without relying solely on predefined signatures. By leveraging algorithms such as supervised learning, unsupervised learning, and deep learning, ML-enhanced IDS can improve detection accuracy, reduce false positives, and adapt to emerging threats more effectively. The next chapter will explore the fundamentals of machine learning, its applications in cybersecurity, and how it transforms intrusion detection systems into more intelligent, proactive defenses.

Chapter III

Machine Learning (ML)

III.1 Introduction

Machine learning and bio-inspired algorithms are two interconnected fields that have revolutionized the way we approach complex computational problems. Machine learning, a subset of artificial intelligence, enables computers to learn from data and improve over time, while bio-inspired algorithms draw inspiration from natural systems to solve computational challenges. This overview will explore the definitions, types, and applications of both machine learning and bio-inspired algorithms, highlighting their synergies and potential applications.

III.2 Definition and Importance of Machine Learning

Machine learning is a branch of artificial intelligence that focuses on developing algorithms capable of learning from data without explicit programming. It is crucial in today's data-driven world, enabling computers to make decisions or predictions based on patterns found in data. The importance of machine learning lies in its ability to handle large volumes of data, drive innovation across sectors like healthcare and finance, and enable automation by performing tasks that were previously manual [19] [20].

III.3 Types of Machine Learning

Machine learning can be categorized into several types:

III.3.1 Supervised Learning

Supervised learning is the most common type of machine learning. It involves training algorithms on labeled data to predict outcomes. In this approach, the model learns a mapping between the input (features) and the output (label) during the training process. Once trained, the model can predict the output for new, unseen data.

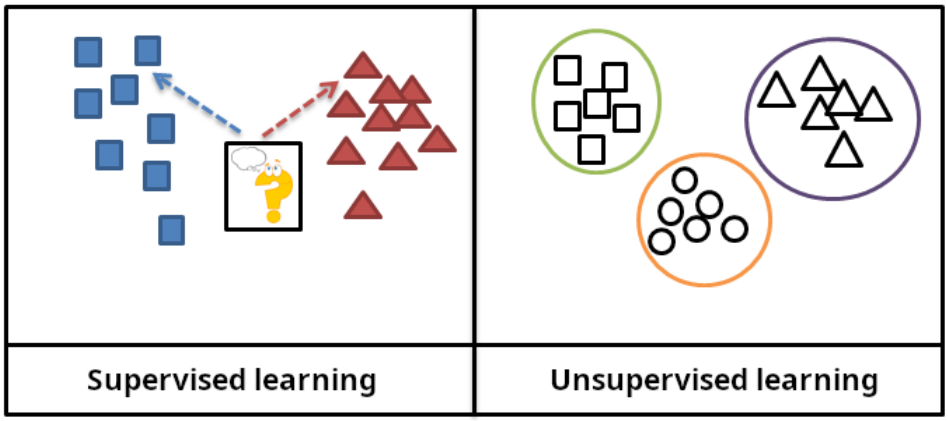

Fig III.1: Supervised and unsupervised learning

Examples of Supervised Learning Algorithms:

III.3.1.1 Linear Regression

Linear regression is a statistical method used in machine learning for predictive analysis. It models a linear relationship between one or more independent variables and a dependent variable. The goal is to find a best-fit line that minimizes the difference between observed and predicted values, typically using the least squares method.

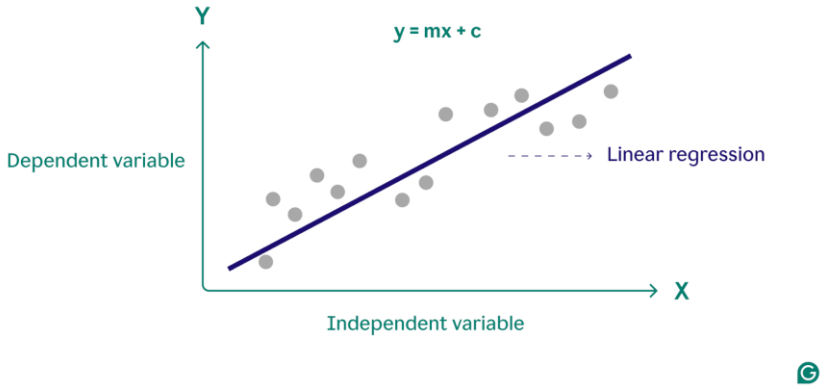

Fig III.2: Linear Regression in machine learning (ML)

III.3.1.2 Logistic Regression

Logistic regression is a supervised learning algorithm used for classification problems to predict binary outcomes. It models the probability of an event occurring based on one or more independent variables. Logistic regression is commonly used in scenarios where the outcome is binary, such as predicting whether a customer will buy a product or not. The model outputs a probability between 0 and 1, which can be converted into a binary class label.

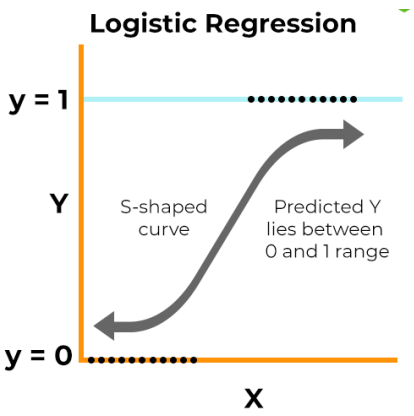

Fig III.3: Logistic Regression in machine learning (ML)

III.3.1.3 Decision Trees

Decision Trees are a type of supervised learning algorithm useful for both classification and regression tasks. They use a tree-like structure to make decisions based on input features. Each internal node represents a feature or attribute, each branch represents a decision or test, and each leaf node represents the predicted class or value. Decision Trees are easy to interpret and visualize but can suffer from overfitting if not properly regularized.

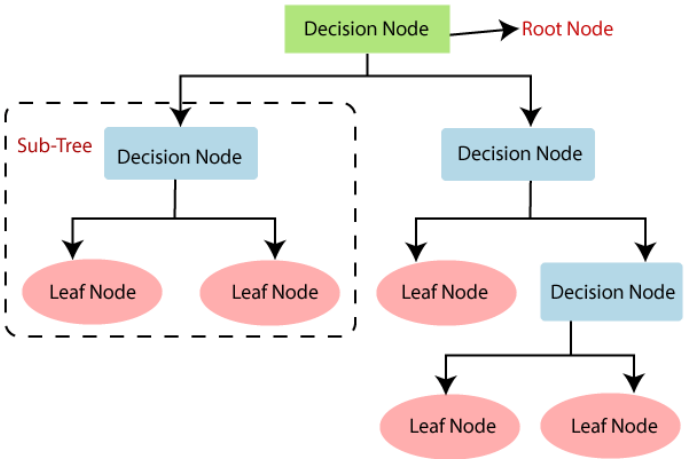

Fig III.4: Decision Tree Classification Algorithm

III.3.1.4 Support Vector Machines (SVM)

SVM is a supervised learning algorithm primarily used for classification tasks. It aims to find the optimal hyperplane that maximally separates classes in the feature space. SVM can handle high-dimensional data and is effective in cases where the number of features is large compared to the number of samples. SVMs can also be used for regression tasks, known as Support Vector Regression (SVR).

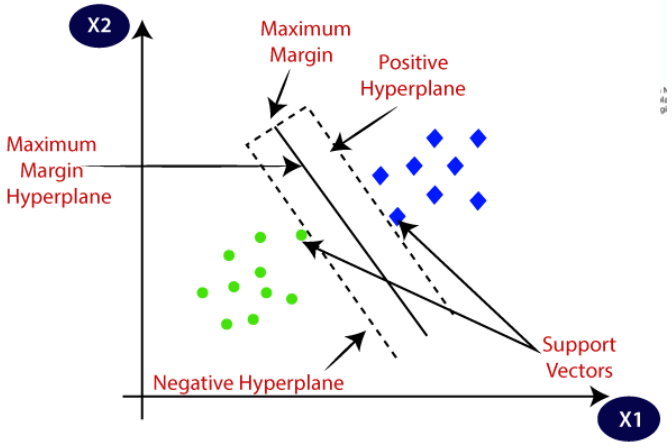

Fig III.5: Support Vector Machine Algorithm

III.3.2 Unsupervised Learning

Unsupervised learning involves training the model on an unlabeled dataset. The model is left to find patterns and relationships in the data on its own. This type of learning is often used for clustering and dimensionality reduction. Reinforcement learning: Uses rewards or penalties to guide the learning process.

Examples of Unsupervised Learning Algorithms:

- K-Means Clustering: Groups similar data points into clusters.
- Principal Component Analysis (PCA): Reduces the number of variables in a dataset while retaining most of the information.
- Association Rule Learning: Identifies typical relations between variables in a large dataset.

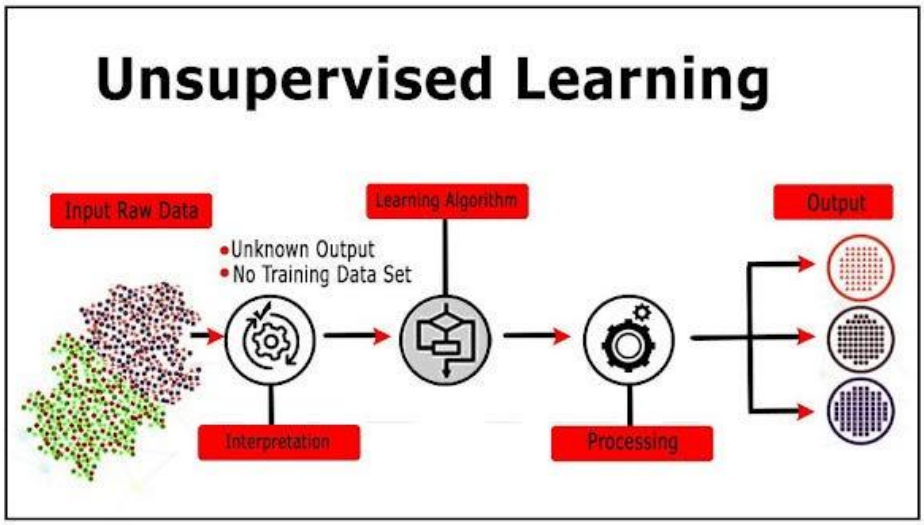

Fig III.6: Unsupervised Learning

III.3.3 Self-Supervised Learning

Self-supervised learning is a type of machine learning that learns from unlabeled data by generating its own labels. This approach is useful when labeled data is scarce or expensive to obtain. Self-supervised learning often involves techniques like autoencoders or generative adversarial networks (GANs) to learn representations of the data without explicit supervision.

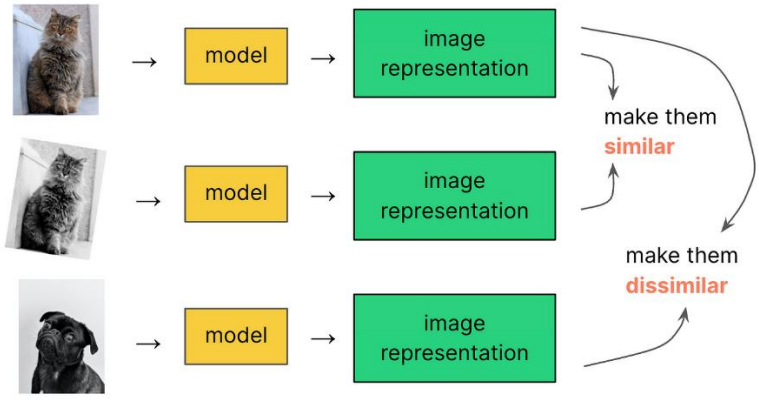

Fig III.7: Self-Supervised Learning

III.4 Machine Learning Algorithms

Some key machine learning algorithms include:

III.4.1 Linear Regression

Linear regression is a supervised machine learning algorithm used to model linear relationships between variables. It predicts a continuous output variable based on one or more input features. The model is represented by a linear equation, where each input variable is assigned a coefficient that reflects its impact on the output variable. The equation typically includes an intercept term, which shifts the line up or down. Linear regression is commonly used for forecasting and prediction tasks, such as predicting house prices based on features like size and location.

Key Features:

- Least Squares Method: The most common method for fitting linear regression models, which minimizes the sum of squared residuals between observed and predicted values.
- Interpretability: Coefficients provide insights into the relationship between variables, making it easy to understand how changes in input variables affect the output.

III.4.2 Decision Trees

Decision trees are a type of supervised learning algorithm that uses a tree-like structure to make decisions. They are particularly useful for classification and regression tasks. Each internal node represents a feature or attribute, each branch represents a decision or test, and

each leaf node represents the predicted class or value. Decision trees are easy to interpret and visualize but can suffer from overfitting if not properly regularized.

Key Features:

- Handling Categorical Variables: Decision trees can handle both numerical and categorical data without needing additional preprocessing.
- Handling Missing Values: They can handle missing data by using surrogate splits or other strategies.

III.4.3 Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to improve the accuracy and robustness of predictions. By averaging the predictions from multiple trees, Random Forest reduces overfitting and improves generalization. It is widely used for both classification and regression tasks due to its ability to handle high-dimensional data and its robustness against outliers.

Key Features:

- Ensemble Learning: Combines predictions from multiple decision trees to reduce variance and improve accuracy.
- Handling High-Dimensional Data: Effective in handling datasets with a large number of features.

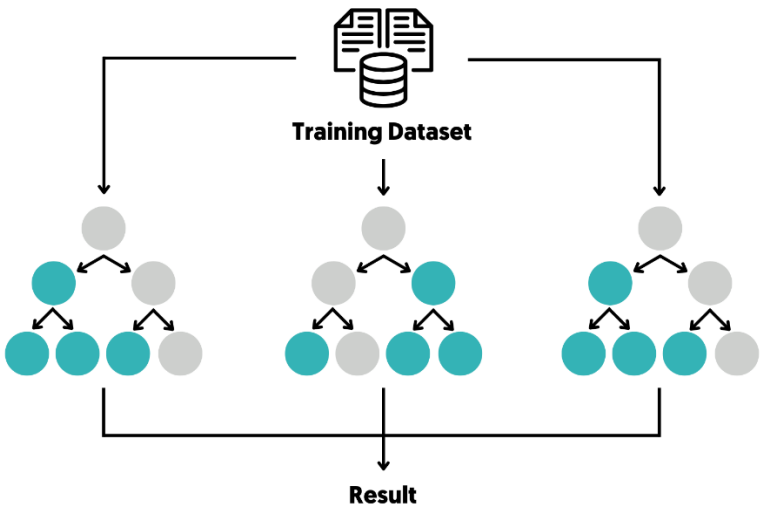

Fig III.8: Random Forest

III.4.4 Support Vector Machines (SVM)

SVM is a supervised learning algorithm primarily used for classification tasks. It aims to find the optimal hyperplane that maximally separates classes in the feature space. SVM can handle high-dimensional data and is effective in cases where the number of features is large compared to the number of samples. SVMs can also be used for regression tasks, known as Support Vector Regression (SVR).

Key Features:

- Maximal Margin: SVMs find the hyperplane that maximizes the margin between classes, which helps in achieving good generalization.
- Kernel Trick: Allows SVMs to handle non-linearly separable data by mapping it to a higher-dimensional space using kernels.

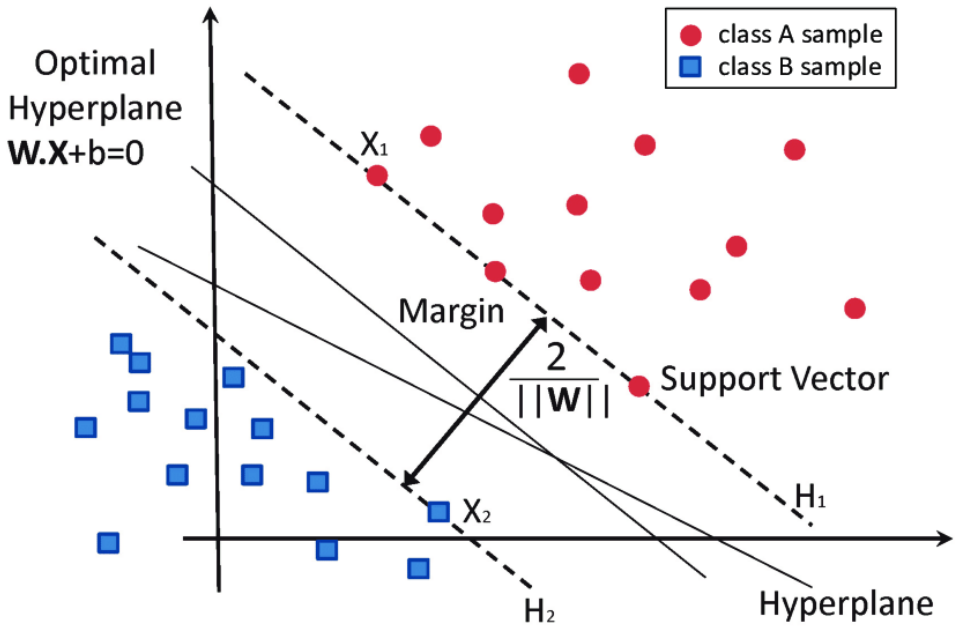

Fig III.9: Support Vector Machines (SVM)

III.4.5 Gradient Boosting

Gradient Boosting is an ensemble learning algorithm that combines multiple weak models to create a robust predictive model. It iteratively adds decision trees to correct the errors of previous models, with each new tree focusing on the residuals of the previous ensemble. Gradient Boosting is widely used for both classification and regression tasks due to its high accuracy and ability to handle complex interactions between variables.

Key Features:

- Iterative Improvement: Each new model is trained to correct the errors of the previous ensemble, leading to gradual improvement in predictions.
- Handling Complex Interactions: Effective in capturing non-linear relationships and interactions between variables.

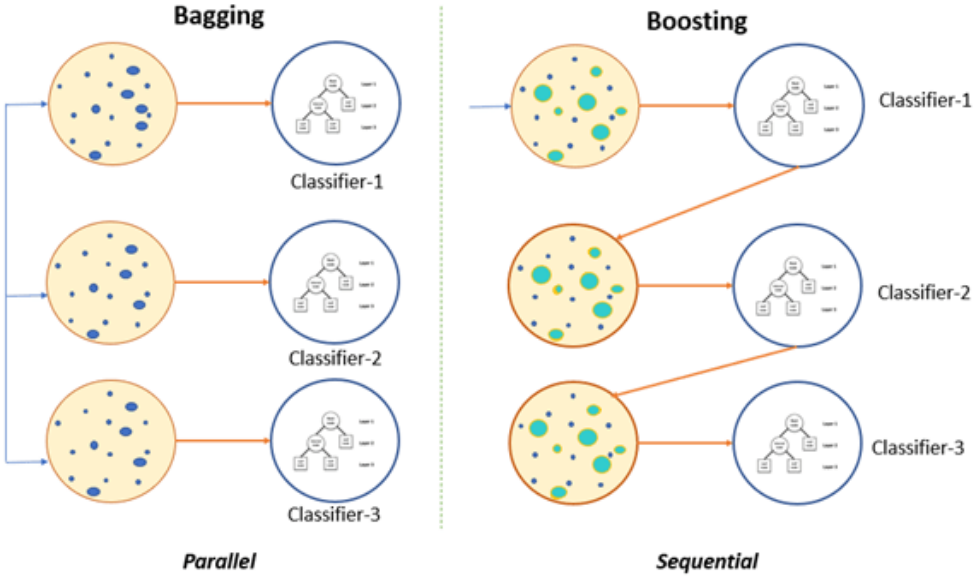

Fig III.10: Gradient Boosting

III.4.6 K-Means Clustering

K-Means is an unsupervised learning algorithm used for clustering data points into groups based on their similarity. It initializes centroids randomly and then iteratively updates these centroids to minimize the sum of squared distances between each data point and its closest centroid. K-Means is useful for discovering patterns or structures in unlabeled data.

Key Features:

- Unsupervised Learning: Does not require labeled data, making it useful for exploratory data analysis.
- Scalability: Efficient for handling large datasets due to its simplicity and speed.

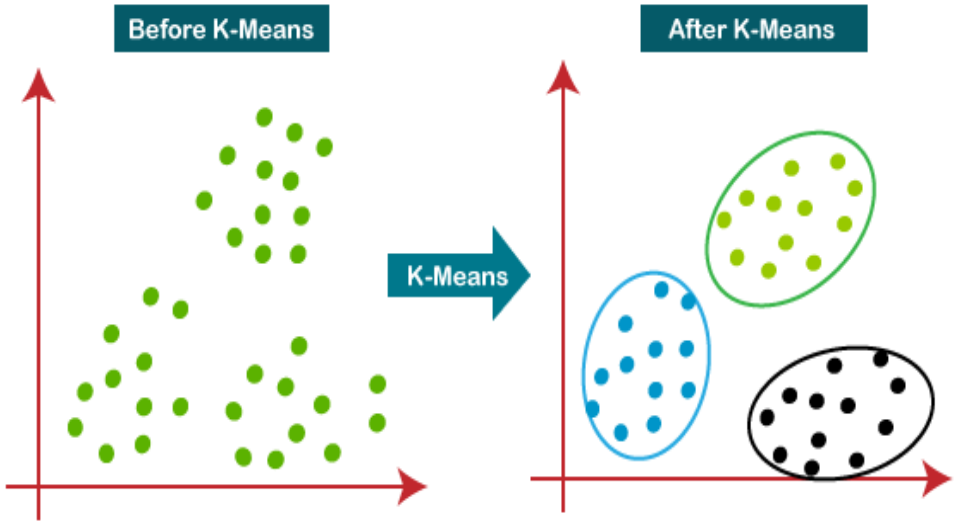

Fig III.11: K-Means Clustering

III.5 Fundamentals of Bio-Inspired Algorithms

Bio-inspired algorithms are computational methods that mimic natural systems, such as biological processes or behaviors observed in nature. These algorithms often involve evolutionary principles, swarm intelligence, or neural networks inspired by the brain. Examples include genetic algorithms, ant colony optimization, and artificial neural networks [21].

III.6 Machine Learning Techniques Inspired by Nature

Here are detailed explanations of six prominent machine learning techniques inspired by nature, often referred to as nature-inspired or bio-inspired algorithms. These techniques mimic natural processes such as evolution, swarm behavior, and biological intelligence to solve complex optimization and learning problems efficiently:

III.6.1 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are modeled after biological neural networks. They consist of layers of interconnected nodes (neurons) that process and transmit information. ANNs are widely used for tasks such as image recognition, speech processing, and natural language processing. The inspiration from biological neural networks lies in how neurons communicate through synapses, influencing the development of algorithms like backpropagation for training ANNs.

III.6.2 Genetic Algorithms (GAs)

Genetic Algorithms (GAs) are a powerful nature-inspired optimization technique modeled on the principles of natural selection and genetics [22]. They are widely used to solve complex problems where traditional methods may struggle, by iteratively evolving a population of candidate solutions toward better fitness.

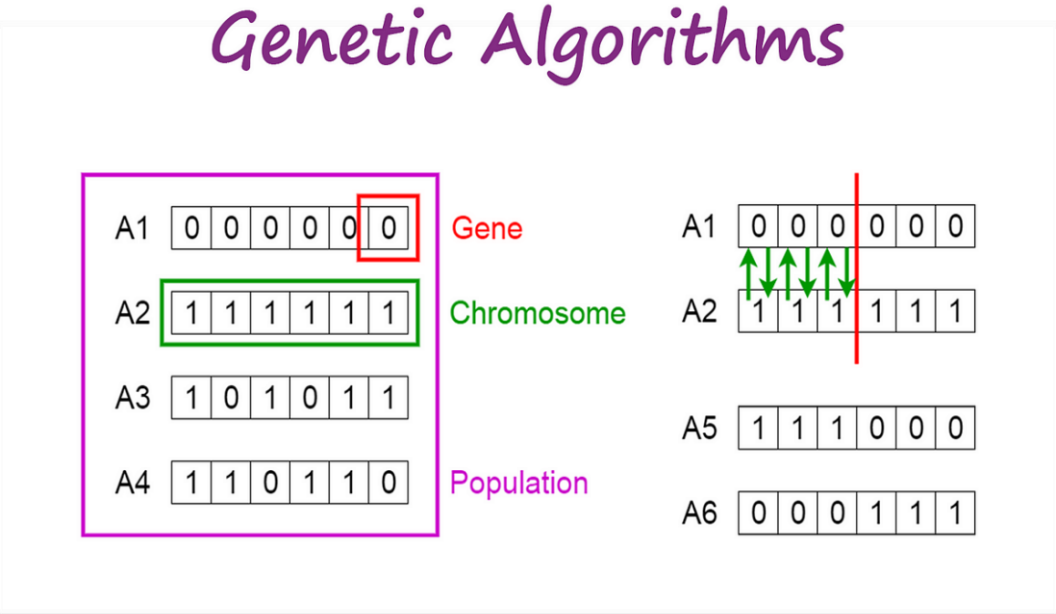

Fig III.12: Genetic Algorithms (GAs)

III.6.2.1 Population: GAs maintain a population of candidate solutions, often encoded as chromosomes (strings of bits, real numbers, or other data structures) representing possible solutions to the problem. The population is usually initialized randomly to ensure diversity.

III.6.2.2 Fitness Function: Each individual in the population is evaluated using a fitness function that quantifies how well the solution solves the problem. This function guides the evolutionary process by favoring better solutions for reproduction.

III.6.2.3 Selection: Inspired by "survival of the fittest," individuals with higher fitness have a greater chance of being selected to reproduce. Common selection methods include roulette wheel selection, tournament selection, and rank-based selection, each balancing exploration and exploitation differently.

III.6.2.4 Crossover (Recombination): Pairs of selected individuals exchange parts of their chromosomes to create offspring. This mimics biological sexual reproduction, combining traits from two parents to produce genetically diverse children. Various crossover methods exist, such as single-point, multi-point, and uniform crossover.

III.6.2.5 Mutation: To maintain genetic diversity and explore new areas of the solution space, random changes are introduced to offspring chromosomes by flipping bits or altering genes. Mutation rates must be carefully tuned to avoid premature convergence or excessive randomness.

III.6.2.6 Generation Cycle: After selection, crossover, and mutation, a new population is formed, replacing some or all of the old population. This cycle repeats for many generations until a stopping criterion is met (e.g., maximum generations or satisfactory fitness).

III.6.3 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a nature-inspired, population-based optimization algorithm modeled on the social behavior of bird flocks or fish schools. It iteratively improves candidate solutions to an optimization problem by simulating a swarm of particles moving through the search space, guided by individual and collective experiences.

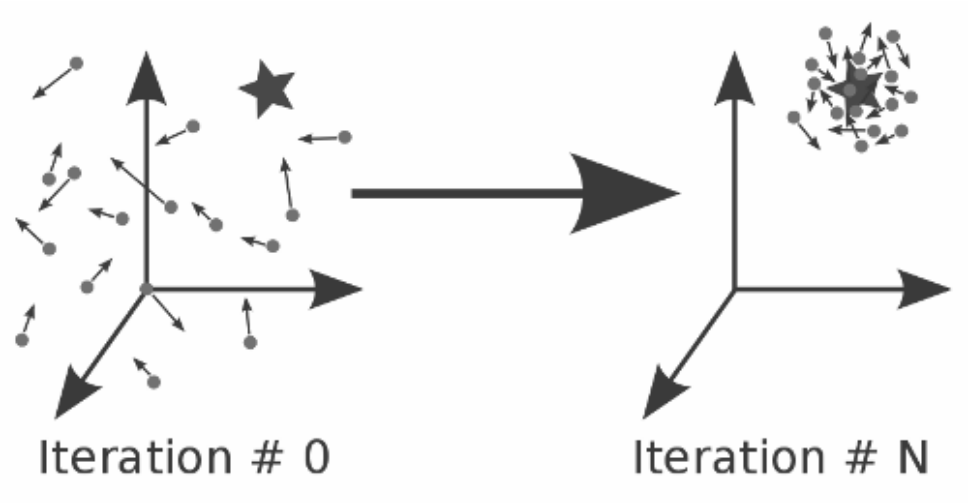

Fig III.13: Particle Swarm Optimization (PSO)

III.6.3.1 Particles and Positions: Each particle represents a potential solution with a position in the search space.

III.6.3.2 Velocity: Each particle has a velocity that determines its movement direction and speed.

III.6.3.3 Personal Best (pbest): Each particle remembers the best position it has found so far.

III.6.3.4 Global Best (gbest) or Neighborhood Best: The swarm shares information to identify the best position found by any particle (global best) or within a local neighborhood (local best).

III.6.3.5 Update Rules: At each iteration, particles update their velocities and positions based on:

- Their current velocity (momentum/inertia),
- The difference between their personal best and current position (cognitive component),
- The difference between the global or neighborhood best and current position (social component).

III.6.4 Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) is a nature-inspired metaheuristic algorithm modeled on the foraging behavior of real ants, particularly how they find shortest paths between their colony and food sources by depositing and following pheromone trails [23].

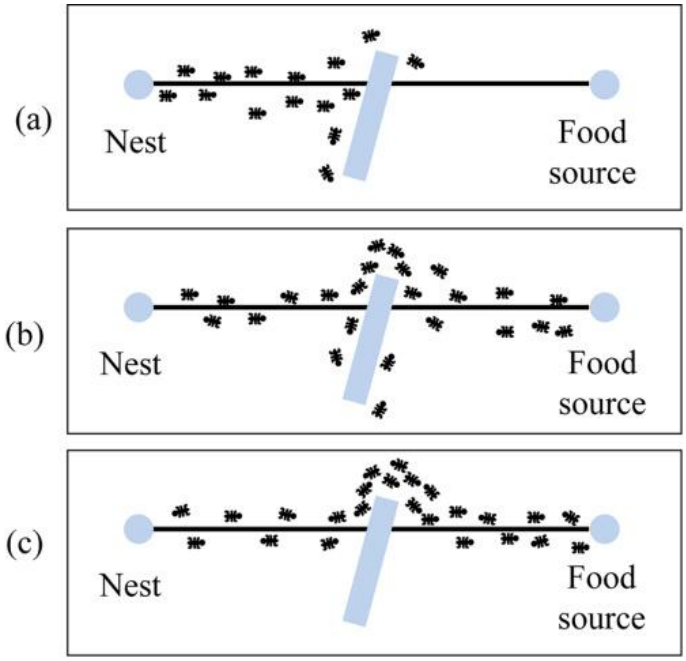

Fig III.14: Ant Colony Optimization (ACO)

III.6.4.1 Artificial Ants and Construction Graph: In ACO, artificial ants build solutions by traversing a fully connected graph representing the problem space. Each edge or node corresponds to a component of the solution.

III.6.4.2 Pheromone Trails: As ants move through the graph, they deposit virtual pheromones on edges they travel. The amount of pheromone represents the learned desirability of that path based on previous ants' experiences.

III.6.4.3 Probabilistic Path Selection: Ants choose their next move based on a probability influenced by two factors:

- The intensity of pheromone on the path (representing collective learning),
- A heuristic measure such as the inverse of distance or cost (representing problem-specific knowledge).

III.6.4.4 Pheromone Evaporation: Over time, pheromone levels evaporate, reducing the attractiveness of less optimal paths and preventing premature convergence on suboptimal solutions.

III.6.4.5 Positive Feedback Loop: Paths that lead to better solutions accumulate more pheromone, attracting more ants and reinforcing those paths, guiding the colony toward optimal or near-optimal solutions.

III.6.5 Firefly Algorithm (FA)

The Firefly Algorithm (FA) is a nature-inspired metaheuristic optimization technique developed by Xin-She Yang in 2008, modeled on the flashing behavior and attraction patterns of fireflies [24]. It is widely used to solve complex optimization problems by simulating how fireflies attract each other based on their brightness, which corresponds to the quality of candidate solutions [25].

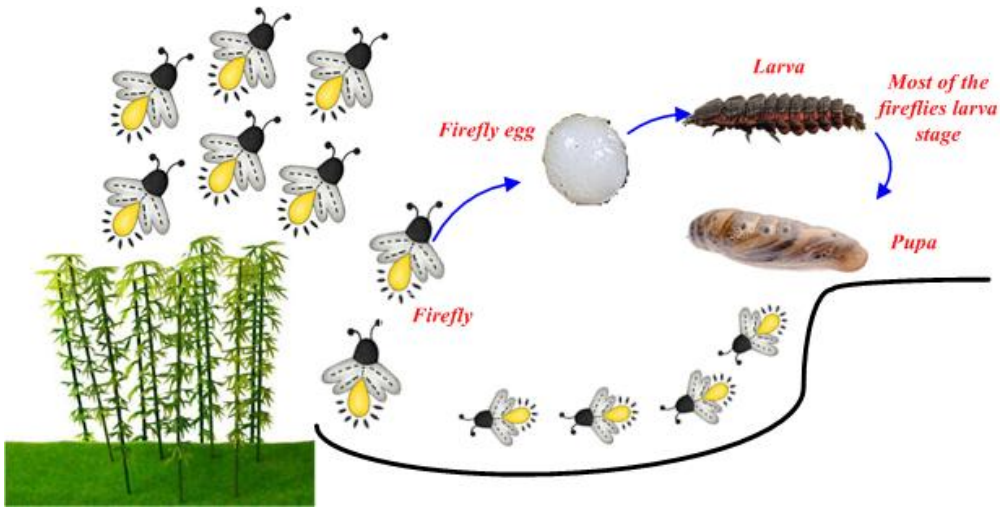

Fig III.15: Life cycle of Firefly Algorithm

III.6.5.1 Inspiration: Fireflies use bioluminescent flashes to attract mates or prey. Brighter fireflies attract others more strongly, but light intensity decreases with distance due to absorption.

III.6.5.2 Population: Each firefly represents a potential solution in the search space.

III.6.5.3 Brightness (Light Intensity): Corresponds to the objective function value; brighter fireflies represent better solutions.

III.6.5.4 Attractiveness: A firefly's attractiveness to others is proportional to its brightness but decreases exponentially with distance, modeled by an absorption coefficient γ.

III.6.5.5 Movement: Less bright fireflies move toward brighter ones. If no brighter firefly is found, a firefly moves randomly, allowing exploration.

III.6.6 Bacterial Foraging Optimization (BFO)

The Bat Algorithm (BA) is a nature-inspired metaheuristic optimization algorithm developed by Xin-She Yang in 2010, inspired by the echolocation behavior of microbats. It is designed for solving complex global optimization problems by simulating how bats use sound pulses to navigate and locate prey [26].

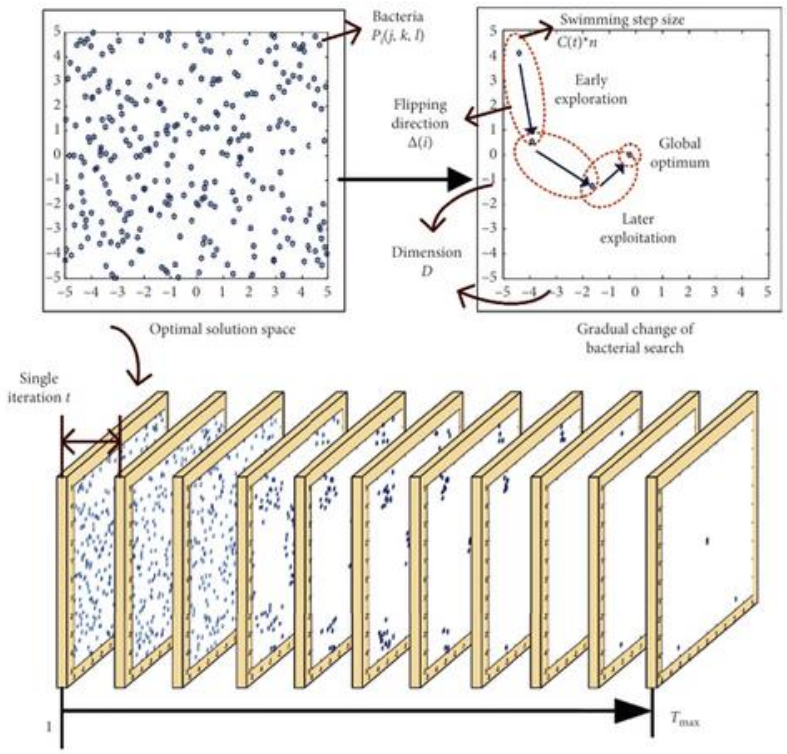

Fig III.16: Fundamental Structure of the BFO Algorithm

III.6.6.1 Echolocation Metaphor: Bats emit ultrasonic pulses and listen to the echoes to estimate the distance and location of prey or obstacles. The algorithm models this by having virtual bats "fly" through the solution space, adjusting their positions and velocities based on pulse frequency, loudness, and pulse emission rate.

III.6.6.2 Population-Based Search: A swarm of bats (candidate solutions) explores the search space. Each bat has a position (solution), velocity, frequency (controlling step size), loudness (intensity of search), and pulse emission rate (probability of local search).

III.6.6.3 Frequency Tuning: Bats adjust their pulse frequency to control the pace and direction of movement, balancing exploration (global search) and exploitation (local search).

III.6.6.4 Movement and Updating: At each iteration, bats update their velocities and positions based on their frequency and the global best solution found so far. Loudness and pulse emission rates adapt dynamically: loudness typically decreases as bats get closer to prey (better solutions), while pulse rate increases to intensify local search.

III.6.6.5 Local Random Walk: When a bat finds a promising solution, it performs a local random walk to explore the neighborhood, improving exploitation.

III.6.6.6 Selection: The algorithm selects the best solutions iteratively until convergence or a stopping criterion is met.

III.7 Applications of Bio-Inspired Machine Learning

Bio-inspired machine learning has diverse applications:

III.7.1 Optimization Problems

Bio-inspired algorithms, such as genetic algorithms and ant colony optimization, are highly effective in solving complex optimization problems. These algorithms mimic natural processes like evolution and swarm behavior to find optimal solutions. Genetic algorithms use principles of natural selection to evolve better solutions over generations, while ant colony optimization simulates how ants find the shortest path to food sources by depositing pheromones. These methods are particularly useful in scenarios where traditional optimization techniques struggle due to the complexity or non-linearity of the problem [21].

III.7.2 Pattern Recognition

Techniques inspired by the brain, such as artificial neural networks, are widely used for pattern recognition tasks like image and speech recognition. Neural networks mimic the structure and function of biological neural networks, allowing them to learn complex patterns from data. Deep learning models, which are a subset of neural networks, have achieved state-of-the-art results in image recognition, speech processing, and natural language processing [27].

III.7.3 Robotics and Autonomous Systems

Bio-inspired approaches are crucial in developing more adaptive and efficient autonomous systems. For example, swarm intelligence algorithms can be used to control swarms of robots, enabling them to perform complex tasks collectively. Additionally, bio-inspired robotics often involves designing robots that mimic the movement or sensing capabilities of animals, such as robotic snakes or birds, which can navigate through challenging environments more effectively.

III.7.4 Healthcare and Biotechnology

Bio-inspired machine learning is increasingly applied in healthcare and biotechnology for tasks such as drug discovery, disease modeling, and personalized medicine. Bio-inspired algorithms can help in analyzing complex biological data, predicting protein structures, and identifying potential drug targets. Moreover, machine learning models inspired by biological systems can aid in diagnosing diseases more accurately and developing personalized treatment plans based on genetic profiles [28].

III.7.5 Telecommunications and Network Design

Swarm intelligence is applied to optimize network routing, spectrum allocation, load balancing, and fault tolerance in wireless and wired communication networks. These algorithms enhance data transmission efficiency, reduce congestion, and improve network resilience.

III.7.6 Big Data and Cloud Computing

With the growth of big data and cloud services, bio-inspired algorithms are used for scalable data analysis, feature selection, and cloud resource management. Their parallelism and adaptability make them suitable for handling large-scale, distributed computing environments.

III.7.7 Cybersecurity and Anomaly Detection

Artificial immune systems and other bio-inspired models detect network intrusions, malware, and insider threats by learning normal behavior patterns and identifying anomalies. These adaptive systems improve the robustness and responsiveness of cybersecurity defenses.

III.8 Conclusion

Machine learning and bio-inspired algorithms form a synergistic partnership that enhances the ability to solve complex computational problems by drawing inspiration from natural processes. Bio-inspired algorithms-such as genetic algorithms, swarm intelligence, and neural network models-mimic biological evolution, social behaviors, and brain functions, enabling machine learning systems to become more adaptable, efficient, and robust. This integration leads to innovative solutions across diverse fields including healthcare, robotics, finance, and big data analytics.

By incorporating principles like hierarchical information processing, context-dependent learning, and adaptive evolution, bio-inspired machine learning models can better handle noisy, dynamic, and high-dimensional data. For example, convolutional neural networks (CNNs) are inspired by the hierarchical structure of the visual cortex, enabling efficient feature extraction and object recognition. Similarly, recent advances explore integrating biological complexity such as neuron-astrocyte interactions to improve transformer architectures, potentially enhancing learning efficiency, robustness, and energy consumption.

As both fields evolve, their fusion is expected to drive significant advancements in artificial intelligence by enabling systems that learn more like natural organisms-capable of generalization, causal reasoning, and adaptation to new environments. This convergence also addresses limitations of traditional AI models, such as brittleness and lack of deep understanding, by embedding biological principles of plasticity, modularity, and self-organization. Furthermore, bio-inspired algorithms contribute to managing big data challenges by providing scalable, intelligent, and robust methods for data fusion, storage, and processing.

In summary, the ongoing integration of machine learning with bio-inspired algorithms leverages nature's proven strategies to create more powerful, flexible, and intelligent AI systems. This interdisciplinary approach not only enhances current capabilities but also opens new avenues for research and application, promising transformative impacts across science, engineering, and industry.

Chapter IV

Particle Swarm Optimization (PSO)

IV.1 Introduction

Swarm intelligence (SI) is based on the collective behavior of decentralized, self-organized systems. It may be natural or artificial. Natural examples of SI are ant colonies, fish schooling, bird flocking, bee swarming and so on.

Besides multirobot systems, some computer program for tackling optimization and data analysis problems are examples for some human artifacts of SI.

The most successful swarm intelligence techniques are Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). In PSO, each particle flies through the multidimensional space and adjusts its position in every step with its own experience and that of peers toward an optimum solution by the entire swarm. Therefore, the PSO algorithm is a member of Swarm Intelligence [29].

IV.2 Particle Swarm Optimization (PSO) Algorithm

The Particle Swarm Optimization algorithm (PSO) is a novel population-based stochastic search algorithm and an alternative solution to the complex non-linear optimization problem. The PSO algorithm was first introduced by Dr. Kennedy and Dr. Eberhart in 1995 and its basic idea was originally inspired by simulation of the social behavior of animals such as bird flocking, fish schooling and so on. It is based on the natural process of group communication to share individual knowledge when a group of birds or insects search food or migrate and so forth in a searching space, although all birds or insects do not know where the best position is. But from the nature of the social behavior, if any member can find out a desirable path to go, the rest of the members will follow quickly.

IV.3 Dataset

dataset is a collection of related, discrete items of related data represented by rows and columns. Columns are also called features which represent variables that could be numerical, categorical …etc. Rows are the instance of the variables (columns).

In this thesis two datasets were used, NSL-KDD & MQTTset.

IV.3.1 NSL-KDD Dataset

In the realm of cybersecurity and network intrusion detection, the NSL-KDD dataset stands as a benchmark for evaluating machine learning models' performance. This dataset, derived from the original KDD Cup 1999 dataset, addresses the limitations and biases present in its predecessor, making it a vital resource for researchers and practitioners in the field of IDS.

No	Features	Form of value	No	Features	Form of value
1	Duration	Integer	22	is_guest_login	Integer
2	protocol_type	Nominal	23	count	Integer
3	service	Nominal	24	srv_count	Integer
4	flag	Nominal	25	serror_rate	Float
5	src_bytes	Integer	26	srv_serror_rate	Float
6	dst_bytes	Integer	27	rerror_rate	Float
7	Land	Integer	28	srv_rerror_rate	Float
8	wrong_fragment	Integer	29	same_srv_rate	Float
9	Urgent	Integer	30	diff_srv_rate	Float
10	Hot	Integer	31	srv_diff_host_rate	Float
11	num_failed_logins	Integer	32	dst_host_count	Float
12	logged_in	Integer	33	dst_host_srv_count	Float
13	num_compromised	Integer	34	dst_host_same_srv_rate	Float
14	root_shell	Integer	35	dst_host_diff_srv_rate	Float
15	su_attempted	Integer	36	dst_host_same_src_port_rate	Float
16	num_root	Integer	37	dst_host_srv_diff_host_rate	Float
17	num_file_creations	Integer	38	dst_host_serror_rate	Float
18	num_shells	Integer	39	dst_host_srv_serror_rate	Float
19	num_access_files	Integer	40	dst_host_rerror_rate	Float
20	num_outbound_cmds	Integer	41	dst_host_srv_rerror_rate	Float
21	is_host_login	Integer	42	Class	Category

Table IV.1: The 41 features of the NSL-KDD dataset

IV.3.2 MQTTset Dataset

The proposed work aims to create a dataset linked to the IoT context, in particular on the MQTT communication protocol, in order to give to the research and industrial community an initial dataset to use in their application. The dataset is composed by IoT sensors based on MQTT where each aspect of a real network is defined. In particular, the MQTT broker is instantiated by using Eclipse Mosquito and the network is composed by 8 sensors. The scenario is related to a smart home environment where sensors retrieve information about temperature, light, humidity, CO-Gas, motion, smoke, door and fan with different time interval since the behaviour of each sensor is different with the others.

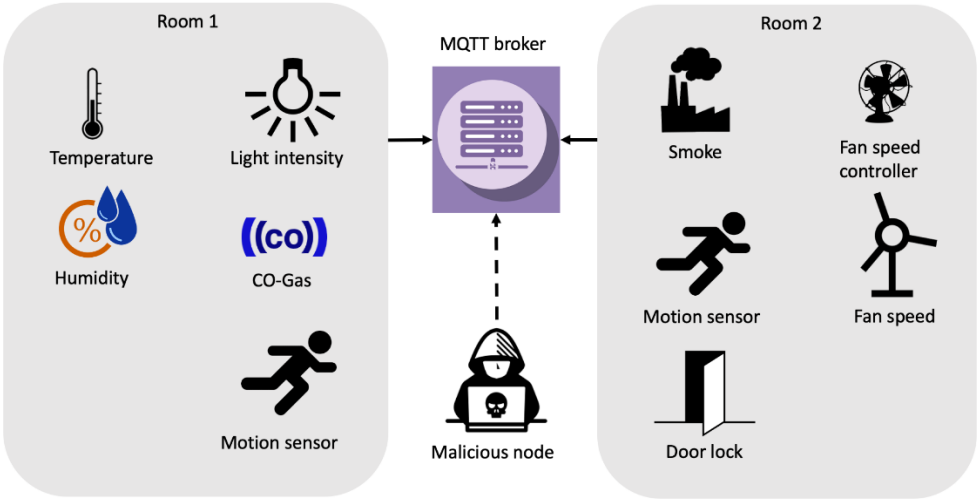

Fig IV.1: New Dataset for Machine Learning Techniques on MQTT

IV.3.3 Comparison Between NSL-KDD and MQTTset

Aspect	NSL-KDD	MQTTset
Focus	General network intrusion detection	IoT-specific, MQTT protocol intrusion detection
Domain	Traditional network traffic	IoT networks using MQTT protocol
Number of Features	41 features	33 features
Types of Attacks	DoS, Probe, R2L, U2R, etc.	Brute force, DoS, Flood, Malformed, Slow attacks
Machine Learning Models Used	K-means, Random Forest, CNN, Decision Trees, etc.	Decision Trees, Random Forest, Neural Networks, Boosting
Feature Selection	Information Gain, PCA, Select KBest	Feature engineering with selection down to 10 features

Table IV.2: Comparison Between NSL-KDD and MQTTset

IV.4 PSO Methode Overview

IV.4.1 Binary Particle Representation

In binary Particle Swarm Optimization (BPSO), particles are represented as binary strings (0, 1, 1), where each bit (0 or 1) corresponds to a decision variable (feature selection, on/off states). This adaptation of PSO for discrete optimization problems involves unique mechanisms for updating positions and velocities. Below is a detailed breakdown:

IV.4.1.1 Position Encoding

- Each particle's position is a binary vector of length n, where n is the number of decision variables.

Example: For feature selection with 5 features, a particle may be encoded as [1, 0, 1, 1] indicating the selection of features 1, 3, and 4.

IV.4.1.2 Velocity Interpretation

- Unlike continuous PSO, velocity in BPSO determines the probability of flipping a bit (0↔1).
- Velocity values are mapped to probabilities using a sigmoid function:

$$Prob(x_i = 1) = \frac{1}{1+e^{-v_i}} \quad (1)$$

Where v_i is the velocity of the i-th bit.

IV.4.1.3 Velocity Update

- Velocities are updated similarly to continuous PSO but constrained to prevent divergence:

$$v_i^{t+1} = \omega * v_i^t + c_1 r_1 (pbest_i - x_i^t) + c_2 r_2 (gbest_i - x_i^t) \quad (2)$$

- Clamping: Velocities are often clamped to $[-v_{max}, v_{max}](e.g., v_{max} = 6)$ to stabilize probabilities.

IV.4.1.4 Position Update

- Each bit *xi* is updated by comparing its velocity-derived probability to a random number r ∈ [1]:

$$x_i^{t+1} = \begin{cases} 1, & \frac{1}{1+e^{-v_i^{t+1}}} > r \\ 0, & otherwise \end{cases} \quad (3)$$

IV.4.2 Initialization

- A swarm of particles is initialized with random positions and velocities within the defined search space.
- Each particle's position corresponds to a potential solution.

IV.4.3 Fitness Evaluation

- The objective function (fitness) is evaluated at each particle's position to assess solution quality.

IV.4.4 Update Personal and Global Bests

- Each particle keeps track of its best position found so far (personal best, p_i).
- The swarm maintains the best position found by any particle (global best, g_i).

IV.4.5 Velocity and Position Update

- Particle velocities are updated based on three components:
  - Inertia: maintains the current velocity to explore the search space.
  - Cognitive component: attraction toward the particle's personal best position.
  - Social component: attraction toward the swarm's global best position.
- The velocity update formula for particle i in dimension d is:

$$v_{i,d} \leftarrow \omega * v_{i,d} + \emptyset_p * r_p * (p_{i,d} - x_{i,d}) + \emptyset_g * r_g * (g_{i,d} - x_{i,d}) \quad (4)$$

where:

- $\boldsymbol{\omega}$ is the inertia weight controlling exploration vs. exploitation,
- $\boldsymbol{\emptyset_p, \emptyset_g}$ are acceleration coefficients (cognitive and social),
- $\boldsymbol{r_p, r_g}$ are random numbers uniformly distributed in 1,
- $\boldsymbol{x_{i,d}}$ is the current position.

Positions are updated by:

$$x_i \leftarrow x_i + v_i \quad (5)$$

IV.4.6 Boundary Handling

- Positions and velocities are clamped to predefined bounds to keep particles within the search space.

IV.4.7 Iteration and Termination

- Steps 4.3–4.6 are repeated until a stopping criterion is met (e.g., maximum iterations or satisfactory fitness).

IV.5 The basic flow of the PSO approach

The basic flow of the Particle Swarm Optimization (PSO) algorithm can be summarized in the figure below:

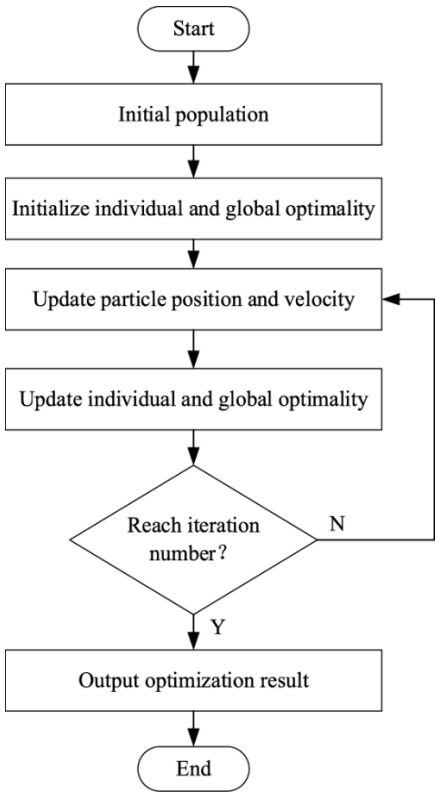

Fig IV.2: The basic flow of the PSO approach

IV.6 Python Code

Python is a high-level, interpreted programming language known for its simplicity and readability. It was created by Guido van Rossum and first released in 1991.

IV.6.1 Environment

IV.6.1.1 Anaconda

Anaconda is a package manager, an environment manager, a Python/R science distribution and a collection of over +7500 open-source packages [30].

It is used for scientific computing like Data science, Machine learning applications, predictive analytics and many other fields.

This package manager makes installing and using libraries such (ScikitLearn, Numpy, …etc) easy and simply, which grants a stable environment with less time consumption while developing a project.

It could be installed for Windows, Linux and MacOS, in our case It was installed in windows operating system following these steps [31]:

1- Downloading the Anaconda installer: (https://www.anaconda.com/download/#windows)

2- Verify data integrity with SHA-256.

3- Following the steps on the Anaconda installer window for installing any package needed a simple command anaconda install Package name.

IV.6.1.2 Visual studio code (VS Code)

Visual Studio Code (VS Code) is a free, lightweight, and highly customizable source code editor developed by Microsoft. It supports hundreds of programming languages and offers powerful features such as syntax highlighting, IntelliSense (smart code completion), integrated debugging, Git version control, and an integrated terminal. VS Code's extensible architecture allows users to enhance its functionality through thousands of extensions available in its marketplace. It runs on Windows, macOS, and Linux, making it a versatile tool for a wide range of development tasks-from simple scripting to complex application development-while providing a fast and streamlined coding experience.

IV.6.2 Libraries

IV.6.2.1 NumPy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.[32]

It's an open-source library that is used widely in Python programming and almost in all fields of science and engineering. It contains multidimensional array and matrix data structures, and can perform various mathematical operations on them.

IV.6.2.2 Pandas

Pandas is a fast, powerful, flexible and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language.[33]

It provides a high performance, easily used data structures and data analysis tools, It9s the library that is used to manipulate datasets data frames.

IV.6.2.3 Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python [34], it's a visualization library that offers the ability to display data, results in various shapes and diagrams.

IV.6.2.4 Scikit-Learn (Sklearn)

Scikit-learn is an open-source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities.[35]

This library provides a large number of built-in machine leaning algorithms and models such as Label Encoder for label encoding (explained later on this article), confusion_matrix to calculate the true/false negatives and true/false positives, train_test_split to split the data frame obtained from the dataset to a training set and to a testing set, and many other useful models.

IV.6.3 The Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import time
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.utils.validation import check_is_fitted # Import
check_is_fitted
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score, precision_recall_fscore_support
```

Libraries and Classifiers Used in the Python Code

```
def get_classifier(classifier_type='nb', **kwargs):
    if classifier_type == 'knn':
        return KNeighborsClassifier(n_neighbors=3, n_jobs=1)
    elif classifier_type == 'lr':
        return LogisticRegression(max_iter=1000, random_state=42,
                                  n_jobs=-1, **kwargs)
    elif classifier_type == 'svm':
        return SVC(probability=True, random_state=42, **kwargs)
    elif classifier_type == 'nb':
        return GaussianNB(var_smoothing=1e-9, **kwargs)
    elif classifier_type == 'mlp':
        return MLPClassifier(hidden_layer_sizes=(100,), max_iter=1000,
                                  random_state=42, **kwargs)
    elif classifier_type == 'dt':
        return DecisionTreeClassifier(random_state=42)
    elif classifier_type == 'rf':
        return RandomForestClassifier(random_state=42, **kwargs)
    else:
        raise ValueError(f"Unknown classifier type: {classifier_type}")
```

Approaches for Selecting the Desired Classification Algorithm for Implementation

```
def _initialize_swarm(self, n_features):
     # Initialize positions (binary vectors)
     positions = np.random.randint(0, 2, size=(self.n_particles,
                                               n_features))

     # Ensure each particle selects at least one feature
     for i in range(self.n_particles):
         if np.sum(positions[i]) == 0:
             # Select a random feature if none selected
             positions[i, np.random.randint(0, n_features)] = 1

             # Initialize velocities (real values)
             velocities = np.random.uniform(-1, 1,
                                    size=(self.n_particles, n_features))

             return positions, velocities
```

Initialization of Positions and Velocities in Binary PSO with Feature Selection Constraints

```
def _calculate_fitness(self, X, y, position):
    # Ensure at least one feature is selected
    if np.sum(position) == 0:
        return 0

    # Select features based on particle position
    selected_features = np.where(position == 1)[0]
    X_selected = X[:, selected_features]

    # Calculate cross-validation accuracy
    try:
        cv_scores = cross_val_score(
            self.classifier, X_selected, y, cv=5, scoring='accuracy', n_jobs=-1)
        accuracy = np.mean(cv_scores)
    except Exception:
        # Fall back to 3-fold CV if 5-fold fails (e.g., for small datasets)
        try:
            cv_scores = cross_val_score(
                self.classifier, X_selected, y, cv=3, scoring='accuracy', n_jobs=-1)
            accuracy = np.mean(cv_scores)
        except Exception:

            # Last resort: use a simple train/test split
            X_train, X_test, y_train, y_test = train_test_split(
                X_selected, y, test_size=0.3, random_state=42)
            self.classifier.fit(X_train, y_train)
            accuracy = self.classifier.score(X_test, y_test)

    # Calculate feature reduction ratio
    feature_ratio = np.sum(position) / len(position)

    # Calculate fitness (maximize accuracy, minimize features)
    fitness = self.alpha * accuracy - self.beta * feature_ratio

    return fitness
```

Fitness Evaluation Method for Feature Selection using Classification Accuracy and Feature Reduction Trade-off

```
# PSO feature selection
print("\nRunning PSO feature selection...")
pso = PSOFeatureSelection(
    n_particles=50,
    n_iterations=50,
    w=0.7,
    c1=1.5,
    c2=1.5,
    alpha=0.9,
    beta=0.1,
    classifier_type=classifier_type,
    classifier_params=classifier_params,
    verbose=True)
```

Parameter Configuration of the PSO-Based Feature Selection Algorithm

IV.7 Experiments & Results

In this table, we conduct empirical experiments on a set of algorithms including KNN, LR, NB, SVM, MLP, RF, and DT. Their performance is compared based on all-features accuracy versus PSO-selected accuracy, in addition to the number of selected features, feature reduction rate, accuracy improvement, and training time reduction.

Algorithm	All Features Accuracy	PSO Selected Accuracy	n_Features Selected	Features Reduction Rate	Accuracy Improvement	Training time reduction
KNN	0.7910	0.8345	37	68.4%	4.35%	50.0%
LR	0.8083	0.8722	40	65.8%	6.39%	7.9%
NB	0.5657	0.7799	54	53.8%	21.43%	50.0%
SVM	0.8057	0.8203	41	65.0%	1.46%	42.3%
MLP	0.8336	0.8802	41	65.0%	4.66%	1.5%
RF	0.7866	0.8567	40	65.8%	7.01%	12.1%
DT	0.8873	0.8372	36	69.2%	5.01%	55.9%

Table IV.3: Comparison of Classification Algorithms Using All Features VS PSO Selected Features in Terms of Accuracy and Training Efficiency

In the following figures, we illustrate the convergence behavior of the PSO algorithm during the feature selection process for each of the classification algorithms used in this study: KNN, LR, NB, SVM, MLP, RF, and DT. The top part of each figure shows the evolution of the fitness value across iterations, reflecting the algorithm's convergence toward an optimal solution. The bottom part displays the change in the number of selected features over iterations, showing a gradual reduction, which demonstrates the effectiveness of PSO in dimensionality reduction while maintaining model performance.

IV.7.1 KNN Algorithm

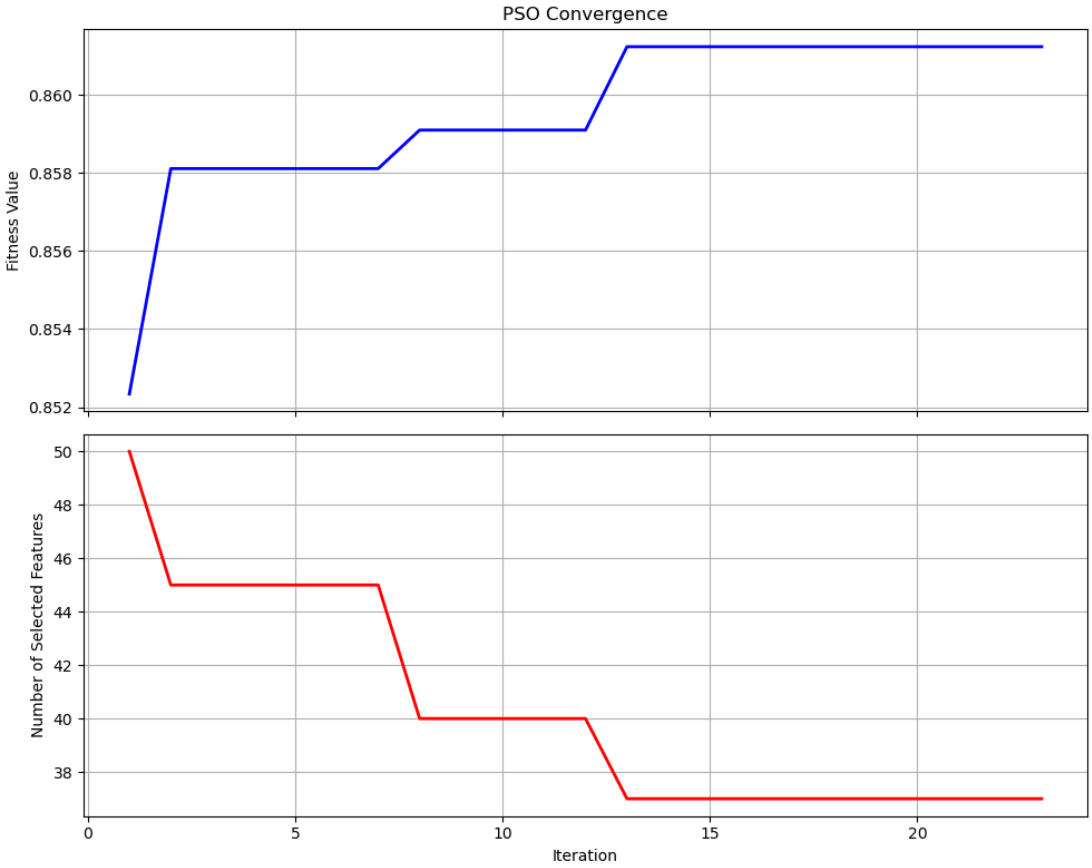

Fig IV.3: KNN Algorithm 50 iteration in 50 Particles

IV.7.2 LR Algorithm

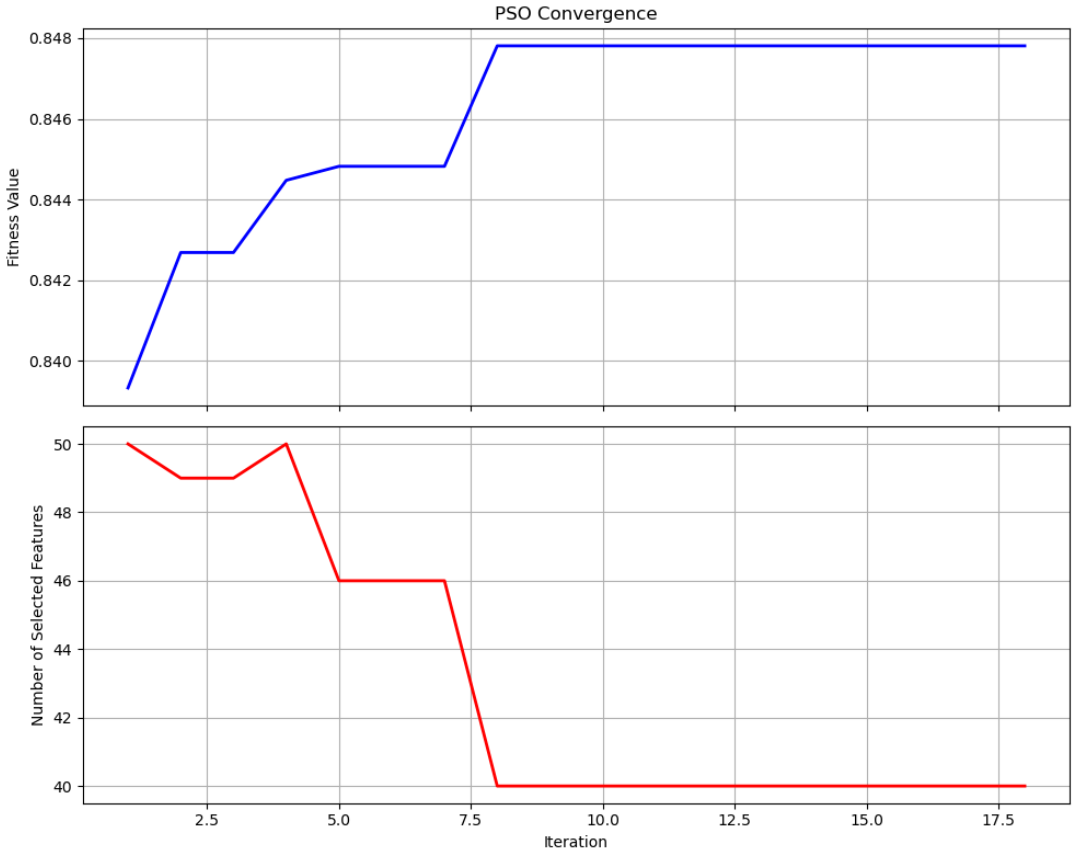

Fig IV.4: LR Algorithm 50 iteration in 50 Particles

IV.7.3 NB Algorithm

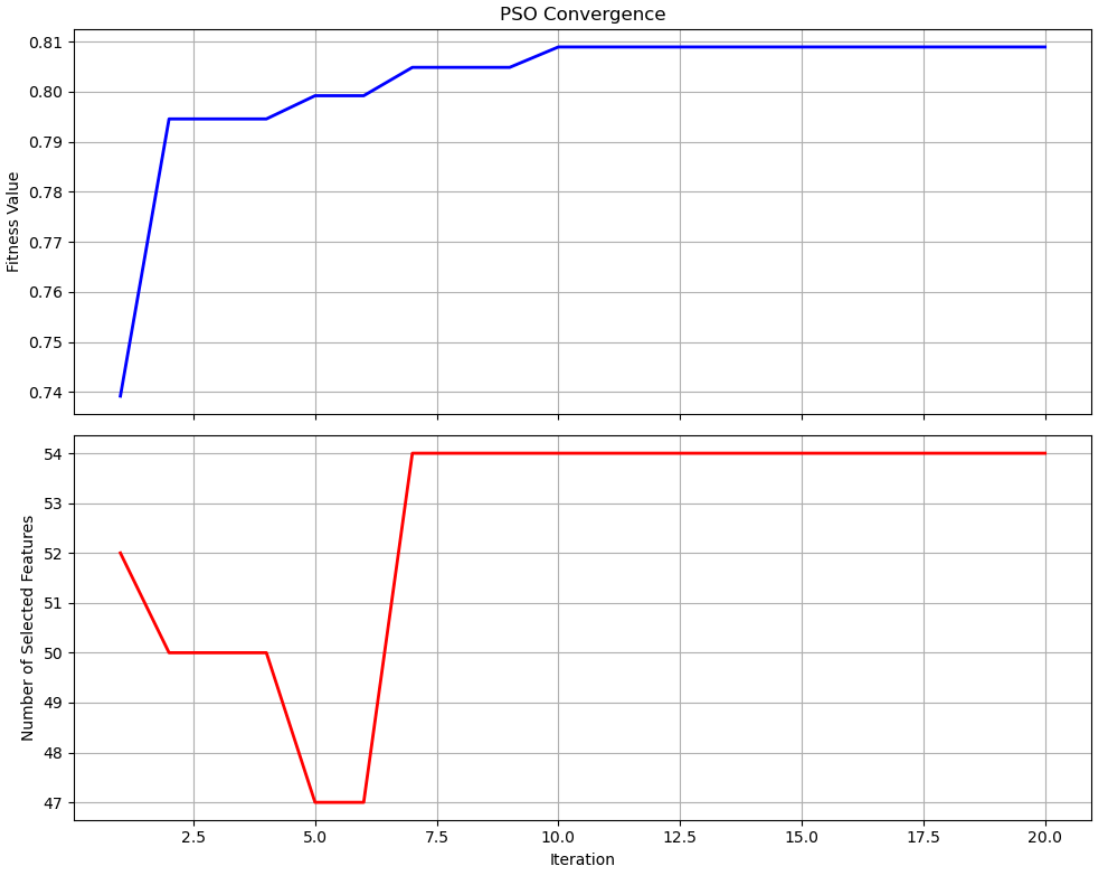

Fig IV.5: NB Algorithm 50 iteration in 50 Particles

IV.7.4 SVM Algorithm

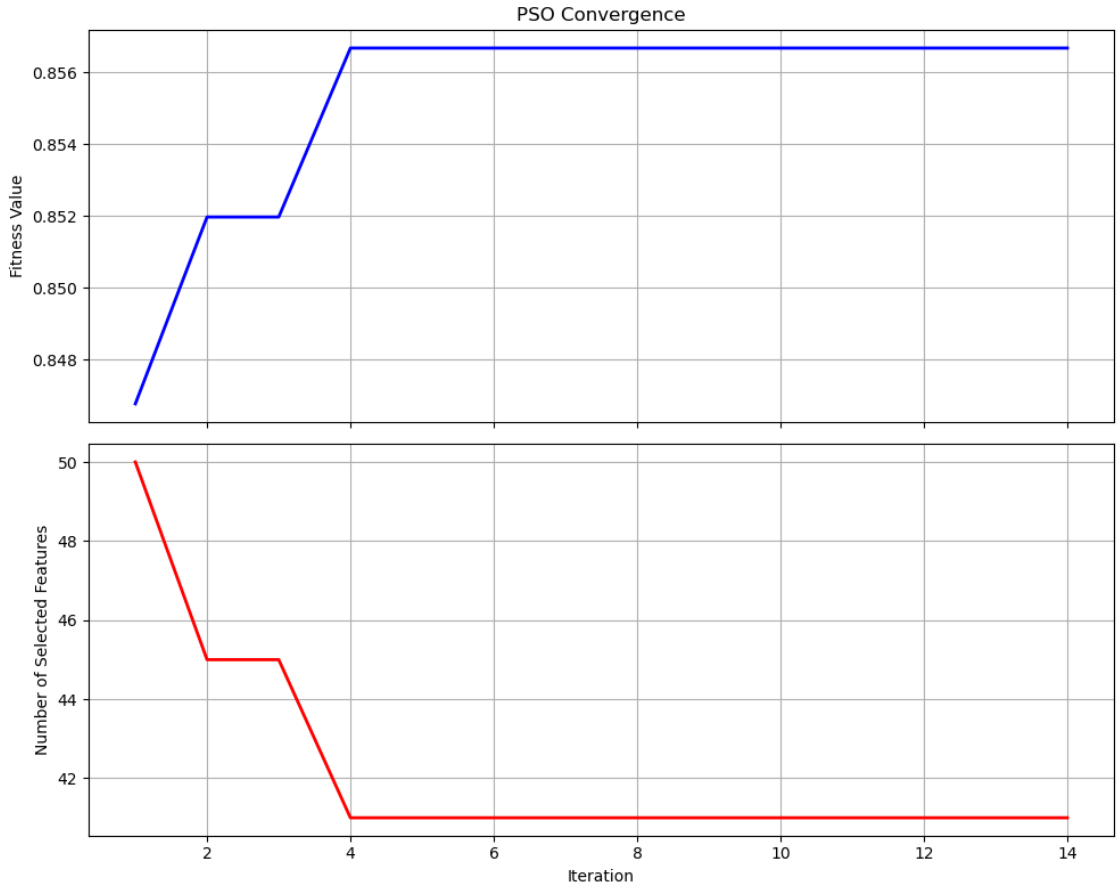

Fig IV.6: SVM Algorithm 50 iteration in 50 Particles

IV.7.5 MLP Algorithm

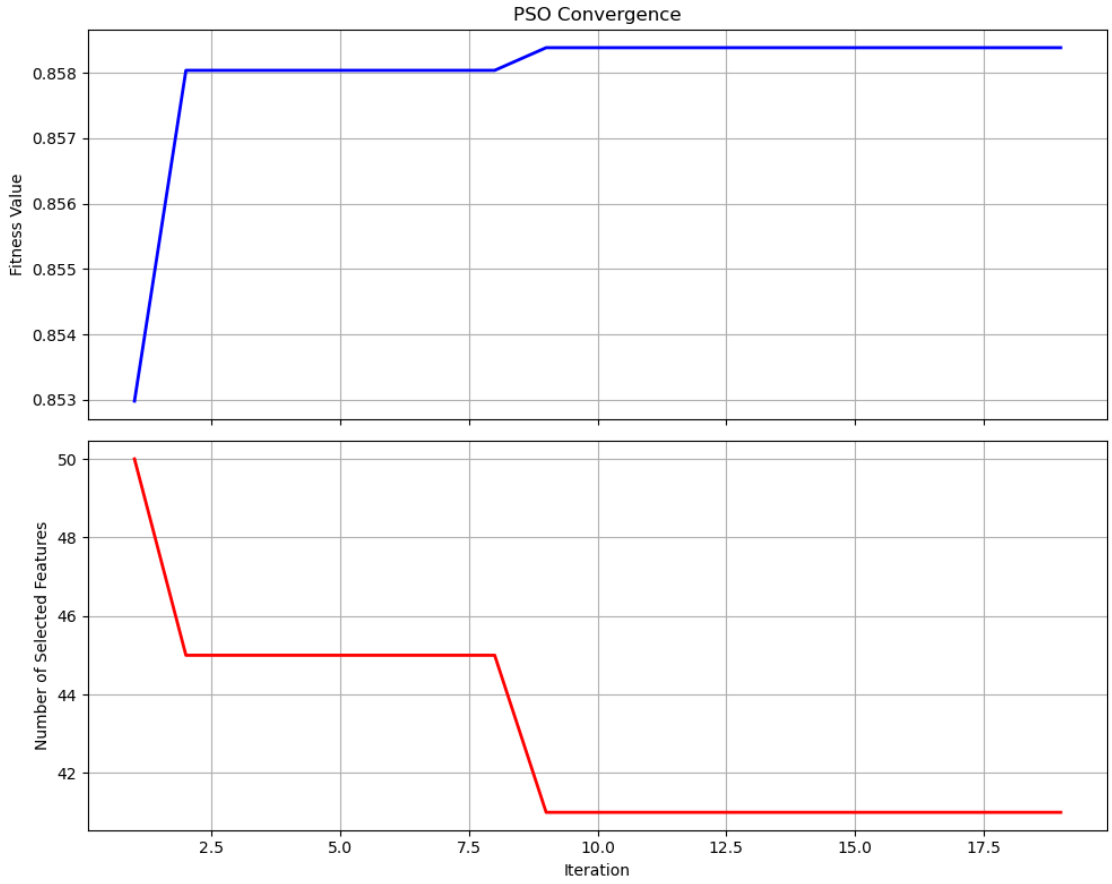

Fig IV.7: MLP Algorithm 50 iteration in 50 Particles

IV.7.6 RF Algorithm

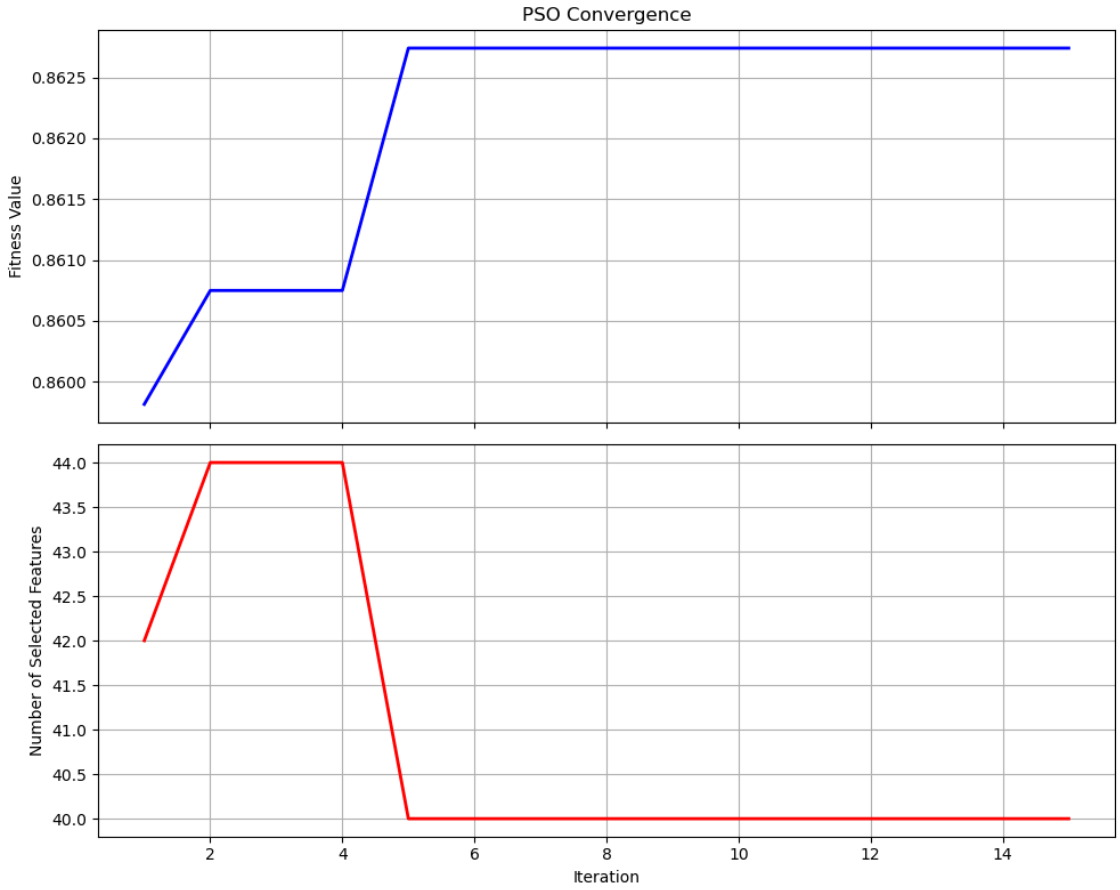

Fig IV.8: RF Algorithm 50 iteration in 50 Particles

IV.7.7 DT Algorithm

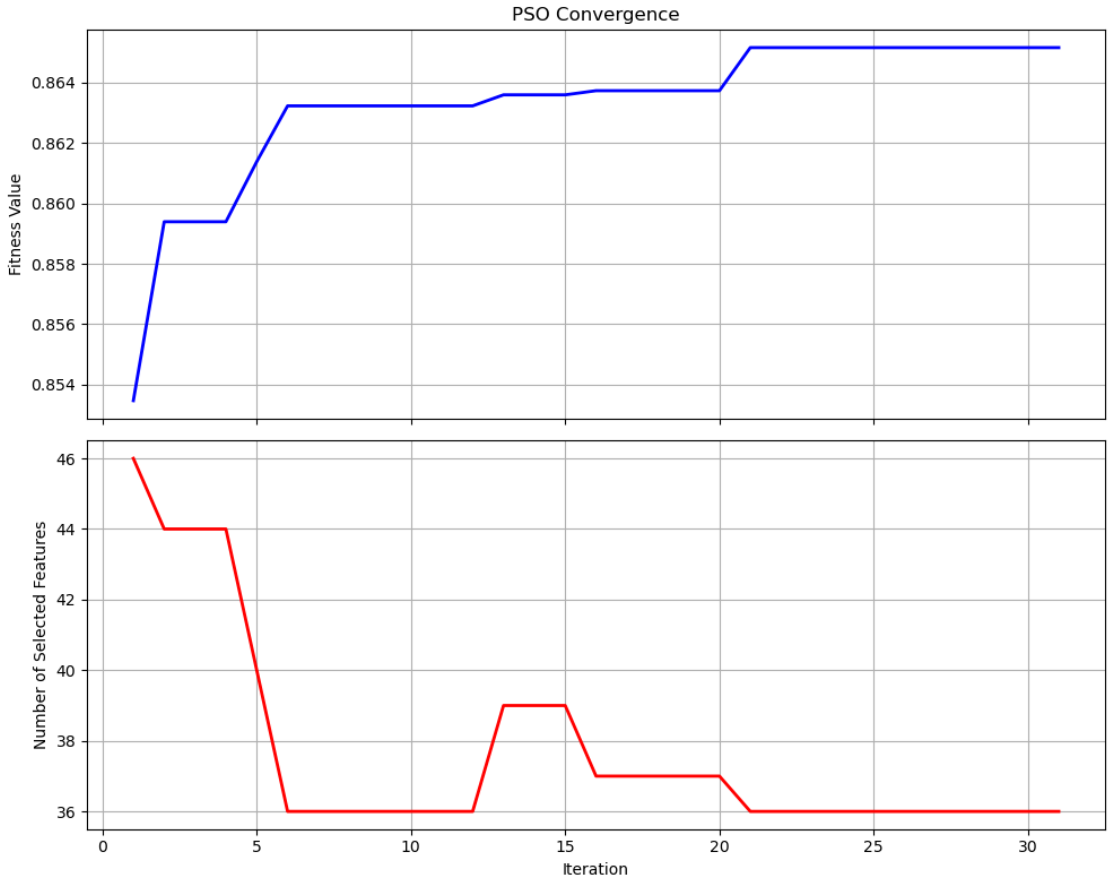

Fig IV.9: DT Algorithm 50 iteration in 50 Particles

IV.8 Discussions of Results

The experimental results using the Particle Swarm Optimization (PSO) algorithm for feature selection across multiple classification models—namely K-Nearest Neighbors (KNN), Logistic Regression (LR), Naïve Bayes (NB), Support Vector Machine (SVM), Multilayer Perceptron (MLP), Random Forest (RF), and Decision Tree (DT)—demonstrated significant improvements in classification accuracy when using a reduced subset of features compared to using the full feature set.

The PSO convergence plots revealed two key patterns across all models: an upward trend in fitness values and a downward trend in the number of selected features. These patterns confirm that PSO was effective in optimizing feature subsets that not only improved model performance but also reduced computational complexity.

IV.8.1 Model-Wise Analysis:

- **K-Nearest Neighbors (KNN):**

The accuracy improved from 0.7910 to 0.8345 with only 37 features selected. The feature reduction rate reached 68.4%, resulting in a substantial 50% reduction in training time. This suggests that KNN, being a distance-based model, greatly benefits from reduced feature dimensionality.

- **Logistic Regression (LR):**

Accuracy increased from 0.8083 to 0.8722 with 40 selected features, and a feature reduction rate of 65.8%. Though the training time reduction was relatively modest (7.9%), the improvement in accuracy (6.39%) validates the positive impact of removing redundant or irrelevant features in linear models.

- **Naive Bayes (NB):**

Despite having the lowest baseline accuracy (0.5657), NB saw the most significant relative improvement reaching 0.7799 after feature selection. With 54 features retained and a 53.8% reduction rate, training time was halved. This highlights NB's sensitivity to irrelevant features and the importance of dimensionality reduction for probabilistic models.

- **Support Vector Machine (SVM):**

The accuracy increased slightly from 0.8057 to 0.8203, with 41 features selected and a 42.3% reduction in training time. While the accuracy gain (1.46%) was modest, maintaining or slightly improving performance with fewer features is valuable for SVMs, which can be computationally intensive in high-dimensional spaces.

- **Multi Layer Perceptron (MLP):**

Performance improved from 0.8336 to 0.8802 using 41 features. The feature reduction rate was 65%, although the training time was only slightly reduced (1.5%). This result suggests that PSO successfully identified the most informative features, improving the generalization of deep learning models without sacrificing speed.

- **Random Forest (RF):**

The model's accuracy rose from 0.7866 to 0.8567 using 40 selected features. With a 65.8% reduction in features and a 12.1% decrease in training time, the model achieved a solid 7.01% improvement in accuracy, indicating PSO's effectiveness even for ensemble methods that are already robust to irrelevant features.

- **Decision Tree (DT):**

Interestingly, DT achieved one of the best overall results in terms of trade-off. Accuracy reached 0.8873 using only 36 features (a 69.2% reduction), along with the highest training time reduction (55.9%). Despite a small drop from the original accuracy (from 0.8372), the final performance still surpassed most other classifiers.

IV.8.2 Overall Insights:

The results clearly demonstrate the strength of PSO as a feature selection method for improving model performance and efficiency. All classifiers benefited to varying extents either through accuracy gains, reduced training time, or both. Notably, models like Naïve Bayes and KNN, which are particularly sensitive to irrelevant or noisy features, exhibited significant improvements. Even complex models like MLP and ensemble methods like RF showed better generalization and training efficiency.

In conclusion, PSO-based feature selection proves to be a powerful tool in optimizing classification tasks, reducing dimensionality while maintaining or enhancing predictive performance. These results reinforce the importance of feature selection as a critical step in the machine learning pipeline.

IV.9 Conclusion

This chapter discussed the fundamentals of the Particle Swarm Optimization (PSO) algorithm, including its geometric and mathematical foundations, particle movement and velocity updates within the search space, the role of acceleration coefficients, and different particle neighborhood topologies. It also explored the application of PSO in solving path planning problems. Finally, the Decision Tree (DT) algorithm demonstrated the best performance, achieving high accuracy with fewer selected features compared to other models, highlighting its effectiveness in building a more efficient and less complex intrusion detection system.

General Conclusion

General Conclusion

The convergence of Internet of Things (IoT), Intrusion Detection Systems (IDS), and machine learning (ML) enhanced by metaheuristic optimization algorithms such as Particle Swarm Optimization (PSO) constitutes a promising research direction in the field of cybersecurity. With the exponential growth of IoT deployments across critical domains including healthcare, smart cities, and industrial systems the threat landscape has become increasingly complex, heterogeneous, and dynamic. These environments demand robust, intelligent, and adaptive security solutions.

Machine learning-based IDSs have emerged as effective tools for detecting a wide range of cyber threats in real-time. However, their performance heavily relies on the quality of input features, model parameters, and computational efficiency, especially under the constrained resources typical of IoT devices. PSO, inspired by the social behavior of bird flocking, provides an efficient method for feature selection and hyperparameter optimization in ML based IDS. It enables the reduction of data dimensionality, enhances detection accuracy, and mitigates the risk of overfitting all while maintaining low computational overhead.

Through the application of PSO-enhanced machine learning algorithms, IDSs can achieve a balance between high detection performance and system efficiency, making them suitable for deployment in real-world IoT scenarios. Moreover, the integration of these techniques supports the development of scalable and generalizable security models capable of adapting to evolving attack patterns.

This thesis has addressed several key aspects related to enhancing security in IoT environments through intelligent intrusion detection. It began by presenting the fundamentals of the Internet of Things (IoT), including its architecture, characteristics, and communication protocols. Then, it explored Intrusion Detection Systems (IDS), their architectures, characteristics, and the common types of intrusion attacks they are designed to detect. The third part focused on the integration of Machine Learning (ML) techniques with IDS to improve detection accuracy and system adaptability. Finally, the Particle Swarm Optimization (PSO) algorithm was applied for feature selection, demonstrating its effectiveness in reducing feature dimensionality and improving classifier performance. Together, these components contribute to building a more efficient and

intelligent intrusion detection framework suitable for the dynamic and resource-constrained nature of IoT networks.

For future works, an integration of the proposed model could be done by trying to install the IDS in different approaches such as central, distributed and hybrid approach in a real environment with real traffic data.

In conclusion, the synergy between IoT, IDS, ML, and PSO represents a significant advancement toward the realization of autonomous, intelligent, and lightweight security frameworks. Future research may focus on real-time implementation, cross-layer optimization, and the incorporation of federated or distributed learning paradigms to further strengthen the resilience and applicability of these systems in next-generation IoT ecosystems.

References

[1] J.Gubbi , R. Buyya, Internet of Things (IoT): A vision, architectural elements, and future directions, 2013

[2] Acharjya, D.P.; Geetha, M.K., eds. (2017). Internet of Things: Novel Advances and Envisioned Applications. Springer. p. 311. ISBN 9783319534725

[3] O.Vermesan,P.Friess,internet of things-Converging Technologies for Smart Environments and Integrated Ecosystems

[4] Internet of Things - DiVA portal discusses the characteristics of IoT, including interconnectivity, heterogeneity, dynamic changes, and thing-related services

[5] Understanding IoT Networks: A Beginner's Guide by Device Authority highlights the role of interconnectivity and heterogeneity in IoT networks

[6] Completely Introduction of IoT Networks In 2024 by MineW provides insights into how IoT networks manage dynamic changes and adapt to different environments

[7] Zipit Wireless: Explains the four layers of IoT architecture, including device, communications, and cloud ingest, and application layers

[8] EITC: Discusses IoT architecture with layers like sensing, network, data processing, and application

[9] Device Authority: Highlights the perception, network, processing, and application layers.

[10] TechTarget: Describes six layers, including physical/device, network, data/database, analytics/visualization, application/integration, and security/management

[11] Vation Ventures: Provides an overview of IoT architecture with a focus on device, network, management, and application layers

[12] Scarfone, K., & Mell, P. (2007). Guide to Intrusion Detection and Prevention Systems (IDPS) (NIST Special Publication 800-94). National Institute of Standards and Technology

[13] Bace, R., & Mell, P. (2001). Intrusion Detection Systems (NIST Special Publication 800-31). National Institute of Standards and Technology

[14] IBM. (n.d.). What is an intrusion detection system (IDS)? IBM Security

[15] Imperva. (n.d.). Host-based Intrusion Detection System (HIDS). Retrieved from https://www.imperva.com/learn/application-security/host-based-intrusion-detection-hids/

[16] Cisco. (n.d.). What is a Network-Based Intrusion Detection System (NIDS)? Retrieved from https://www.cisco.com/c/en/us/products/security/what-is-an-intrusion-detection-system-ids.html

[17] Stamus Networks. (n.d.). What are the Three Types of IDS? Retrieved from https://www.stamus-networks.com/blog/what-are-the-three-types-of-ids

[18] Classification model for accuracy and intrusion detection using machine learning approach - PMC

[19] Vanchurin, V. (2022). Bio-inspired Machine Learning: programmed death and replication. arXiv preprint arXiv:2207.04886. Available at arXiv

[20] Darwish, A. (2018). Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications. Future Computing and Informatics Journal, 3(2), Article 9. Available at Digital Commons

[21] Oga.ai (n.d.). Bio-inspired optimization algorithms. Available at oga.ai

[22] Mitchell, Melanie. An introduction to genetic algorithms. MIT press, 1998

[23] ALBCOM, LSI, Universitat Politècnica de Catalunya, Jordi Girona 1-3, Campus Nord, 08034 Barcelona, Spain Accepted 11 October 2005

[24] Pitas I (2000) Digital image processing algorithms and applications. Wiley, New York

[25] Vandenbroucke N, Macaire L, Postaire JG (2000) Color image segmentation by supervised pixel classification in a color texture feature space. Application to soccer image segmentation. In: Proceedings 15th IEEE international conference on pattern recognition. ICPR-2000, vol 3, pp 621–624 (September)

[26] Holland J. H., Genetic algorithms, 1992, Scientific American, London, UK

[27] Strisciuglio, N. (2016). Bio-inspired algorithms for pattern recognition in audio and image processing. University of Groningen

[28] Kadry, S., & Kumar, T. K. (2025). Bio-inspired Algorithms in Machine Learning and Deep Learning for Disease Detection

[29] Satyobroto Talukder ”Mathematical Modelling and Applications of Particle Swarm Optimization”, Master’s Thesis. February 2011

[30] Anaconda.com, "Anaconda.

[31] Anaconda.com, "Anaconda Installation guide.

[32] Numpy.org. *What is NumPy*. Available: https://numpy.org/

[33] Pandas. *Pandas*. Available : https://pandas.pydata.org/

[34] https://matplotlib.org/. *Matplotlib: visualisation with python*. Available: https://matplotlib.org/

[35] Scikitlearn. *Scikitlearn*. Available: https://scikit-learn.org/

الملخص

تُعدّ الهجمات السبرانية تهديدًا حقيقيًا للأنظمة المعلوماتية، مما يستدعي الكشف المبكر عنها للحد من أنشطة الهكر وحماية البيانات الحساسة. من الضروري تأمين الأنظمة عبر تطوير وسائل ذكية تعتمد على تقنيات التعلم الآلي (ML) التي تمكّن من تحليل السلوك واكتشاف الأنماط غير الطبيعية. من أبرز هذه الوسائل أنظمة كشف التسلل (IDS) التي تستفيد من الذكاء الاصطناعي لتحسين دقتها. كما يمكن استعمال خوارزميات مستوحاة من الطبيعة مثل PSO (Particle Swarm Optimization) و GA (Genetic Algorithm) في اختيار الخصائص وتحسين أداء الكشف. هذه الأساليب تساهم بشكل فعال في رفع مستوى الأمان السيبراني. قمنا بإجراء سلسلة من التجارب باستخدام مجموعة من خوارزميات التصنيف تشملKNN ، LR، NB، SVM، MLP، RF، وDT، وذلك بهدف تحسين أداء نظام كشف التسلل من خلال استخدام خوارزمية PSO لاختيار الميزات. أظهرت النتائج أن تطبيق PSO ساهم في تحسين دقة التصنيف وتقليل عدد الميزات في معظم الحالات. من بين جميع المصنفات، حققت خوارزمية Decision Tree (DT) أفضل أداء، حيث سجلت دقة عالية مع عدد ميزات أقل مقارنة بالخوارزميات الأخرى، مما يدل على فعاليتها في بناء نظام كشف تسلل أكثر كفاءة وأقل تعقيدًا.

Abstract

Cyberattacks pose a serious threat to information systems, making early detection essential to limit hackers' activities and protect sensitive data. Securing systems requires the development of intelligent methods based on Machine Learning (ML) techniques that analyze behavior and detect anomalies. One of the most effective tools is Intrusion Detection Systems (IDS), which leverage AI to improve accuracy. Nature-inspired algorithms such as Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) can also be used for feature selection and enhancing detection performance. These approaches play a vital role in strengthening cybersecurity. We conducted a series of experiments using several classification algorithms, including KNN, LR, NB, SVM, MLP, RF, and DT, with the aim of enhancing intrusion detection performance through PSO-based feature selection. The results showed that PSO improved classification accuracy while reducing the number of features in most cases. Among all classifiers, the Decision Tree (DT) algorithm achieved the best performance, attaining high accuracy with fewer selected features compared to the other models. This highlights its effectiveness in building a more efficient and less complex intrusion detection system.

Résumé

Les cyberattaques représentent une menace sérieuse pour les systèmes d'information, rendant la détection précoce essentielle pour limiter les activités des hackers et protéger les données sensibles. La sécurisation des systèmes nécessite le développement de méthodes intelligentes basées sur l'apprentissage automatique (Machine Learning) permettant d'analyser les comportements et de détecter les anomalies. Parmi les outils les plus efficaces figurent les systèmes de détection d'intrusion (IDS), qui utilisent l'intelligence artificielle pour améliorer leur précision. Des algorithmes inspirés de la nature, tels que PSO (Particle Swarm Optimization) et GA (Genetic Algorithm), peuvent également être utilisés pour la sélection de caractéristiques et l'amélioration des performances de détection. Ces approches jouent un rôle crucial dans le renforcement de la cybersécurité. Nous avons mené une série d'expériences en utilisant plusieurs algorithmes de

classification, notamment KNN, LR, NB, SVM, MLP, RF et DT, dans le but d'améliorer la performance du système de détection d'intrusion grâce à la sélection de caractéristiques basée sur l'algorithme PSO. Les résultats ont montré que l'utilisation de PSO a permis d'améliorer la précision de classification tout en réduisant le nombre de caractéristiques dans la plupart des cas. Parmi tous les classifieurs, l'algorithme Decision Tree (DT) a obtenu les meilleures performances, en atteignant une précision élevée avec un nombre de caractéristiques sélectionnées inférieur par rapport aux autres modèles, ce qui démontre son efficacité dans la conception d'un système de détection d'intrusion plus performant et moins complexe.