AN ENHANCED DETECTION OF ADVANCED PERSISTENT THREATS USING MACHINE LEARNING

BY

ABDULLAH SAID ALI AL-AAMRI

A thesis submitted in fulfillment of the requirement for the degree of Doctoral of Philosophy in Computer Science

Kulliyyah of Information and Communication Technology
International Islamic University Malaysia

JANUARY 2023

ABSTRACT

Nowadays, countries are targeted by many electronic threats, which have expanded to reach almost all business sectors, either in private corporate entities or public institutions. Advanced Persistent Threats (APTs) are well-known examples of these electronic threats. APTs are very advanced and stealthy computer network attacks designed to gain unauthorized access to computer networks and remain undetected for an extended period. They represent one of the most critical cybersecurity challenges facing governments, corporations, and individuals. Since APT are categorized as the most critical cybersecurity threats, this study came to understand the nature of these attacks and propose a multi-stage framework to detect APT attacks based on the building on time series data. Unlike the previous model, the proposed approach can detect real-time attacks based on stored attack scenarios. This study has reviewed the background research, identified their strengths and weaknesses, and identified improvement opportunities. Moreover, available standardized techniques have been enhanced to detect APT attacks. Furthermore, the datasets used to feed the learning process are generated from different sources, including Journal logs, Traceability audits, and Systems monitoring statistics. Then, an effective APT detection and prevention system of Composition-Based Decision Tree (CDT) has been built/ developed/ implemented in complex environments. The results indicated that the proposed approach, on average, outperformed the existing algorithms reported in the literature. For example, the precision estimate of detecting whether the attack was malicious for the proposed model (CDT) was 96%, consistent with precision estimates by the existing algorithm: PRISM 96.9%, JRip 96%, and OneR 96%. However, the proposed model outperformed the existing algorithm when detecting whether the attack was benign. For example, the precision of CDT in this scenario was 50% compared to 0% for OneR, 10% for JRip, and 13.6% for PRISM. Overall, the average score indicates that the proposed model has outperformed the existing algorithms. For example, the average precision estimate for the proposed model was 94.3% compared to the existing algorithms, with values of 93.7%, 92.6%, and 92.1% for PRISM, JRip, and OneR, respectively. The evaluation of the CDT algorithm has been achieved by adopting the algorithm number 3 outputs to the NB Tree standard upon the WEKA software.

خلاصة البحث

في الوقت الحاضر، تُستهدف الدول بالعديد من أنواع التهديدات الإلكترونية، والتي توسعت لتصل إلى جميع قطاعات الأعمال تقريبًا، سواء في الشركات الخاصة أو المؤسسات العامة. وإذ تعتبر التهديدات الموجهة المتقدمة (APT) أحد الأمثلة المعروفة لهذه التهديدات الإلكترونية. APTs هي هجمات متطورة جدًا ومخفية لشبكات الكمبيوتر ومصممة للوصول لغير المصرح به إلى شبكات الكمبيوتر وتظل غير مكتشفة لفترة طويلة. حيث تمثل أحد أهم تحديات الأمن السيبراني التي تواجه الحكومات والشركات وحتى الأفراد. ونظرًا لتصنيف APT على أنها أكثر تهديدات الأمن السيبراني أهمية، فقد جاءت هذه الدراسة لفهم طبيعة هذه الهجمات، وإقتراح إطار عمل متعدد المراحل لإكتشاف هذه الهجمات تلقائيًا على بيانات السلاسل الزمنية. يمكن للنهج المقترح، بخلاف النماذج السابقة بالدراسات والبحوث، اكتشاف الهجمات في الوقت الفعلي بناءً على سيناريوهات مصممة لذلك. وأشارت النتائج إلى أن النهج المقترح، في المتوسط، تفوق على الخوارزميات الورادة بالدراسات السابقة. على سبيل المثال، كان تقدير الدقة لإكتشاف ما إذا كان الهجوم ضارًا لنموذجنا المقترح (CDT) هو ٩٦٪، بما يتوافق مع تقديرات الدقة بواسطة الخوارزمية الحالية: ٪96.9 PRISM و٪96 JRip و٪96 OneR. ومع ذلك، فقد تفوق نموذجنا المقترح على الخوارزمية الحالية عند اكتشاف ما إذا كان الهجوم حميدًا أم لا. ومن ناحية آخرى، كانت دقة CDT في هذا السيناريو ٥٠٪ مقارنة بـ ٠٪ لـ OneR و١٠٪ لـ JRip و١٣,٦٪ لـ PRISM. فبشكل عام، يشير متوسط الدرجات إلى أن نموذجنا المقترح قد تفوق على الخوارزميات الحالية. وكذلك كان متوسط تقدير الدقة للنموذج المقترح ٩٤,٣٪ مقارنة بالخوارزميات الحالية، بقيم ٩٣,٧٪ و ٩٢,٦٪ و٩٢,١٪ لـ PRISM وJRip وOneR على التوالي. كما تم تحقيق تقييم خوارزمية CDT من خلال اعتماد مخرجات الخوارزمية رقم ٣ لمعيار NB Tree على برنامج WEKA.

APPROVAL PAGE

The thesis of Abdullah Said Ali Al-Aamri has been approved by the following:

_____

Rawad Abdulkhaleq
Supervisor

_____

Shuhaili Bt Talib
Co-supervisor

_____

Imad Fakhri Taha Alyaseen
Co-supervisor

_____

Norbik B Idris
Internal Examiner

_____

Rabiah Ahmad
External Examiner

_____

Akram M Zeki Khedher
Chairman

DECLARATION

I hereby declare that this thesis is the result of my own investigation, except where otherwise stated. I also declare that it has not been previously or concurrently submitted as a whole for any other degrees at IIUM or other institutions.

Abdullah Said Ali Al-Aamri

Signature: ……... Date: ……...

INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA

DECLARATION OF COPYRIGHT AND AFFIRMATION OF FAIR USE OF UNPUBLISHED RESEARCH

AN ENHANCED DETECTION OF ADVANCED PERSISTENT THREATS USING MACHINE LEARNING

I declare that the copyright holders of this thesis are jointly owned by the student and IIUM.

By signing this form, I acknowledge that I have read and understood the IIUM Intellectual Property Right and Commercialization Policy.

Affirmed by Abdullah Said Ali Al-Aamri

…….. ……..

Signature Date

This thesis is dedicated to my family.

ACKNOWLEDGMENTS

All praise and thanks, first and foremost, to the Creator, Glory be to Him, who, without His mercy and success, would not have reached this stage and would not have achieved this outstanding achievement. Mainly, I was among those researchers affected by the Corona pandemic, which caused many delays and prevented me from meeting with the supervisor and the supervision committee, accessing information sources like libraries, and meeting face-to-face with experts. It was a difficult period, but with the grace of God, I overcame those difficulties and completed this dissertation with all merit.

My sincere thanks to my family members for being patient with me. They always motivated me and tolerated my negligence towards them during that period because I was too busy with research. I was looking for quantity and style to accomplish this modest research. Therefore, I ask God Almighty to benefit every researcher, learner, and reader.

I also extend my great thanks to my colleagues and brothers who stood with me and were a great incentive for my progress and achieving this goal, which I have been waiting for it several years.

I am truly indebted to the supervisors, Prof. Dr. Imad Fakhri Taha and Asst. Prof. Dr. Rawad Abdulkhaleq, for the direct support and supervision they provided me with for this work, especially their valuable observations and periodic follow-up through remote meetings during the Corona pandemic period. I also do not forget the role of Prof. Dr. Akram M Zeki, the head of the supervision committee and Asst. Prof. Dr. Shuhaili Bt Talib, the co-supervisor.

I also give special thanks to my dear brother Dr. Adil Al-Busaidi for his excellent cooperation and strong motivation for me while completing this work.

I also express my appreciation and thanks to anyone who offered their efforts, support, and time to help me in this research work.

We thank God Almighty for His success and guidance to me, and I ask God Almighty to enable me to finish writing this dissertation successfully. Praise be to Allah.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of Acronyms and Abbreviations

AE	AutoEncoder	KNN	K-Nearest Neighbors
AI	Artificial Intelligence	LAN	Local Area Network
ANN	Artificial Neural Network	LSTM	Long Short-Term Memory
API	Application Programming Interface	MAC	Media Access Control
APT	Advanced Persistent Threats	MATLAB	Matrix Laboratory
ARIMA	AutoRegressive Integrated Moving Average	NLP	Natural Language Processing
ARP	Address Resolution Protocol	NS	Network Simulator
ART	Adaptive Resonance Theory	OS	Operating System
AUC	Area Under the Curve	RF	Random Forest (RF)
AUROC	Area Under the Curve of the Receiver Operating Characteristic	RMSE	Root Mean Squared Error
CNN	Convolutional Neural Network	RNN	Recurrent Neural Network
CS	Computer Science	ROC	Receiver Operating Characteristic
DNS	Domain Name System	SDN	Software-Defined Networking
DDoS	Distributed Denial of Service	SIEM	Security Information and Event Management
FN	False Negative	SNMP	Simple Network Management Protocol
FP	False Positive	SPSS	Statistical Package for the Social Sciences

FPR	False Positive Rate	SSH	Secure Shell
FTP	File Transfer Protocol	SVM	Support Vector Machine
F1 Score	Harmonic Precision-Recall Mean	TCP	Transmission Control Protocol
GAN	Generative Adversarial Network	TN	True Negative
HIDS	Host-based Intrusion Detection System	TP	True Positive
IDS	Intrusion Detection System	TPE	Tree-structured Parzen Estimator
IIUM	International Islamic University Malaysia	TPR	True Positive Rate
IDES	Intrusion Detection Expert System	U2R	User to Root
IoT	Internet of Things	UDP	User Datagram Protocol
IPS	Intrusion Prevention System	VLAN	Virtual Local Area Network
ISO	International Organization for Standardization	VM	Virtual Machine
ISP	Internet Service Provider	XSS	Cross-Site Scripting
KDD	Knowledge Discovery and Data mining	WEKA	Waikato Environment for Knowledge Analysis
KICT	Kulliyyah of Information and Communication Technology		

CHAPTER ONE

INTRODUCTION

1.1. OVERVIEW

In recent years, intelligence has been accompanied by the ability to learn. With learning, an intelligent system that is able to carry out a task can improve its performance over time. Through training, it can learn to accomplish new tasks and self-improve based on its inherent capabilities.

Traditionally, creating an intelligent system involved manually writing a program to play chess (by using a tree search method), identifying printed characters (by comparing them to prototype images), or diagnosing medical conditions from symptoms (by utilizing logical deduction from rules established by experts). Nevertheless this "manual" approach has its limitations and could not survive in the current technology revolution. As well as, people nowadays do most of their tasks from one location and with one click without extra effort or need, for example, to do that separately from different sites. So, people live in an era in which computer systems play a prominent role in their lives with fast, easy, and trusted communication channels between service providers and consumers.

On the other side, the same situation is there for the technical or support team, which is facing an extreme increase in the daily cases of troubleshooting that need to be solved fast and flawlessly. In addition, some of these cases may require continuous working hours or days to sort out. At the same time, the end user will not be patient with them, or another issue might appear if that case does not close immediately. An example is if a vulnerability is explored in a network or a system and the response team does not take immediate action, then the attackers could steal data or shift the service down, which can suffer that organization a lot.

Additionally, people in the technical support field need an intelligent system to help accelerate their daily work to prevent, detect or solve any problems, including cyber threats. Therefore, identifying the vulnerabilities in a network is crucial in detecting any cyber-attack. Additionally, the defense and monitoring team must have a clear understanding of the motivation behind the attack and the specific information that is being targeted. With this knowledge, appropriate and effective plans can be developed to counter the attack (Al-Mohannadi et al., 2016).

An Advanced Persistent Threat (APT) is a protracted and embattled cyber-attack in which an unauthorized individual gains access to a network and remains undetected for an extended period. APT attacks typically focus on monitoring network activity and stealing data rather than causing harm to the network or organization. These attacks are often directed toward national defense, finance sectors, and, the industry where sensitive information such as intellectual property, military plans, and data from government agencies and companies is of paramount importance.

Additionally, APTs are typically designed to establish persistent access to the targeted network rather than quickly entering and exiting. Due to the significant resources and effort required to execute APT attacks, hackers tend to select high-value targets, such as nation-states and large corporations, with the intention of stealing valuable information over an extended period.

APT groups often employ advanced tactics to infiltrate their targets, such as utilizing "zero-day" vulnerabilities, spear phishing, and further social engineering procedures. They utilize multiple methods to maintain access to the targeted network without detection. To do this, they must constantly modify their malicious code and implement advanced concealment techniques. Some APT threats are so complex that managing compromised systems and software requires a dedicated administrator.

There are various motivations for hackers to carry out persistent, advanced threats. State-sponsored hackers may aim to steal intellectual property in order to gain a competitive edge in certain industries. Industries such as energy distribution, telecommunications, infrastructure systems, social networks, media companies, and political targets are often targeted. Criminal organizations may also use APT threats to acquire information that can be used to commit criminal acts for financial gain.

Detection of threats or a virus based on signatures is relatively straightforward as it relies on comparing it to the signatures in the anti-virus database. This process simply requires updating the anti-virus program's database, and the appropriate action is taken based on the comparison. However, detecting advanced attacks is more complex as it relies on identifying unusual behavior rather than a specific signature (Ghafir et al., 2018).

While APT attacks can be difficult to detect, data theft is not always entirely undetectable. One of the main indicators of an APT attack is the exfiltration of data. To detect these types of attacks, cybersecurity professionals focus on identifying anomalies in outgoing data, in order to determine if the network has been compromised by an APT.

1.2. PROBLEM STATEMENT

Transiting sensitive information and data across digital environments like smart cities raises concerns about digital security and data assurance. This is particularly challenging in ad-hoc architectures, impacting secure data transmission. Organizational information security policies significantly influence employee awareness (Young et al., 2017).

Digital content delivery's importance in smart cities is recognized by commercial companies. This research delves into this aspect, focusing on dedicated communication systems and APT attack interfaces (Alshamrani et al., 2019).

Creating an intelligent environment necessitates a robust system based on secure architecture, incorporating machine learning analytics and specialized hardware.

Monitoring system performance over time is vital to assess its acceptability and success in society. AI's growth in digital commerce is notable, especially in scientific services. This research establishes constraints to measure parameter changes, contributing to technological advancement.

In summary, this study addresses challenges in digital security, data transmission, and content delivery in smart cities, aiming to enhance technology for societal benefit. Herewith is a summary of the problem statement for this study as following:

I. Problem 1: Lack of Pattern-Based Detection
II. Problem 2: Miss-feeding AI standards to learn detecting APT patterns
III. Problem 3: Insufficient Analysis of threat Events

1.3. RESEARCH QUESTIONS

Research questions are designed to provide answers and solutions to a specific problem and serve as a guide for conducting a literature review throughout the research process. The following research questions have been formulated in this study:

I. What are methodological issues that could inhibit the detection of APT attacks?
II. How to enhance the existing standardized techniques aimed at detecting APT attacks?
III. How to develop an efficient autonomous APT detection model?
IV. How to compare and evaluate the results of the autonomous APT detection model against the currently used algorithms?

1.4. RESEARCH OBJECTIVES

Mobile devices' mobility means global mobility of the sensor nodes, which gives more scalability to the system in the sense of availability and response to live requests. The network scan for suspicious data at rest (S et al., 2012) also works well when looking at a non-structured network. Because the network's structure is not envisioned, it is easy to be configured and monitor as a non-structured ad-hoc network (Chamola et al., 2021).

Recent studies demonstrate that Nano Electromechanical Systems (NEMS) are implemented to enhance the digital content delivery within data networks (Wichlund, 2012) and are implemented on tiny microchips as physical components adjusted, software-wise and hardware-wise. So, when dealing with these systems; cause the size of the end sensor is a question that depends on case-by-case when deployed (Chatterjee et al., 2011). All of these added values are an added complexity to the research.

Thus, the main objectives for this research are mentioned below:

I. To identify methodological issues that could inhibit the detection of APT attacks.
II. To enhance the existing standardized techniques aimed at detecting APT attacks.
III. To develop an efficient autonomous APT detection model.
IV. To evaluate and compare the results of the autonomous APT detection model against the currently used algorithms.

1.5. SIGNIFICANCE OF RESEARCH

Contemporary computer and network systems have improved their ability to withstand internet threats and attacks. Nevertheless, numerous organizations continue to depend heavily on human personnel for network security (Weems et al., 2018).

This research makes contributions both methodologically and technologically, as well as in practical aspects, to achieve its objectives.

I. Methodological and Technological Contributions:
    - To propose a framework that functions as a building block to fully implement active and autonomous enhanced algorithms that can detect APT attacks.
    - To design an autonomous feeding up SNORT rules for APT detection.

- To develop a highly prediction model for APT attacks with more accuracy levels.
- To design an algorithm to detect the threat while analyzing data at rest.
- To develop an intelligent self-secure network.

II. Practical Contribution:

- To implement the proposed APT detection methods with fewer costs and higher levels of accuracy.
- To aware professionals and cybersecurity experts regards potential security threats.
- To highlight the importance of machine learning technologies for improving organizational IT security-related practices.
- To bring to the attention of the decision-makers regards 4^{th} industry revolution.
- To provide the world's nations the ability to create effective national global secure digital environments.

1.6. THE SCOPE OF RESEARCH

In this research, the main focus is going to be on as follow:

1. This study will use penetrating testing tools from different Operating Systems (Linux, Windows, Android, IOS, etc.).
2. A scenario of attack and exploit will be there (social engineering tools to inject an APT threat; and Platforms like Pineapple on Kali or WiFiSlacks to broadcast).
3. The APT tool will be developed by adapting famously recognized techniques and customized Algorithms to enlarge the range of prevention and detection success.

1.7. ORGANIZATION OF THE THESIS

This thesis has a total of five chapters. The first chapter consists of the introductory section that gives an essential background of the study, problem statement, research questions, objectives, and the work scope. Chapter two is about the Literature review, which provides previous research related to the research topic. It explains previous studies and the implementation of AI in detecting APTs. It also discusses previous methodologies, AI algorithms, and techniques used to countermeasure APTs attacks.

Chapter three is about the research methodology and workflow design used to achieve this research's four objectives.

Chapter four covers the research experimental setup, design, and implementation. This part shows the experimental results after the analysis.

Chapter five is the results and discussion of this research, where the study's implications are discussed based-on the experiments final results.

Chapter six is the conclusion and future work, where future recommendations for other researchers are endorse to carry on the research improvement.

CHAPTER TWO

LITERATURE REVIEW

2.1. INTRODUCTION

In the following few paragraphs, an explanation of the Advanced Persistent Threat (APT) will be based on previous studies and the thesis. It consists of the characteristics and the meaning of APT. It also includes stages of APT which attackers rely on to reach their goals. In addition, APT approaches and some tools on which attackers may rely on APT attacks have been discussed.

The authors (Chatterjee et al., 2011; Grayver & Utter, 2020; Olson et al., 2016) have pointed out that sensor networks are considered vital tools, particularly in vulnerable industries such as petroleum exploration and extraction, which are managed through dedicated communication platforms. However, this research did not specifically address the security concerns and measures necessary to create a secure environment for these tools.

Furthermore, the research findings have shown that technical advancements, particularly in communication platforms, have led to the development of high-performance digital processors (Tornai et al., 2016). Consequently, it requires advanced and fast systems to keep up with these high performances in order to utilize these technologies fully. This research paper compares previous studies and enabling technologies that assist in building communication platforms. It also examines the architectures of a specific communication platform and how services are delivered through standardized layers.

2.2. ADVANCED PERSISTENT THREATS DEFINITION

In recent years, cyber warfare and the rise of the Internet of Things (IoT) have become significant factors in the increase and diversity of cyber-attacks (Conti et al., 2018). Despite the efforts of security services to detect these attacks, cybercriminals continue to develop new and advanced methods and technologies to attack networks and telecommunications (S. Singh et al., 2019).

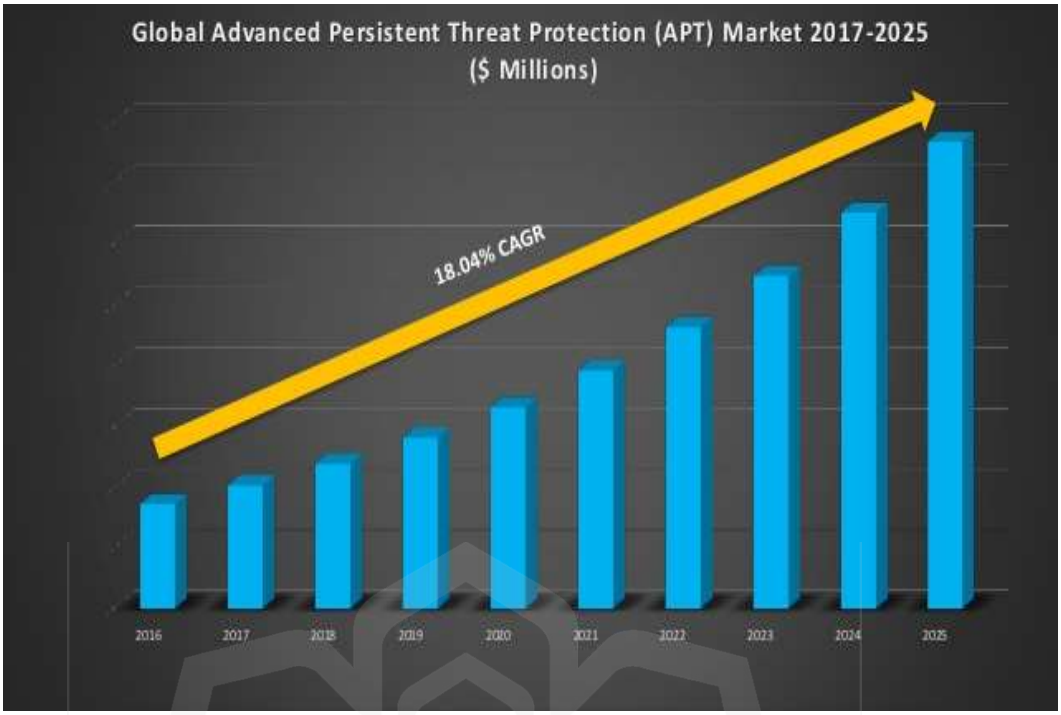


Figure 2. 1 Revenue Growth APT Protection Market 2017-2025[1]

As per Trend Micro's technical report, many companies that provide defense mechanisms against cyberattacks consider these attacks to be random and primarily target those who are not well-protected. Additionally, as per the Figure 2.1, the predicted revenue growth of the APT market between 2016-2025 is expected to grow at a CAGR of 18.04%. However, this assumption is incorrect because of the emergence of Advanced Persistent Threats (APT), as the attackers in this type of attack have specific targets before launching the attack (Ghafir et al., 2018).

Advanced Persistent Threat (APT) is considered one of the most serious types of attack in the cyber landscape due to its complexity, diversity, and unpredictable stages. Furthermore, one of the key factors that allows attackers to gather large amounts of information and access protected data is that these attacks can potentially go undetected within a network for an extended period (Marchetti et al., 2016).

As per (Marchetti et al., 2016; Siddiqui et al., 2016), APT could be defined as follows:

---

[1] https://www.slideshare.net/sherrythomas13/advanced-persistent-threat-protection-market-global-industry-analysis-report-20172025

Advanced (A): The attackers possess advanced skills in hacking and are part of a group with knowledge of various methods of piracy, and receive support from unknown sources that aid in creating multiple attack tools and strategies.

Persistent (P): The attackers are determined to reach their goals and use multiple techniques that follow a "low and slow" approach to increase their chances of success.

Threat (T): The attackers aim to steal intellectual property and cause damage to the victim.

Generally, APTs are challenging to detect due to their stringent security and technical measures, and are characterized by geographical distance and lack of time constraints. They are also characterized by their ability to easily conceal evidence and quickly retreat after achieving their goals, with the ultimate goal of causing severe economic and political damage to countries worldwide (J. Choi et al., 2015).

2.3. APT ATTACKS PHASES

APT attacks employ multi-step methods to achieve their goals successfully. Previous studies have shown that understanding these stages can help identify the target of the attacks, which in turn can aid in developing a robust defensive mechanism (Alshamrani et al., 2019). Therefore, the technologies used at each stage should be disclosed to enable the development and innovation of machinery that can counter APT attacks (Ghafir et al., 2018).

Previous research has outlined various models of APT stages, but they all convey a similar sequence of operations (Alshamrani et al., 2019). For example, previously some studies divided these stages into three phases (Yan et al., 2020a), while others divided them into five or seven stages (Brogi, 2018). But in this research, five debated steps are described in an article (Alshamrani et al., 2019), Figure2.2 expressing this steps graphically.

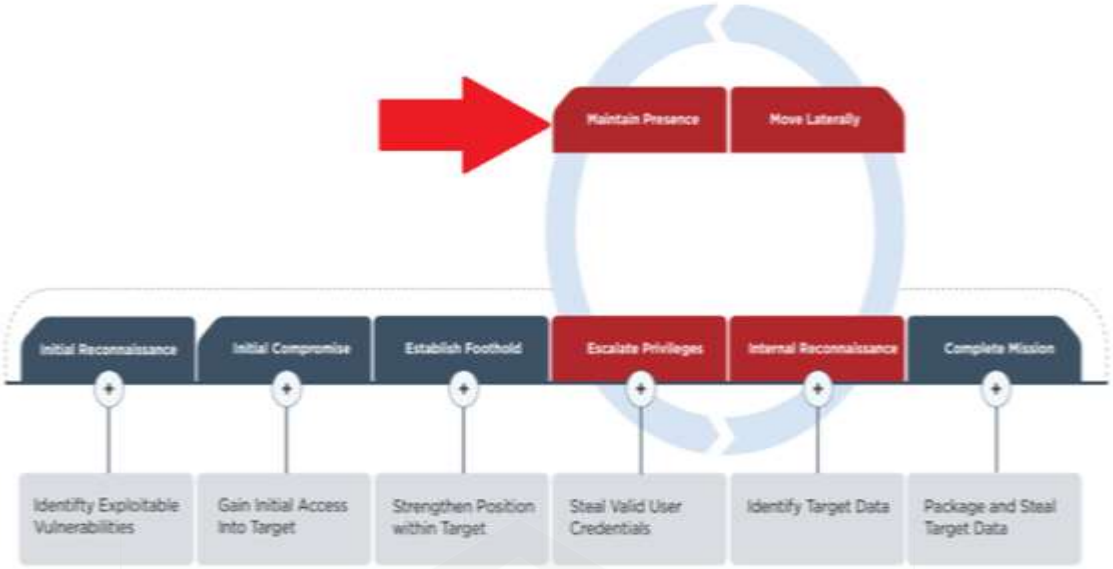


Figure 2. 2 APT main Steps for attacking the network[2]

2.3.1. Reconnaissance

The initial step in APT attacks is to conduct reconnaissance on the victim to gather information about employees and the critical IT infrastructure of the targeted victim. This can be achieved through network scanning techniques and Border Gateway Protocol (BGP) or via social networks or online services (Alminshid & Omar, 2020).

2.3.2. Deliver and Establish Foothold

The second step of an attack, after gathering information about the victim, is to successfully penetrate the target network. This can be achieved through phishing mechanisms, by sending malicious software through email for the victim to open an attachment or click on a web link, or through external devices such as CDs or USBs. Additionally, attackers may exploit Zero-day vulnerabilities or employ Watering-Hole attacks (Alshamrani et al., 2019).

Moreover, attackers will use various tools, such as Fuzzer software, to search for any security vulnerabilities in the victim's network. This allows them to enter the network undetected and without causing any disruptions (Herløw & Hansen, 2015). Once they have successfully entered the network, APT attackers will establish a

[2] https://www.fireeye.com/blog/threat-research/2019/09/sharpersist-windows-persistence-toolkit.html

communication channel known as Command and Control (C&C) to control the compromised systems and establish their presence in the network using techniques such as Domain Name Systems (DNS) manipulation (Alshamrani et al., 2019).

2.3.3. Lateral Movement Stay Undetected

In this stage of an APT attack, the attackers gain access to the victim's infrastructure and move deeper into their systems and networks. They then proceed to gather data, which is typically done by encrypting and decrypting it using various techniques and methods to transfer it between systems and obtain sensitive information (Alminshid & Omar, 2020). This data gathering process is a crucial step in the APT attack, as it allows the attackers to obtain the information they need to achieve their goals (Alshamrani et al., 2019).

At this point, APT attackers use various tools, such as the local password dump tool and the mimikatz tool in order to obtain passwords used to protect corporate data and employee personal info. Furthermore, APT attackers create multiple connections from C&C to detect any modification to network infrastructure security policies (Alminshid & Omar, 2020).

2.3.4. Exfiltration Impediment

In this stage, APT attackers transmit the information and data they obtained to their Command and Control (C&C) server using various IP addresses to evade detection and tracking (Ussath et al., 2016). They use the fast flux technique to achieve this (Alminshid & Omar, 2020).

2.3.5. Post-Exfiltration and Post-Impediment

This step is the stage of concealment, where APT attackers remove any traces or tools used during the attack to avoid detection (Alshamrani et al., 2019).

2.4. APT ATTACKS APPROACHES

2.4.1. Social Engineering

It is a technique that attackers use to manipulate users psychologically in order to make them commit security mistakes or reveal sensitive information. This method is considered one of the simplest and oldest techniques for gathering reliable data (P. Li et al., 2018).

2.4.2. Spear Phishing

Attackers usually send a legitimate message with attachments containing malware to trick a previously identified user (Caldwell, 2013). It is a sophisticated method where an email is sent which contains a link that entices the victim to click on it, and it does not require attachments (Bhadane et al., 2019).

2.4.3. Rootkits

It is seen as a harmful program that is concealed within the victim's systems. Once installed on any system, attackers can move between systems on the network. This tool is considered one of the most hazardous tools on the victim's network as it is challenging to detect. It can also acquire a complete snapshot of the victim's desktop computer, record keystrokes, and enable the downloading of files (Bhadane et al., 2019; Jagadale et al., 2017; Prodanovic et al., 2017).

2.4.4. Zero-Day Exploits

APT attackers often use zero-day vulnerabilities, which are previously unknown weaknesses in software or systems, to gain access to victims' networks. These types of attacks can go unnoticed for a long period, as the vulnerabilities have not yet been discovered or addressed. Zero-day exploitation is a primary tactic used by APT attackers to infiltrate a network and maintain access for an extended period. (Sahabandu et al., 2019).

2.4.5. Encrypted Network Communication

The APT feature allows attackers to evade detection by commonly used anti-virus software. They also use encryption to access the target victim's network servers and transfer the data to their Command and Control server (Young et al., 2017; Alshamrani et al., 2019). This method also helps them stealthily exfiltrate sensitive data during the APT exfiltration step.

2.4.6. Fast Flux Exfiltration

At the exfiltration step, APT attackers use the Fast-Flux method to extract sensitive data from the target network (Paganoni et al., 2022). This method involves using DNS to

hide malware activities and link them to APT servers (C&C) (Gu et al., 2022; Xiong et al., 2022). This allows the attackers to covertly transfer data without detection.

2.4.7. Drive by Download Evasion

The Drive-by Download Evasion technique makes it challenging to trace the source of APT attacks and analyze them (Avgerinos et al. 2014). This technique is characterized by three elements:

1. Utilizing a technique that redirects victims to various malicious sites repeatedly (Bahrami et al., 2019; Caviglione et al., 2021).
2. Using fingerprint technology to analyze the browser used by the victim (Bahrami et al., 2019; Caviglione et al., 2021).
3. Implementing obfuscation to encode JavaScript code and conceal malware from web scanners (Gu et al., 2022; Xiong et al., 2022).

2.4.8. Avoid Network Packet Manipulation Detection

This technique is used to evade detection by disguising itself as normal network activity, making it difficult for security systems to identify the malicious activity (Bahrami et al., 2019; Caviglione et al., 2021). By manipulating the NDIS protocol, the attackers are able to conceal their actions and evade detection by network security systems (Xiong et al., 2022). In general, many corporations focused on the security field conducted studies to evaluate several features to measure the maturity of the security awareness program in societies to reduce attacks and generate a security environment for small and large organizations.

In the past, many companies did not prioritize security awareness, viewing it as solely the responsibility of employees. They also lacked the resources to effectively educate employees about the dangers of cyber-attacks. (SANS Security Awareness Report, 2018). According to previous research, several methods have been proposed to improve security awareness within organizations:

1. Implementing a mandatory training program for employees on the various tactics used by attackers to exploit them (Boutaba et al., 2018).
2. Encouraging employees to report any suspicious activity they may encounter to the Information Security Team (Boutaba et al., 2018).

3. Hiring qualified and experienced information security professionals who have a thorough understanding of the organization's network infrastructure and are aware of the current cyber threats (Boutaba et al., 2018).
4. Utilizing advanced security technologies such as Next Generation Firewall (NGFW) and Deep Packet Inspection (DPI) to monitor the organization's network (Boutaba et al., 2018).
5. Allocating an annual budget for security awareness initiatives (SANS Security Awareness Report (2018).

2.5. APT ATTACKERS TOOLS

APT attackers use a variety of tools to carry out their attacks and achieve their goals. Some examples of these tools, according to (Young et al., 2017; Alshamrani et al., 2019) include:

1. LSB Steganography: A technique used to conceal harmful information and files within images.
2. Netbox: A tool used to manage, track and document network and infrastructure details, allowing the attackers to repeatedly target and gain access to the systems.
3. Truesec Lslsass: A tool used to obtain the victim's password, making it easier for the attackers to navigate the targeted company's systems.
4. HUC Packet Transmit Tool/HTran: Tools used by attackers to redirect TCP traffic to the harmful C&C server after gaining access to the victim's network and systems.

2.6. ENTHUSIASTIC ANALYSIS FOR APT DEDICATED

Every day, data gets enlarged, and data stores management systems work unremittingly with high efficiency to have classified and useful data that can be manipulated and retrieved easily while needed. So, the technology has to be more complicated and critical simultaneously when analyzing and managing this data, especially for use in predictive purposes or even real-time ones. In addition, this kind of data processing technology can manipulate different data types, like in motion or at rest. The following sub-sections explained these types in detail, as shown below.

2.6.1. Data at Rest

Data at rest is defined as static data, which is stored in any physical media type like a mobile device or storage, and it is not moving from that media to another or between networks. So, securing this type of data and protecting it from attackers' modifications, wraps, or accessing, who typically find it more valuable than the live data, has to be highly considered (Marchetti et al., 2016).

In these circumstances, data is mainly protected by using network appliances such as firewalls plus anti-virus programs. However, these solutions are not impenetrable enough, so additional security layers must be there, like encrypting media drives or saving these data in multiple locations.

2.6.2. Data in Motion

The motion data is known by the live or active data which is available to reach through the Internet or inside the local network. Actually, this data is easy to breach by attackers if it is not well protected and monitored (Marchetti et al., 2016). Electronic services like online applications and emails are the best examples of this kind of data.

2.6.3. Former Realizations

The artificial intelligence approach has demonstrated malware detection capabilities from different perspectives. Yet, the standardized practices frequently demonstrate limitations toward the up-to-date malicious intent computer malware presents. The following Table 2.1 draws a clear view of other technologies work to capture APT in different scenarios with additional APT capabilities.

To visually illustrate the limitations posed by various applied technologies in each reference, a specific category named "Proportional Limitation Readings" was introduced. This category showcases accuracy values proportionally attributed to the outcomes of each machine learning technology as discussed in the research papers. These equilibrium readings are then graphically displayed, aiding in comprehending the constraints that each technology encounters.

Table 2. 1 ML APT Detection Technical Limitations

Ref	Algorithm	Approach	Approach Detail	Detail APT Life Cycle Used	Proportional Limitation Readings	LiA
(Ghafir et al., 2018)	SVM, k-NN, Decision Tree and Ensemble learning	AI-ML for APT	Alerts association Threats exposure Attacks forecast	six phases	182	
(Sharma et al., 2020)	SVM, GPC, Tree Regression, and dynamic Bayesian model	DoubleClick for Advertisers	Network flow packets Events Correlation Voting service	unspecified	15	
(Siddiqui et al., 2016)	dimension fractal D2 k-NN	D2	pcap Feature extraction Removing Noise ML Anomaly	unspecified	64	
(Shenwen et al., 2015)	k-NN	Big Data Detection	4 Rs (reduce, reuse, recycle,recover)	unspecified	-2	
(Y. Li et al., 2019)	DT, LB, LR, RF and GNB	RDP Detection	Dataset preprocess, Data Measure, ML algorithms	One phase	1	
(Chu et al., 2019)	MLP, NB, SVM, PCA, and DT	APT attack early discovers	Preprocess Dataset Classification	unspecified	30	
(Yuan Wang et al., 2019)	Fuzzy clustering	APT states	Dataset preprocess threats classification Fuzzy clustering	Four Phases	-2	

From another perspective, theories and research demonstrate different definitions than what the actual implementations are achieving. From this view, there were many machine learning studies and researches around data analysis, shown in Table 2.1, in order to detect the anomalies regardless of data types or condition. Deep learning stack is one of the proposed studies that is used for detecting and classifying data using a sequential neural network model. This approach used more than one algorithm simultaneously for detection in order to get highly accurate results. Bodström mentioned that: “this approach, the stack used to extract the data by classifying the available data and the unknown one or anomalies. Then, A comparison of previous outliers and identification of their connections within the network traffic flow is being conducted.” (Bodström & Hämäläinen, 2019a) Actually, this approach is good enough, but it dealt only with data at rest and did not cover the data at motion, which most of the attackers are hidden there.

To add, Berrada said that: “APT is hard to detect via investigating system-level activity by humans because normally systems generated a huge number of logs and events so difficult to analyze each one deeply and know every detail in a short time. In

this study, an unsupervised streaming anomaly detection algorithm using ML is proposed to detect APT attacks generated by four different operating systems in a concise time with highly efficient and accurate results. This approach identified abnormal activity of APT based on process behavior, intensely shrinking the monitoring problem to a low level, but the way or technique used for deployment this approach was not properly implemented and its consequence as well inaccurately." (Berrada et al., 2020)

Traditional incident response techniques, such as IDS and anti-virus, are considered conservative defense mechanisms that focus on identifying vulnerabilities in the network. They are inadequate for sophisticated computer network intrusions like APT because it represents well-resourced and qualified rivals. In order to achieve its objectives, APT uses advanced tools and techniques that can bypass traditional network defense methods like IDS and anti-virus. Experts believe that the shift towards new defense tools must involve the implementation of artificial intelligence techniques to analyze network traffic, identify threats, and develop effective countermeasures, including proactive measures (Trifonov et al., 2017). This study lacked behavioral validation because it depended on the anomalies behavior on the network flow which APT can stay hidden in the system and not only on the network stream. Table 2.2 summarizes some studies that used AI and ML to deal with APT but need some enhancement, so in this study, we will cover them.

Table 2. 2 ML Technologies for APT detection deficiency

	AI Impact	Experiment Analysis (Theories) or Operational Fields	Methodological Issues
(Marchetti et al., 2016)	• Multi-Factor Method for Big Data Analytics	• Regular access patterns • DGA analysis • Blacklist filtering	• Data at rest analysis
(Patil, P. , 2016)	• Applications of Visual Nets can Keep on in Cyber Security	• Visual nets • Expert systems • Intelligent agents • Searching • Learning • Constraint finding	• Data flow sampling
(Trifonov et al., 2017)	• Operational Intelligence Studies	• Doctrine of Active Defense. • Suitable features derive for behavioral validation. • Building a classifiers ensemble based on trained module	• Network flow behavior vs. anomalies detection
(X. Li & Jiang, 2017)	• Technology Framework Upon Four Layers	• AI models and algorithms • AI Techniques	• Data flow sampling • Data flow capturing
(Poola, 2017)	• AI & ML for Social Structures	• Machine Learning and Cognitive Systems • Supervised Learning Algorithms • Artificial Neural Networks • Algorithms and Complex Optimizations	• Systematic behavior and anomalies detection
(Adams et al., 2019)	• ML on User Workstation	• detect APT behavior as an anomaly • Red Team Automate (RTA)	• Flow Data sample, and anomalies detection
(Ghafir et al., 2018)	• Novel ML Entitled ML-APT	• Alert correlate, Threat detection, and Attack forecasting	• Data at rest and penetration alerts
(Rad et al., 2018)	• CNN Implementation Classifier of Binary Malware	• Windows Portable Executable (PE) classification. • ML for malware detection • SVM • GPU utilizing by Python	• Sampling & Classification
(Berrada et al., 2020)	• Unsupervised Streaming Anomaly Detection Algorithm Based on Process Behavior	• data mining provenance to analyze system activities • Attribute Value Frequency (AVF) for streaming anomaly detection technique	• Deployment Techniques

APT attack detection methods that have been researched in the past have limitations, such as being unable to detect the attack in real-time and having a high rate of false attack detection. APT attacks utilize advanced methods to stay undetected for extended periods, making them difficult to detect and remove, which traditional IDS cannot detect. These methods typically rely on patterns or signatures and use pre-set rules to detect advanced threats like APT. Researchers have found that machine-learning techniques are highly effective and accurate in detecting APT attacks (Ashrafuzzaman et al., 2018; Brogi, 2018; Gu et al., 2022; M. J. Zhao et al., 2018).

A literature review of studies on using ML for detecting APTs found that many approaches use a combination of features, such as network traffic data, system logs, and user behavior, to train an ML model. Some studies have used supervised learning methods, such as decision trees and random forests, while others have used unsupervised methods, such as clustering and anomaly detection.

Some studies that used unsupervised learning to detect APTs found that it could accurately identify known APT attacks and discover previously unknown attacks. Other studies that used supervised learning found that it achieved high accuracy in detecting APTs, but struggled with balancing false positive and false negative rates.

Overall, the literature suggests that ML can be an effective tool for detecting APTs as well it can analyze large amounts of data and identify patterns that indicate an attack. But, the choice of features, algorithms and evaluation methods will depend on the specific attack scenario. It is also important to consider the trade-off between detection accuracy and false alarms.

Horng et al. (Horng et al., 2011) presented an intrusion detection system that uses an SVM-based and triple-structured approach. The proposed system includes the use of feature selection, SVM, and a hierarchical clustering algorithm. The clustering algorithm is used to provide more relevant training examples to the SVM, which reduces the training time and improves the performance of the SVM. Therefore, an enhancement to the SVM model by incorporating a feature selection process to remove unimportant features from the training set, resulting in a more accurate system for detecting Dos and Probe attacks. They reported that this method had a higher overall accuracy rate of 95.72% compared to other intrusion detection systems. However, it is unclear if they have tested the technique on different types of datasets or network environments and if the results are robust and generalizable.

Moreover, the NSL-KDD dataset has been studied by Salama (Salama et al., 2011) to investigate the advanced threats. It has combined a Deep Belief Network (DBN) technique with Support Vector Machine (SVM). As a result, a percentage of 92.84% was obtained through a 40% dataset training, which provided better accuracy and performance compared with the non-combined DBN and SVM. To conclude, this study was done on the non-live extracted dataset.

In addition, a feature selection study was done by (Chae et al., 2013) for IDS using the NSL-KDD dataset. This study was mainly done to extract the inappropriate and redundant features resulting in a lengthy detection process and degrading the performance of an intrusion detection system. IG (Information Gain), CFS (Correlation-based Feature Selection), and GR (Gain Ratio) were evaluated to build an effective and efficient performance IDS. The proposed method used the decision tree classifier algorithm to assess the feature reduction method. After applying this method for R2L, Probe, DDos, and U2R, a comparing result shows that this method is more efficient for IDS. Also, an inverse correlation between accuracy and Attribute Ratio (AR) up to 22 features was found which got 99.794% compared with 99.763% when selecting all features. CFS accuracy with 25 features, IG with 23 features, and GR with 19 features have got 99.781%, 99.781%, and 99.794% respectively. This study did not take in consideration the time which is the main parameter in exploring the threats. Also, the classifier based on True and False Positive rates (TPR and FPR) but they are not declared here.

The study conducted by Chu and his team (Joshi et al., 2014) used a specific dataset and employed a specific method, Principal Component Analysis (PCA), to reduce the size of the dataset. They also found that using the Support Vector Machine (SVM) algorithm with a radial basis function (RBF) kernel had a better performance for detecting attacks, resulting in a detection accuracy rate of 97.22% compared to other algorithms such as multilayer perceptron (MLP), J48, Naive Bayes and decision tree. So, it would be helpful to know if the proposed method can detect different types of attacks or only specific ones.

Furthermore, a study conducted by Aziz et al. (Aziz et al., 2014) utilized a multi-layered hybrid machine learning based IDS, which employed various classification algorithms such as Decision Trees, Naive Bayes, and Multilayer Perceptron Neural networks to detect anomalous traffic. They used the NSL-KDD dataset for their research and found that the Naive Bayes classifier had the highest accuracy, while the Decision

Tree classifier (J48) had a detection rate of about 65% for probe attacks and 82% for DoS attacks. This study has investigated the flow in the anomaly traffic where a persistent threat can be founded in either anomaly traffic or while data at rest.

In this study, Hasani, Othman, and Mousavi (Hasani et al., 2014) proposed a combination of Linear Genetic Algorithms and Bee Algorithms as a method to enhance the detection rate of cyber threats and decrease the rate of false alarms. They found that the Support Vector Machine (SVM) algorithm was effective in handling issues related to intrusion detection systems. The research used a dataset from which four sample datasets containing 4000 random records were extracted. They found that the feature subcategory provided by LGP_BA offered a superior representation of the data. The classifier J48 had a detection rate of about 65% for probe attacks and 82% for DoS attacks. Nonetheless, it should be noted that the study only used the SVM algorithm, and incorporating additional algorithms may further improve the results.

In their study (Schmidhuber, 2015), proposed a hybrid approach that combines the SVM and GA algorithms. They reported that this approach resulted in improved accuracy (98.33%) compared to the standalone SVM algorithm. One disadvantage of this approach is that it may be computationally expensive to run the GA algorithm in addition to the SVM algorithm, which could result in longer processing times. Additionally, the hybrid approach may be more complex and difficult to implement compared to a standalone algorithm.

Additionally, (Ingre & Yadav, 2015) have evaluated NSL-KDD dataset performance to detect persistent threats using Artificial Neural Network (ANN). The obtained rate of intrusion detection was 81.2% and the accuracy of 79.9% for an attack-type classification. Also, the approach achieved a higher detection accuracy rate reaching 75.49% compared to five classes (i.e. Probe, DoS, U2R, R2L attacks, and normal status) and the Self-Organization Map (SOM) method. The study did not go deep into supervised ML algorithms which could help to increase the accuracy percentage and to have a higher detection rate.

In this study, Singh, Kumar, and Singla (R. Singh et al., 2015) developed a method for intrusion detection that utilizes the Online Sequential Extreme Learning Machine (OS-ELM) algorithm. They used alpha and beta profiling to decrease the time complexity of irrelevant attribute selection and reduce the size of the training dataset. Their results indicated that the proposed technique was effective in detecting network attacks, achieving high accuracy, low false positive rates, and fast detection times. This

technique may not be able to handle large and complex datasets effectively, as it relies on profiling of alpha and beta to reduce the size of the dataset and the time complexity. Additionally, the technique is based on a specific machine learning algorithm, the Online Sequential Extreme Learning Machine (OS-ELM), which may not be suitable for all types of datasets and applications. This may limit the generalizability and applicability of the technique in different contexts and environments.

Gupta and Shrivastava (M. Gupta & K. Shrivastava, 2015) proposed a method that utilizes a combination of Bee Colony and Support Vector Machine (SVM) algorithms to improve the accuracy of intrusion detection. The results of their experiments showed that the proposed solution had an average accuracy rate of 88.46%. Actually, this approach may not be as accurate as other methods, as an 88.46% accuracy rate is relatively low compared to other studies that have reported accuracy rates of over 90%.

Perez and his colleagues (Perez et al., 2017) proposed a method for detecting intrusions using machine learning algorithms like SVM, J48, Naive Bayes, and Decision Tree to classify the KDD-99 dataset. They employed the Information Gain technique for feature selection. Their findings indicate that the J48 Decision Tree method combined with the AdaBoost method achieved the highest accuracy rate of 97%, outperforming other methods. However, a limitation of this approach is that it is only tested on the KDD-99 dataset, which may not be representative of other datasets or network environments, so it may not be widely generalizable. Additionally, the use of only one feature selection method (Information Gain) and a limited number of algorithms may also limit the effectiveness of the approach in different scenarios.

In their study, Al-Yaseen, Othman and Nazri (Al-Yaseen et al., 2017) combined the SVM and Extreme Learning Machine method in a multi-level hybrid model to classify data as normal or abnormal for real intrusion detection problems. They reported that this approach resulted in high efficiency and the best accuracy performance, with a score of 95.75%, compared to other studies. However, one disadvantage of this study is that it only used two algorithms (SVM and Extreme Learning Machine) and one dataset, which may limit the generalizability of the results and the ability to detect other types of intrusions or attacks. Additionally, the study only focused on accuracy as the measure of performance, without considering other important factors such as false positive rate and detection time.

Rakha and his team (2018) proposed a deep learning-based method for intrusion detection using the NSL-KDD dataset. They used an iterative neural network model and reported improved performance and accuracy (83.28%) when compared to other traditional methods such as SVM, ANN, and J48 (Rakha et al., 2018). Though, this approach is only tested on one specific dataset and may not perform as well on other types of data or in different environments. Also, deep learning models can be computationally expensive and require a large amount of data for training, which may not be feasible for some organizations.

Additionally, Kaveh and Dadras (2018) proposed a solution that addresses real-time issues and is efficient in solving the multi-class classification problem with a 3-stage structure. The study employs K-Nearest Neighbors algorithm for classification, the Naive Bayes algorithm for feature selection, and One-Class Support Vector Machine (OSVM) for outlier rejection. They have also observed that the K-Nearest Neighbors algorithm was efficient for solving multi-label classification problems, achieving an accuracy of 95.77%. (Kaveh & Dadras, 2018) To conclude, one limitation of this approach is that it only uses a single algorithm, the K-Nearest Neighbors algorithm, for classification, which may not be suitable for all types of data and may lead to lower accuracy for some datasets. Also, the approach relies heavily on feature selection, which may not always be effective in identifying relevant features for classification, leading to a lower accuracy rate.

Ghafir and his colleagues (Ghafir et al., 2018) examined the performance of various machine learning algorithms, including decision trees, different types of SVM models, nearest neighborhood, and ensemble methods on network traffic data. They discovered that the linear SVM algorithm had the best performance with an accuracy of 84.8%. Additionally, they built a system called "MLAPT" and used accuracy as the only metric to evaluate the effectiveness of the algorithms used. It should be noted that the use of one measure to evaluate the performance of the algorithms can be limited. The gab founded of this study is that it only uses one measure, accuracy, to evaluate the performance of the algorithms. This might not give a complete picture of the effectiveness of the algorithms as other measures such as precision and recall should also be considered. Additionally, the study only uses one dataset, which may limit the generalizability of the results to other datasets or network environments.

Bodström and Hämäläinen (2019) in their study proposed a theoretical model for detecting Advanced Persistent Threat (APT) attacks which emphasizes that APT attacks

are persistent and multi-step, using the entire network stream as input. They found that using a deep learning stack that employs sequential neural networks is a more efficient and flexible architecture for detecting APT attacks (Bodström & Hämäläinen, 2019). However, the study's limitation is that it only focuses on one type of deep learning architecture, and it does not consider other potential approaches or methods for APT detection. Additionally, the study does not provide information on the real-world applicability of the proposed model, which limits its generalizability.

A study by R. Zhao and his team (2019) suggested a self-learning approach for the NSL-KDD dataset, which integrates soft maximum regression and sparse autoencoder classifications. The results of their experiments indicate that the proposed model was effective in detecting network attacks across multiple classes (2-5-23 classes) (R. Zhao et al., 2019). However, the study is limited in its use of a single dataset for evaluation and testing which can affect the accuracy of the proposed solution to other datasets and real-world scenarios.

Additionally, Ghafir and his team (2019) developed a two-step system for identifying and predicting APT attacks. The first phase involves grouping alerts that are related to the same APT campaign over a specific time frame through correlation. The second phase utilizes a hidden Markov model to identify the most probable sequence of APT stages based on a sequence of correlated alerts. The study revealed that the system had an accuracy rate of 91.8% for predicting the sequence of APT stages, and a high accuracy rate of 66.5%, 92.7%, and 100% for predicting the next step of the APT campaign based on two, three, and four correlated alerts, respectively (Ghafir et al., 2019). However, the study would benefit from testing more classifier algorithms and using a real-time flow dataset as it is not fair to evaluate and predict based on only one classifier and one dataset.

Han and others, (Han et al., 2019) proposed a malware detection method named MalInsight relied on machine learning techniques. They studied three threats aspects including structural features low-level behavior (the registry), basic structure (the network), and high-level behavior (files operations). Actually, they validated experiments by conducting a actual malware dataset. An accuracy of 99.76% and 97.21% had got for new obfuscated and hidden malware respectively. MalInsight also got 94.2% accuracy for malware classification. Nevertheless, this study failed to correlate between different malware characteristics, the detection accuracy fluctuated

obviously, and malicious tendency prediction which was based on the systematic profiling only.

Aljawarneh and his team (2019) in their research compared the effectiveness of three different algorithms, J48 decision tree, Neural Network, and Support Vector Machine (SVM) for intrusion detection. The results showed that the J48 algorithm outperformed the other two algorithms in terms of accuracy, with a lower rate of false positives and a higher rate of true positives (Aljawarneh et al., 2019). However, one disadvantage of this study is that the authors only used a single dataset (NSL-KDD) for evaluating the performance of the algorithms, which may not be representative of other datasets or network environments. This limits the actual results, and the performance of the algorithms in other scenarios cannot be determined.

In the study conducted by Munivara Prasad and his colleagues (Munivara Prasad et al., 2018; Prasad et al., 2020), a real-time detection method for Distributed Denial of Service (DDOS) attacks at the application layer was proposed. The researchers developed a new toolkit using the absolute time span criterion for training and testing models. The study found that the Cuckoo's Binary Clustering strategy improved prediction accuracy and reduced the cost burden of the other two algorithms, Shark and Bat. However, the study was limited to only detecting DDOS attacks at the application layer and did not include other types of attacks. Consequently, the proposed toolkit was only tested on a limited dataset and further testing on a larger and diverse dataset is needed to evaluate its competent and effectiveness in real-world scenarios.

Zhang and others (H. Zhang et al., 2020) have used a method consisting of two parts (i) a classification algorithm for threats detection based on a support vector machine (SVM) and a deep belief network (DBN) and (ii) a real-time threat detection algorithm based on the sliding window for calculating the flow and the frequent patterns. Moreover, CICIDS2017, an open open-source dataset, was used for the experiment. The result indicated that DBN was .7% higher than the traditional classification algorithms, and the SVM accuracy was 2% higher than the integrated algorithms. In fact, actual live data need to be tested rather than a ready dataset that is used here to have accurate actual readings.

Concisely, Yan and his team proposed a technique for identifying APT attacks by utilizing deep learning to examine DNS request records and evaluate the likelihood of suspicious DNS activity. They collected and analyzed more than 4 billion DNS request records from a significant campus network and simulated attack data, and found

that the method had an average accuracy of 97.6% in identifying suspect DNS behavior, with a 2.3% false positive rate and 96.8% recall (Yan et al., 2020b). However, it is important to note that the method has not been tested in a multi-system environment and its ability to perform well in other types of systems is uncertain.

A study was conducted by Raman and his colleagues, where they proposed an intrusion detection method that utilizes a Hyper Graph based Genetic Algorithm (HG-GA) to optimize parameters and select features in a Support Vector Machine (SVM) using the NSL-KDD dataset. The results of the study showed that their HG-GA SVM approach had an accuracy of 96.72% and performed better than other methods like Bayes Net, GA-SVM, Grid-SVM, Random Forest, and PSO-SVM (Gauthama Raman et al., 2020). Actually, there is one potential disadvantage of this method is that it is only tested on one specific dataset (NSL-KDD) and may not generalize well to other datasets or network environments. Additionally, the use of a single algorithm (SVM) with a specific optimization method (HG-GA) limits the flexibility of the method and might not be suitable for other types of IDS tasks.

In this study, (Lavicza et al., 2021) propose a method for identifying and predicting Advanced Persistent Threat (APT) attacks by combining attack and defense patterns. They use the Observe-Orient-Decide-Act (OODA) loop and Black Swan Theory to detect these attacks. They argue that communication is the key aspect of an APT attack, and therefore suggest recording the network data stream to identify the attack. The authors also suggest that the data stream should be transferred to the detection process without any reduction of features. They used Apache Hadoop with a logical layer that includes steps such as Information Gathering, Weaponization, Delivery, Exploitation, Installation, Command, and Control (C2), and Actions. This approach is able to predict and detect APT attacks by following each step necessary to achieve the attack's goals. It should be noted that the study does not present any results which means it did not use a dataset and evaluation environment.

Dalmaz and his colleagues (2021) proposed a method that uses SVM, Naive Bayes, and J48 decision trees as the primary learning algorithms. They used the Information Gain method for feature selection. They found that using the J48 algorithm together with the Adaboost algorithm provided the highest accuracy of 97% compared to other methods (Dalmaz et al., 2021). However, it is important to note that the proposed method was tested on one type of dataset and one network environment.

Additionally, the results may not be robust and generalizable, which could lead to imprecise readings.

In (Extension & Brandao, 2021) study, it was found that detecting APT attacks requires a large amount of data and the results from the detection process can be difficult for security analysts to understand. To address this, Guo et al. proposed a two-layer approach for intrusion detection that includes two anomaly detection components and one misuse detection component using the KDD-99 dataset and the Kyoto University Benchmark Dataset (KUBD). The approach uses the Anomaly Detection method Based on the change of the Cluster Centers (ADBCC) and the K-means algorithm for the first anomaly detection component, and the K-nearest neighbor (k-NN) algorithm for the second anomaly detection and misuse detection components. The results show that the proposed approach has better performance with 93.29% accuracy compared to ADBCC method with 92.71% accuracy on the KDD-99 dataset and 95.76% accuracy compared to ADBCC method with 92.85% accuracy on the KUBD dataset. The proposed approach requires a large amount of data, making it difficult for security analysts to understand the results from the detection process. As well, the proposed two-layer approach, while showing improved performance compared to ADBCC method, has only been tested on specific datasets (KDD-99 and KUBD) and may not be the same to other types of datasets and network environments.

In the study conducted by Finder et al. (2022), a method was proposed for detecting Advanced Persistent Threats (APTs) using a combination of Machine Learning (ML) algorithms, dynamic analysis, and multivariate time-series data (MTSD) associated with behavioral information. The MTSD approach used a time-interval temporal pattern mining technique for exploitation and was able to handle noisy and missing data. The authors also proposed an active learning approach to improve the updatability of antimalware tools and reduce the need for expert labeling. The proposed framework was tested using a dataset of 9,328 files, with 5,000 being benign and 4,328 being malicious, and was found to effectively improve detection capabilities in ML classifiers with a 95.15% area under the curve (AUC) in SVM (Finder et al., 2022) . However, the study has a limitation in the number of algorithms used and the dataset coming from one source, which may affect the accuracy of the results.

In Finder et al.'s study (Finder et al., 2022), a method called "APT-Dt-KC" was proposed for detecting, analyzing, and preventing cyber threats by utilizing the cyber-kill chain model and comparing its fuzzy characteristics to those of APT attacks. To

reduce the amount of data processed, they applied the Pearson correlation test and utilized a hybrid IDS composed of the Bayesian classification algorithm and fuzzy analytical hierarchy process. The results showed a low false negative rate and false positive rate of 3.6% and 1.9%, respectively, outperforming traditional methods. The APT-Dt-KC had a high accuracy and detection rate of 98%, with an average improvement of 5% compared to existing IDS. However, this study has some limitations, as only one type of algorithm was used for each test, such as Pearson correlation for processing data and Bayesian classification for classification, and other ML algorithms such as SVM and decision trees may improve the accuracy and positive and negative rates.

The Industrial Internet of Things (I-IoT) is another study by (Finder et al., 2022) to interconnect various sensors and wireless devices to integrate cyber and physical systems in an extensive industrial network. A smart proposed method to secure I-IoT by detecting and classifying unexpected and unpredictable cyber-attacks like APT was studied to solve the IDS challenges. ML algorithms including SVM, Random Forest, Decision Tree, Bagging, Adaboot, Naïve Bayes, Logistic Regression, and Extreme Gradient Boosting were used alternatively to evaluate the KDDCup99 Dataset. Then, the results indicating Adaboost classifier accuracy outperform 99.9% compared with others during the experiment with an execution time of 0.012 s. To conclude, this study depends on the KDDCup99 dataset which can be a diver in structure from one organization to another, especially in industrial fields. Also, the experiment running time is short reflecting that the dataset is small or it is a trained one that may give inaccurate results.

In their study, Neuschmied and his team (Neuschmied et al., 2022) implemented intrusion detection systems (IDS) for identifying malicious behavior patterns in local network datasets using machine learning algorithms. They applied autoencoder-based methods for detecting such attack patterns, and used statistical analysis to determine features to improve the anomaly detection process. The study conducted experiments on a zero-day-attack test dataset and an APT-attack detection dataset, and found that the autoencoder-based convolutional neural network (AE-CNN) method is well-suited for the preliminary filtering stage, allowing for a reduction of 82% of analyzed network data while still detecting up to 10% of undetected cyberattacks. The combination of the AE-CNN method with a Variational Autoencoder (VAE) in the next stage reduces calculation time and increases precision value. However, the study only used these

methods without any improvements, and the datasets used were not real or actual, which may result in inaccurate outcomes.

Mijalkovic and his colleagues (Mijalkovic & Spognardi, 2022) proposed a study aimed at reducing the false positive rate by combining SVM, Naive Bayes, and Decision Trees algorithms, resulting in an efficient system with an accuracy rate of 99.62% and a false positive rate of 1.57%. However, the proposed method has been tested on a single type of dataset and network environment, which may differently result to other types of datasets and network environments. Additionally, the results may be influenced by the specific parameters and settings used in the study, which may not be optimal for all scenarios.

2.7. Summary of Gaps

Actually, the reviewed studies used various machine learning algorithms to detect Advanced Persistent Threats (APTs). These included decision trees (DT), support vector machines (SVM), k-nearest neighbors (k-NN), and ensemble learning. Additionally, other techniques such as genetic programming, classification and regression tree, dynamic Bayesian game model and SVM, k-NN and Correlation fractal dimension, k-NN, Logistic Regression (LR), Gaussian Naive Bayes (GNB), Decision Trees (DT), Random Forest (RF) and Linear Bayes (LB) were applied. Furthermore, techniques such as Principal Component Analysis (PCA), Support Vector Machine (SVM), Naive Bayes (NB), Decision Trees (DT), and Multi-layer Perceptron (MLP) were applied. Moreover, Fuzzy clustering was also used to detect APTs. It can be noted that these various techniques and algorithms were employed to achieve the best possible performance in detecting APTs.

In general, the limitations of using machine learning algorithms can include the following:

- Limited interpretability: some machine learning models, like ensemble methods, can be challenging to interpret and understand how they arrived at their predictions.
- Overfitting: if models are trained on too little data or with too many parameters, they may fit the training data too closely and perform poorly on new, unseen data.

- Lack of diversity in training data: if the data used to train a model is not representative of the real-world data it will encounter, the model may make inaccurate predictions.
- Insufficient feature engineering: the quality of the input features plays a significant role in the model's performance; if the features are not designed well, the model will not perform well.

However, in this study, some limitations have been identified when using machine learning algorithms to detect Advanced Persistent Threats (APTs). These include limitations in the:

- Data flow sampling and classification.
- Data at rest analysis.
- Data flow capturing.
- Autonomous behavior detection.
- Autonomous anomaly detection.
- Network behavior and anomaly detection.
- Deployment techniques.

Finally, our goal in this research is to fulfill these gaps by using a new method depending on a supervised ML with high accuracy and precision.

2.8. CHAPTER SUMMARY

A literature review chapter is a summary and evaluation of existing research on a specific topic. This section includes an overview of the key findings and trends in the field, as well as an examination of any gaps or inconsistencies in the literature. When it comes to detecting Advanced Persistent Threats (APTs) using Machine Learning (ML), a literature review chapter would typically cover the following topics:

- Introduction to APTs: This section provides a brief overview of APTs, including the types of attacks that fall under this category and the types of information that APT attackers typically target.
- ML for APT Detection: This section discusses the use of ML for detecting APTs, including the different types of ML algorithms and techniques that have been proposed for this purpose. It also examines the advantages and limitations of using ML for APT detection.
- Review of existing research: This section provides a detailed review of the existing literature on using ML for APT detection. This includes a summary of key

findings from different studies and a discussion of the strengths and weaknesses of each study.

- Gaps in the literature: This section highlights any gaps or inconsistencies in the existing literature on using ML for APT detection. It includes areas where research is needed, as well as areas where the results of different studies are conflicting.

Therefore, to summarize, detecting APT is complex and requires new techniques to get high-efficiency results. Thus, in this research, ML technology is proposed to be used as a comprehensive solution which is surely one or more than ML algorithms and programming languages will be there in the solution designing stages. Having reviewed the applications of ML towards detecting electronic threats in general and APT in particular, the next chapter applies the above-stated reviewed algorithms in a lab experiment. The next chapter describes the investigation, the methodology, the sample size, and the research procedures.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1. INTRODUCTION

This chapter is a crucial section of the research as it outlines the methods and procedures used to conduct the study. It provides a clear and detailed description of the research design, the sample selection, the data collection methods, and the data analysis techniques. It also includes a justification for the chosen methods and a discussion of any limitations or potential sources of bias. The methodology chapter has been written clearly and concisely, allowing readers to understand and replicate the study if necessary. In the introduction, the chapter provides an overview of the research design and the objectives, as well as a brief explanation of the chosen methods. It also explains how the chosen methods align with the research objectives and how they will be used to achieve them, which are about using ML to detect APT.

APT has demonstrated that technology alone cannot completely eliminate the issue of persistence. This research focuses on addressing the challenges posed by the obscurity of anomalies that create stress on work platforms. These anomalies propagate rapidly across platforms, making it essential to examine abnormal malicious activity. The goal is to classify patterns as 'infected' or 'unknown.' The workflow for generating rules for this pattern classification algorithm is illustrated in Figure 3.1.

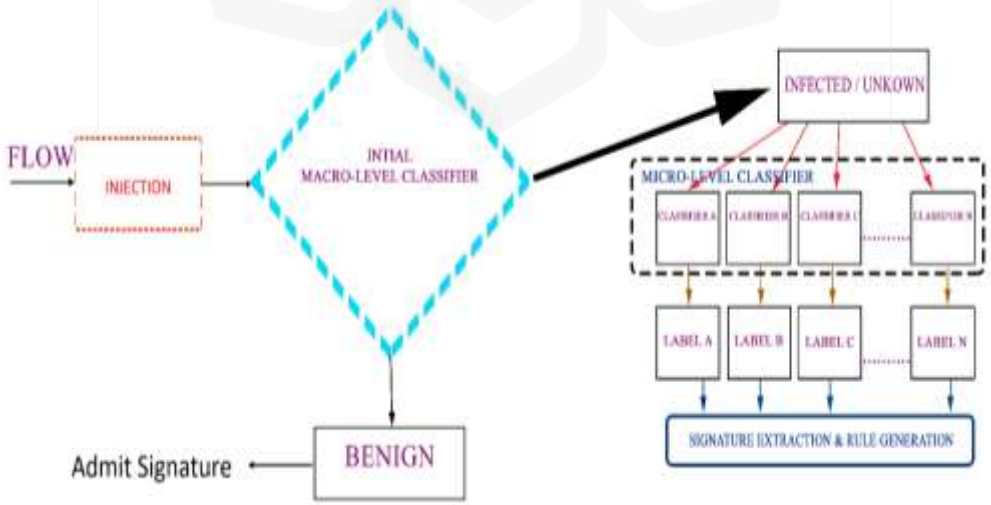


Figure 3. 1 Rules generation Flow Chart

Detecting the predefined famously exposed threats is still challenging due to the data type. Yet, statistical techniques and dedicated probabilities used to generate the filter rules belong to a deep research field that is growing (Shi et al., 2017). In this research, various metrics were utilized to raise the sensitivity and specificity of classifying anomalous packets and frames based on autonomous learning behavior. This study is carried out in two phases; the first phase aims to set up the APTs as anomaly detection techniques (Figure 3.2), and the second phase dedicates to demonstrating the methods to measure and evaluate the detection techniques.

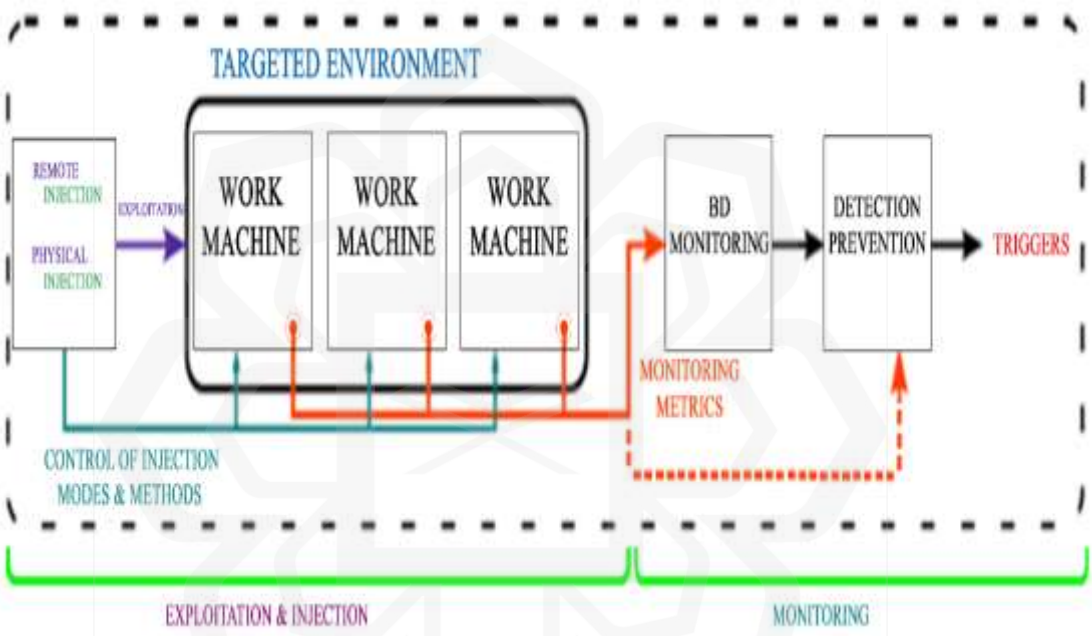


Figure 3. 2 Data flow analysis and APT Detection Diagram

Moreover, based on the environment and the digital surface, a synthetic study of anomaly detection techniques might be set in question towards the level of security. Nevertheless, it could be a problematic view in large computer systems and clouds in particular. Anomaly detection in cloud services is a recent field based on detection techniques in large computer systems. This work is complementary for gaining better and ameliorated techniques for detecting anomalies in cloud services that meet the framework's criteria and malware specificities described in the abstract and the introduction chapter of this research.

This chapter has structured into two main sections with two subsections, each as follow:

a) Strategies and methods for detecting anomalies:
    1) Identify methodological issues that could inhibit the detection of APT attacks.
    2) Enhance the existing standardized techniques aimed at detecting APT attacks.
b) Measures for evaluating these techniques in an experimental context.
    1) Develop an efficient autonomous APT detection model.
    2) Evaluate and compare the results of the autonomous APT detection model against the currently used algorithms.

3.2. SECTION I: ANOMALIES STRATEGIES AND DETECTING METHODS

The analysis of network services and files had been considered from a service provider's perspective where a medium enterprise working in the cyber security industry was recruited to participate in this study. The provider offers security services to newly established medium and small enterprises in the ICT sector. The current workspace is intensely active at a problematic level to measure the approximate size of its end users' surface, yet it serves thousands of daily remote end users. The main OS used by the provider is windows-based, and the provider's clients use a range of OS from windows, Linux, and Mac OS. For this reason, this experimental setup, Windows, Linux, and Mac OS, was selected to offer various services to customers. The goal was to enable the providers to detect anomalies in their working documents and network services. The process of anomaly detection was automated. In other words, it was performed online and programmed to adapt to change in workload independent of the provider's function or type of service implementation.

In this approach, the database was analyzed individually for each service and file extension. This was done because grouping together services that serve different functions would not provide meaningful information. The focus was on detecting anomalies in a potentially distributed system made up of multiple machines, computers, and network devices.

3.2.1. Identify APT Attacks Detection Issues

The fourth industrial revolution has set an enormous technological advancement in malware detection and spatial analysis environment. It can be seen in the long and diversified available technologies developed to hunt malware in different environments. APT detection solutions also had been presented through various technology providers. A deep benchmark had been done in this regard and numerous gaps were found and luminated when describing the application of machine learning technologies to detect advanced persistent threats. Thus, this research's innovative machine learning model solution has been implemented. It involves delimiting the list of gaps that the literature presented as a shortage and which had been covered by setting a better-automated solution framework for APT detection. Successful malware detection techniques have been implemented with the foundation of machine learning technologies. MATLAB from the math works company had shown excellent capabilities in deploying different machine learning standards to capture malicious behaviors within given input flows. These technological realizations have also been measurable within other machine-learning software solutions. WEKA from Waikato university also demonstrated a good potential in malware detection using machine learning standards. Rabbitminer from Rabbit Miner laboratories again demonstrated the machine learning capabilities in malware detection within modern environments, where IoT technologies had been emerging rapidly. This immense success had also drawn a significant explicit limitation when defining the Advanced Persistent Threats (APT) category of malware. Among the famous and exposed limitations, the honeypots could not detect APTs in post-infiltration (Bari, 2021; Xing et al., 2020)**.** The Intrusion kill chains did not set for preventative action in real time (R. Zhang et al., 2017). Statistical analytics-based approaches were ineffective due to the human analysts' need to collaborate to analyze critical threats, which may be weak and result in a greater rate of false positives (Cardenas et al., 2013).

3.2.2. Enhance Standardized APT Detecting Techniques

Using the available measures in machine learning leads to weak anomaly detection when anomaly’s properties describe the Advanced Persistent Threats. Proportionally an enhanced model had been set is showing an ameliorated list of results when compared to different available standards.

3.2.2.1. Statistical Factors

The study aimed to detect anomalies by analyzing patterns in data from various services and files using classification and clustering methods. The performance of these methods was evaluated by determining if the observations were correctly or incorrectly classified as normal or anomalous behavior. Two scenarios were considered for evaluating the performance of clustering: observations that were not assigned to clusters were considered anomalies, or a classification procedure was used after processing the clusters to identify anomalous clusters. In machine learning, a classification procedure is used to predict positive or negative labels, with normal behavior being considered negative and anomalous behavior being considered positive. The results were organized in a confusion matrix and evaluated using metrics such as accuracy, precision, recall, and F1-score. They illustrated in Table 3.1, composed of the following main metrics:

- True Negative (TN): normal behavior forecasted as such.
- True Positive (TP): anomaly forecasted as such.
- False Positive (FP): normal behavior forecasted as an anomaly.
- False Negative (FN): anomaly forecasted as normal behavior.
- Positives: anomalies total number (P = TP + FN)
- Negative: normal behaviors' total number (N = TN + FP)

Different metrics can be derived from the main metrics, such as precision and recall, which are also known as true positive rate (TPR) and false positive rate (FPR).

- The accuracy is the proportion of correctly predicted observations.
- Precision indicates the likelihood that an observation classified as positive is actually positive.
- Precision is calculated by the number of true positives out of the total number of observations classified as positive, whether correctly or incorrectly.
- Recall, on the other hand, represents the percentage of true positives. It is calculated by the number of true positives divided by the total number of positives that the method should have identified.

The precision metric measures the proportion of correctly identified positive observations out of all observations classified as positive, while the recall metric measures the proportion of correctly identified positive observations out of all actual positive observations. This means that the precision looks at the number of true positive

results in relation to all positive results, and the recall looks at the number of true positive results in relation to the total number of actual positive results. The following Table 3.2 define these measures:

Table 3. 1 Confusion Matrix Table

	Real Positives	**Real Negatives**
Positives Predicted	TP	FP
Negative Predicted	FN	TN

Table 3. 2 Definition of Performance Measures

Measure	**Formula**
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
Precision	$\frac{TP}{TP + FP}$
True Positive Rate (TPR)	$\frac{TP}{TP + FN}$
FPR	$\frac{FP}{FP + TN}$

These measurements were particularly used in various detection works (V. Agrawal et al., 2009; Brungard et al., n.d.; Z. Li et al., n.d.; Society & 2015, 2015; Yu Wang et al., 2013; Wu et al., 2020).

The Receiver Operating Characteristic Area Under Cure (ROC AUC) (Ling et al., 2006; Z. Wang et al., 2020) and precision-recall (PR) curves were often used to summarize these metrics for several prediction thresholds (Society & 2015, 2015; Yu Wang et al., 2013; Z. Wang et al., 2020; Wu et al., 2020). In this context, it is necessary to study ROC curves to understand detection performance, but it is not enough to make conclusions. The goal is to increase the number of true positives while reducing the number of false positives while also maintaining a high precision and recall by automating the generation of AI-based rules.

3.2.2.2. Data Types for an APT Detection

The study defined anomaly detection as the process of identifying and characterizing unknown patterns in a dataset made up of attributes (i.e., the columns) and observations (i.e., the rows) related to a target system. The observations that deviate from the overall trend, represented by the majority of observations, are considered anomalies (Guerra-Manzanares et al., 2020; D. Zhao et al., 2017). Different methods are used to accurately identify these global trends in the dataset, including the data and algorithms used for detection. The types of data used in this analysis are discussed further in the research.

3.2.2.3. Different Types of Data

In this research, anomaly detection is defined as identifying unusual patterns in a dataset made up of observations and attributes that are not known in advance. The observations that deviate from the overall trend are considered anomalies. The methods for detecting anomalies vary, depending on the type of data and the detection algorithms used. The data that can be used for detecting anomalies in a computer system includes historical logs of events, audit trails, and application usage statistics or system performance observations. These types of data provide different perspectives on the system and can be used to identify anomalies in different ways (Haixiang et al., 2017; Mahbub et al., 2019; Salcedo-Sanz et al., 2020; K. Singh et al., 2014).

Hence, the data sets used to feed the learning process were created from:

- Traceability audits
- Journal logs
- Systems monitoring statistics

3.2.2.4. Learning-Based Techniques as Detection Methodologies

The anomaly detection has to be, therefore, to distinguish between the observation corresponding to the anomaly which is made while the system undergoing an anomaly or the normal behavior observation. Machine learning algorithms are used to make this distinction using previous observations, which is known as a prediction (S. Agrawal et al., 2015; Palmieri et al., 2014).

The machine-learning algorithm uses previously processed observations to predict the category of a new, unprocessed observation.

The process of determining whether an observation is normal or anomalous is known as labeling. The label assigned to an observation, which can either be normal or anomalous, is used by a machine-learning algorithm to predict the category of new observations based on previously processed observations. The accuracy of the algorithm's prediction can be verified by an operator.

At the stage where a model is constructed using labeled observations to distinguish between different types of observations, the process of classifying data using supervision is referred to as supervised classification (Gliozzi et al., 2009; Long et al., 2014).

3.2.2.5. Experimental Detection Performance Evaluation

The evaluation of detection techniques involves the use of two main types of data:

- Data collected from normal system operations, including errors reported by users.
- Data collected from controlled experiments in which errors are intentionally introduced to the target system being observed.

In this study, the focus was on evaluating the detection techniques by using two types of data: data from operational usages, such as errors reported by users, and data from controlled experiments in which errors are deliberately introduced into the observed target system. This method, known as fault injection, allows for injecting faults into a system while observing it during the injection, using an injection protocol. The detection of anomalies was evaluated by analyzing how well the anomalies were identified, either as actual anomalies or normal behavior. Both classification and clustering techniques were used in the process. The following sections describe the hardware, software, operating systems, and data analysis approaches used in this experiment.

3.2.2.6. Experimental Setup

- **Hardware Devices**

The proposed platform consisted of a cluster made up of FOUR machines.

- Two computers were necessary to deploy the different modules within the cluster.
- A monitoring machine was used to control the behavior remotely.
- A network commutator machine.

- **Software and Applications**

The software described in this section was aimed at fulfilling the following tasks:

- Auto Generate rules: to describe the content - set the patterns from the network packet's payload.
- Define the behavior of the detector, whether in an Operating system-based environment or in, a virtual environment based or a mixture of both.
- Cloning wireless environments.
- Cloning computer network delivered services.

- **Operating Systems**

- Wifislax: an OS used to clone wireless computer networks.
- Kali: is an OS used for digital forensics activities and penetration tests.
- Windows: a standard desktop OS used to communicate with the hardware and allows other programs to run.

- **Analysis Environments**

- MATLAB: MATLAB was used to answer the clustering question because MATLAB offers statistics and a machine learning toolbox that is easier and faster than other software. Most importantly, MATLAB offers clustering methods like DBSCAN. This method was used because it is a widely used algorithm in anomaly detection research (e.g., Al-Amari, 2020; Al-Shaqsi, 2019; Donald J. Trump, 2021).
- Python: Python was used to answer online clustering questions because Python offers online live statistics and Machine Learning Toolbox, which were more straightforward and faster in an online environment.
- MS Azure: a dedicated AI-based platform for dedicated machine learning solutions.
- Google Colab: an open platform for ML solution implementations and tests.

- **Clustering and Classification Algorithms**
  - SVM: classification
  - Decision Tree: classification
  - Random Forest: classification
  - Sliding window classifier
  - Neural network classifier
  - DBSCAN: clustering

- **Intrusion Systems**

Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) were the primary network filters considered to deploy the generated rules:

  - SNORT: is an open-source software used for security purposes to analyze network traffics and logs packets in real-time.
  - Acunetix: is an automated security web application tool used in testing, auditing, and discovering vulnerabilities.
  - Netfilter: a framework that came with Linux kernel which help kernel modules to register callback functions at the Linux network stack.

- **Network Environment**

In the first preview, the environment was made up of two platforms:

1. The implementation and testing environment
   - The ML and research needed software and tools to generate datasets (rules to feed the IDS/IPS).
2. The Injection environment
   - A physical environment where an intruder might succeed and break the rules upon which the prescribed IDS/IPS work.

- **Extraction Tools**
  - Python libraries: Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12(85), 2825–2830.
  - Network debuggers: To explore the background and low-level languages, searching for anomalous behaviors.
  - Iproute2: an open-source collection of utilities to control the network traffic.

- **Topologies Structure**

The suggested topologies were based on the mesh topology, physical and logical networks, to test the consensus and the asymptotic-based methods in generating the datasets and in anomaly detection, which were two primary keys in defining the best topology to follow.

- **Dataset Sources**

The monitoring data were obtained using the monitoring system for distributed systems Ganglia (Cinque et al., 2023).

A local network flow based on ad-hoc topology data and information was generally obtained specifically over monitored applications by instrumentation, testing the applications' availability or latency, etc. Numerous tools might be deployed to enhance the quality of the cultivated data at this level; the python library Pysphere has been mentioned here.

Besides, the literature dataset provided the research with rich and well-correlated datasets samples. Amongst them are the followings:

- The Harmonized Indices of Consumer Prices (HIPC)
- The Industrial Production Indices (IPI).

- **Malware Injection**

Injected the malware during the system action peak time had explored by the usage of the following two tools:

I. IPTABLES; the monitoring API dedicated to the FW Netfilter.
II. IPROUTE2; an open-source network controller.

3.2.2.7. DATASET PREPROCESSING

This study aims to create an APT detection system using machine learning techniques that can differentiate between normal and abnormal network traffic patterns. One benefit of this approach is that, even if the APT attack has a unique signature that is not yet known, the abnormal behavior that follows the attack will deviate from standard traffic patterns, allowing the system to detect and react to the anomaly. We implement commonly used supervised ML techniques through MS Azure Cloud, such as SVM and K-Means, on the dataset for real-time and accurate anomaly detection. The process is illustrated in the following diagram (Figure 3.3).

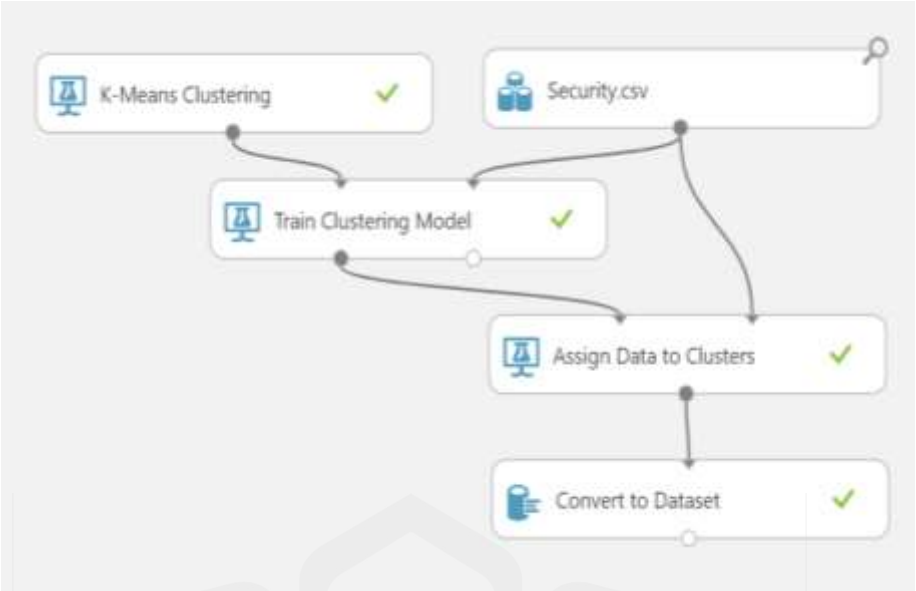


Figure 3. 3 Supervised ML techniques through MS Azure Cloud

- **Data cleaning:** This step involves removing missing, duplicates, or irrelevant data from the dataset. The WEKA tool was used to remove non-valuable values from the dataset by applying the One-Anova test.
- **Data transformation:** This step includes normalizing or scaling the data to ensure that all features have the same weight in the analysis. MS Azure Cloud was used to convert the raw data into a .CSV format for further processing. This allowed us to easily import and manipulate the data for use in an ML-based APT detection system.
- **Data reduction:** This step involves reducing the dimensionality of the dataset by removing any irrelevant or redundant features. Here, the data was further filtered by date in order to decrease the number of logs generated by the system.
- **Data integration:** This step involves combining multiple datasets or sources of data to create a more comprehensive dataset for analysis. This step provides a consistent and accurate view of data across an organization, which can help support better decision-making, improve operational efficiency, and support data-driven business strategies. An example of this stage is exemplified in Figure 3.4.

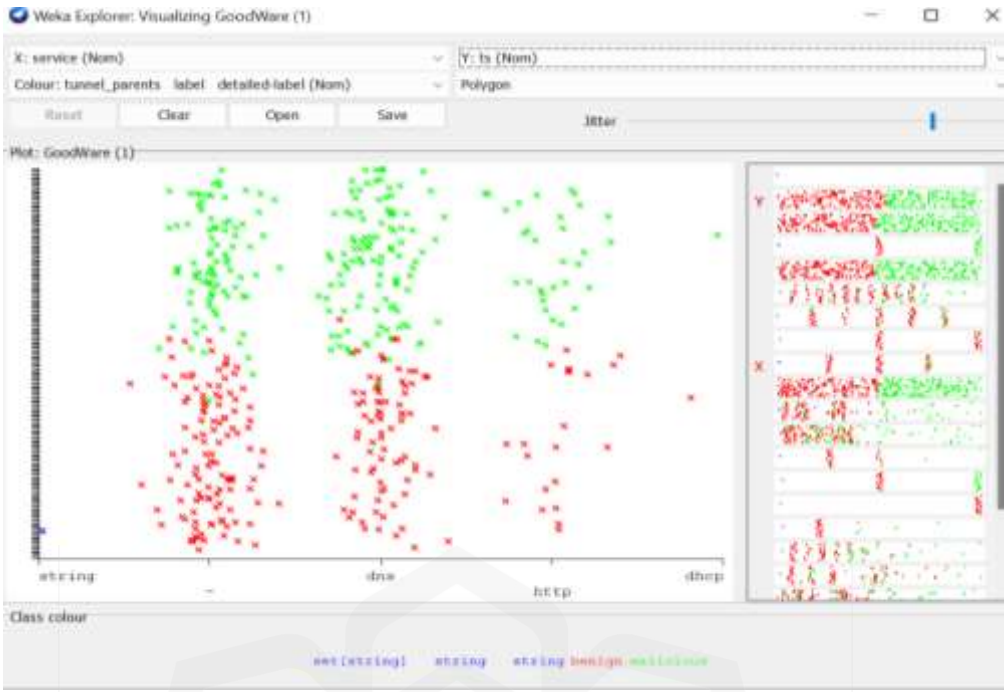


Figure 3. 4 Data Integration Examlpe

- **Data discretization:** This step involves converting continuous data into categorical data for more accessible analysis. This is done by dividing the data into intervals or bins and assigning each value to the corresponding bin. Data discretization aims to reduce the number of values in the data set and make it easier to work with. This research applies these methods using various algorithms, such as k-means clustering or decision trees.
- **Data splitting:** This step involves dividing the dataset into training, validation, and testing sets for model evaluation and selection.
- **Feature selection:** This step involves identifying and selecting the essential features from the dataset that will be used in the ML model.
- **Data augmentation:** This step involves generating new samples from the existing dataset to increase the dataset's size and improve the model's performance.

The followed Figure 3.5 illustrates the steps taken to process the dataset using the ML-SVM algorithm.

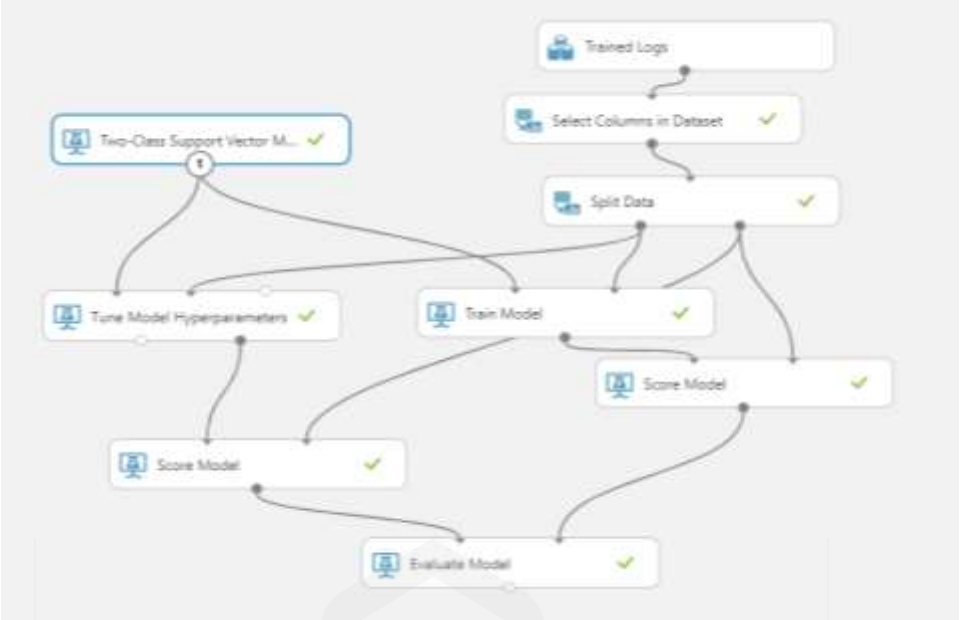


Figure 3. 5 Dataset Process using the ML-SVM algorithm

- **DATASET USED**

The data integration method refers to the process of combining data from multiple sources into a single, unified view. This can include techniques such as data warehousing, data federation, and data virtualization. On the other hand, data discretization methods are techniques used to convert continuous data into discrete or categorical data. This can include techniques such as binning, clustering, and decision trees. This research uses a dataset consisting of 57 features of network packet files, which were recorded at various time intervals. The dataset was preprocessed by removing non-valuable values using the One-Anova test in WEKA and converting the raw data to .CSV format using MS Azure Cloud. The dataset was also customized by date to reduce the number of logs generated by the system. The dataset features include the sequence number, time, source and destination IP addresses, protocol, length of the packet, and additional packet details. The dataset is divided into five main categories: live system logs, Journal logs, and different Network Appliances logs.

In order to analyze the APT attacks, we first need to preprocess the raw data files. The process starts by loading each pcap data file, then applying the corresponding filtering rule. After that, the data is divided into attack and normal data and both are exported to separate CSV files. Then, we label the attacks, combine the two CSV files, and remove any redundant data. This process is repeated for all pcap files. After concatenating the CSV data files, the size of the resulting file will likely be larger,

depending on the size of the original files and the amount of data that was retained after labeling and dropping redundant data. Thus, the resulting file size ranges between 20 MB and 463 MB as shows in Figure 3.6. More details of this process as followed:

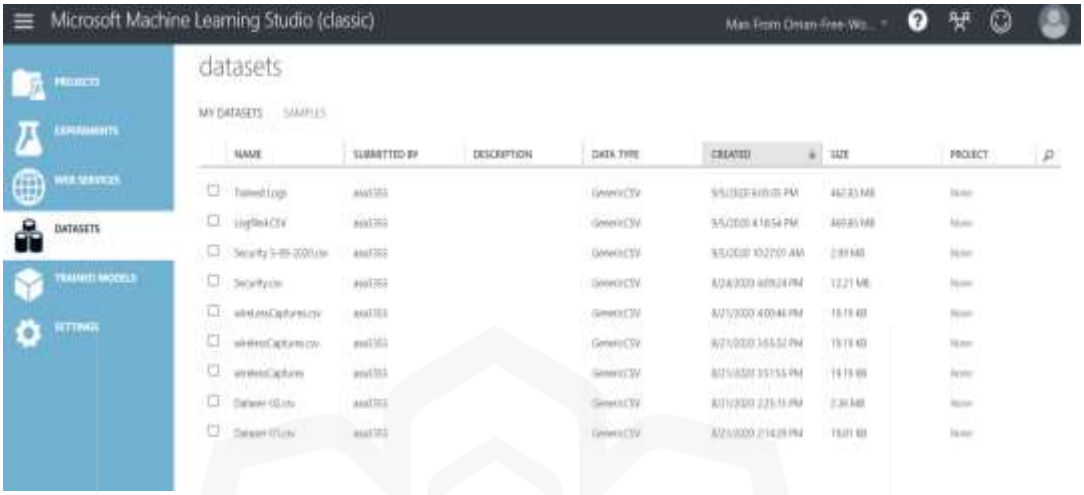


Figure 3. 6 Example of used Datasets in MS Azure

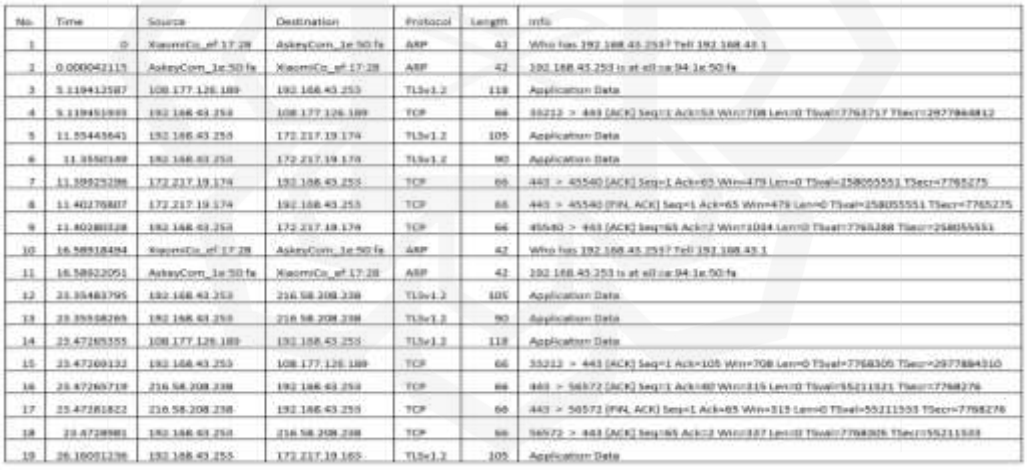


Figure 3. 7 Example of Wireless Captures Data

3.2.2.8. Performance Measures

To measure the effectiveness of the work later in the test phase, a list of measures was examined to analyze and discussed for better refinement. Those measures would be classified on the resources, access methods, and process management.

And for the virtual environment-based approaches, measures for the hypervisor memory space management were employed.

The filtering performance of an IDS was set to cover the entirety of the requests made on IDS in an assessment. The evaluation process focused on several criteria to measure the standing performance of an IDS, such as effectiveness, efficiency, ease of use, security, interoperability, and collaboration. The below sections explain these criteria in detail.

The Effectiveness: This means that an assessment ought to survey IDS's capacity to distinguish between assaults and the level of bogus cautions. Also, an Intrusion Detection System had the potential to raise an alert at whatever point there was an interruption, while the bogus caution rate ought to be kept at a low level that did not work over clients' resilience. In a perfect world, the assault discovery rate should be 1, and the bogus alert rate should be 0.

The Efficiency: An IDS should use less time and recollections to identify interruptions and report cautioning messages. An interruption recognition system was simply used to give security administrations like placing interruptions for PC frameworks.

The Ease of use: This means that an interruption recognition framework should not be too difficult even to consider permitting a client who was not a security master to work with it.

The Security: New malware's assault instruments were significantly complex. They, at this point, do not remain at the phase of utilizing IDS avoidance methods. Instead, some of them attempt to assault IDS and make the system break down.

Interoperability: This means that an Intrusion Detection System could interoperate with another in some augmentation. It was unimaginable that an interruption discovery system could recognize a wide range of assaults. More than one IDS cooperation may fundamentally improve the interruption identification rate; in any case, this may build the bogus alert rate and spend more assets.

The Collaboration: This means that an IDS might be brought with other security systems together to improve PC security systems. So, we need to assess how the interruption discovery framework aligns with the additional security instruments that had to be there, like Firewall. Also, a need to guarantee the blend gives superior security.

3.3. SECTION II: EXPERIMENTAL TECHNIQUES MEASUREMENTS

For this experimental approach, firstly, delimited the respective factors used in different statistical activities for a behavior description, saying detection. Secondly, a deep descriptive scenario demonstrates the data types' impact for a better behavior description and, thus, APT detection. Those data types were the primary datasets to feed the learning process at different phases. Thirdly, a classification of the learning methodologies was described and implemented to deploy the factors described in the first section. Fourthly, metrics of evaluation of the effectiveness towards the detection efficiency based on a realistic injection scenario were described.

3.3.1. Develop an Efficient Autonomous APT Detection Model

3.3.1.1. Methodology of a Live Case Attack

Most malicious attacks happen unexpectedly without an organizational warning system gets interrupted. To simulate this factual scenario, researchers proposed live case attack simulation. Successful technology, based on expert systems and machine learning solutions, was trusted enough to go beyond the malware hunt. This definition was no longer the case when APT had been defined, and the IoT had emerged in the current life. Automation of the processes of malware detection had to be highly sensitive to changes and live flows. Available frameworks stand on the graphical rule-based machine learning algorithms, or even the analytical algorithms had to be enhanced accordingly to match the up-to-date changes. Those changes are demonstrated through the process's interoperability challenges. Univariate and linear regressions were the first technologies to analyze and classify APT behaviors in the "Innovative Data Analysis: Artificial Intelligence for Advanced Persistent Threats Detection" line. So, a need to delimit the lexical terminology that APTs use in their different actual definitions. Those terminologies had been studied and set upon the feature selection and extraction enhanced solution this research had gone through. From within this level and with the help of real-life APT history, expertise had been utilized behind a reference to determine anomalous entries in a given entry flow. The flow was explored during the implementation phase through different datasets; datasets from the literature review and datasets captured from the local framework. A manual process was needed to name the captured anomalous points and generate the decision tree-like model to be passed to the Naive Bayes algorithm to generate the decision rule responsible for the APT detection. The generated rule would be fed to the admitted open-source Intrusion Detection

System (IDS), the SNORT. Then this fed SNORT was seen acting effectively in real cases of APT detection; these steps are explained in Figure 3.4.

A live case attack was made where flow had been captured in live time and sent to the monitoring environment. This process covered the following steps and is presented graphically in Figure3.3:

- Flow capturing (preliminary Dataset).
- Tiny manual classification (signature-based & Behavior scenarios to classify).
- Features extraction & selection (data cleaning & suspicious features extraction).
- Training and Model generation (supervised learning and Deep Learning Algorithms for a higher detection accuracy).
- Rule Generation and IDS feed.

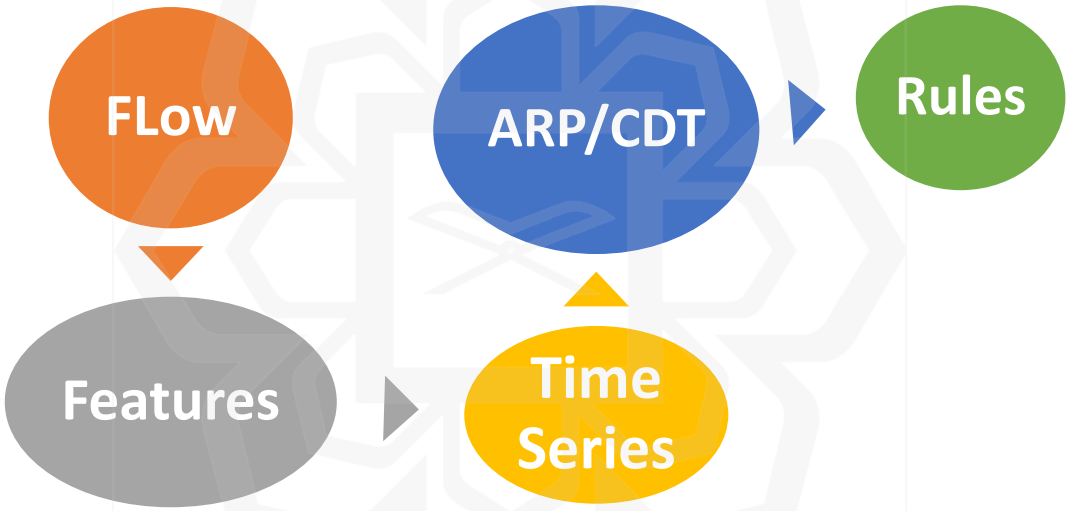


Figure 3. 8 Innovative Methodology of a Live Case Attack

3.3.1.2. Methodology for one APT Case Detection

Case Study: Deployment of the fed IDS

Several studies were analyzed, and practical tests were realized where the conclusion of a familiar feature was summarized to automate the generation of digital anomalies rules and then fed with the pre-set network filters. Accordingly, the use of SNORT filter, thus, rules syntax was matched to succeed the APT detection. New datasets were utilized to test the defined framework result of the methodology. Dedicated datasets to real APT cases were utilized to test the effectiveness of this research finding; in the final analysis, a comparison of the results and their efficacy was needed to prove the research's success. This methodology is presented below in Figure 3.4.

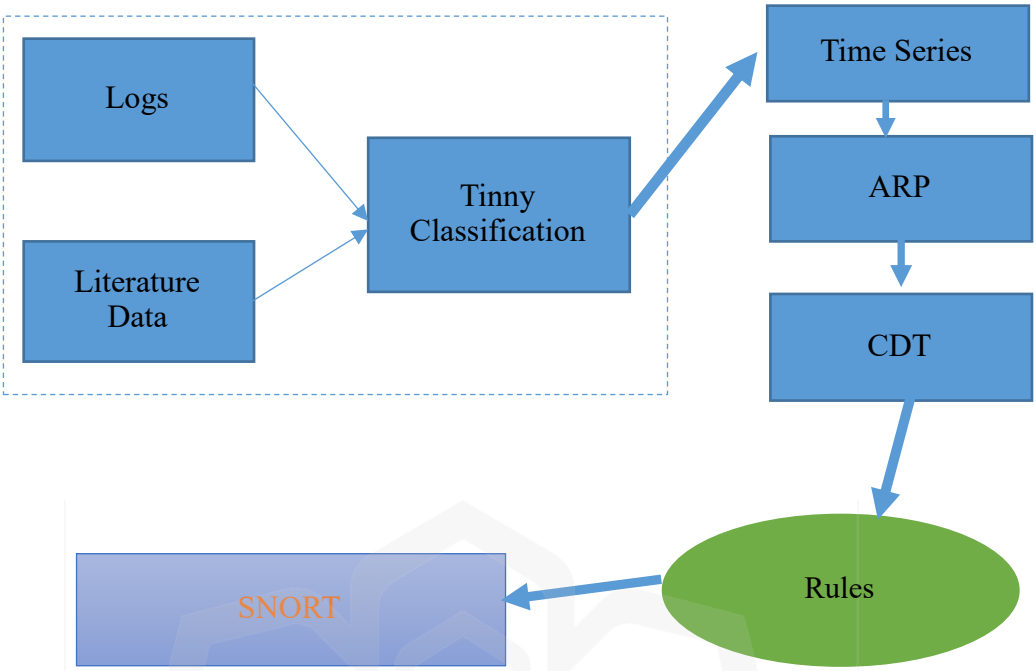


Figure 3. 9 APT Detection Methodology Phases

3.3.2. APT Autonomous Detection Model Results Against Existing Algorithms

It is not easy to evaluate the research methodology of APT detection without knowing specific details about the research in question. In this methodology, an effective research approach in APT detection would involve a systematic and thorough approach to identifying, analyzing, and detecting advanced persistent threats. This could include using various tools and techniques, such as network traffic analysis, malware analysis, and threat intelligence gathering, to gather evidence and develop a comprehensive understanding of the threat. Additionally, the research methodology should include appropriate controls and safeguards to ensure the accuracy and reliability of the research findings. Overall, the effectiveness of this research methodology depends on its ability to accurately and effectively identify and analyze advanced persistent threats. Therefore, this research went through three (3) stages, the time series generation and the anomalous points detection and labeling. Then the composition of decision trees to accomplish the pre-set methodology. After that, detect live APTs appearances, which at the end demonstrates an evaluation of the proposed methodology upon comparison with different rule-based standardized algorithms.

3.4. CHAPTER SUMMARY:

Researchers have explored APT attack detection methods, highlighting limitations such as real-time detection challenges and high false-positive rates. APT attacks use advanced tactics and evade traditional IDS systems. Machine learning (ML) techniques have gained attention due to their effectiveness. ML models often use combined features, like network data and user behavior, for training. Supervised (e.g., decision trees) and unsupervised (e.g., clustering) learning methods have been applied.

Unsupervised ML detected known and unknown APT attacks, while supervised methods achieved high accuracy but struggled with false positives and negatives. Feature selection techniques like Information Gain and SVM-based models improved detection. Hybrid models, like SVM with GA, improved accuracy but may be complex. Some studies focused on specific datasets and algorithms, limiting generalizability. Deep learning approaches showed promise but required more data.

However, several studies used limited datasets, and algorithm choices affected results. Proposed approaches may lack real-world testing and could struggle with different data types. Some studies reported high accuracy rates, while others faced challenges like false positives and a lack of diversity in datasets and algorithms.

CHAPTER FOUR

DESIGN AND IMPLEMENTATION

4.1. INTRODUCTION

In this research, a Supervised Machine Learning Algorithm model has been used for detecting threats. This chapter describes the experimental setup as outlined in chapter one. The implementation phase was performed in virtual computer labs to test the Datasets and ensure targeted accuracy. The decision Tree algorithm was developed using MATLAB application which analysis and processes the collected logs through a programed script. Master data collection methods were done using variant software to achieve as much high accuracy and precision as possible. The current chapter is outlined into four different sections, and each one represents a research objective as indicated below:

a) To identify methodological issues that could inhibit the detection of APT attacks.
b) To enhance the existing standardized techniques aimed at detecting APT attacks.
c) To develop an efficient autonomous APT detection model.
d) To evaluate and compare the results of the autonomous APT detection model against the currently used algorithms.

4.2. IDENTIFY APT ATTACKS METHODOLOGICAL DETECTION ISSUES

4.2.1. Temporal Series Exploration

At first, a problematic view of anomaly detection in the time series was reviewed and supported by the most relevant work of the literature. Thus, the fundamental concepts of this work have been presented. Then, the field description of the application and the anomaly detection techniques were done. Furthermore, different detailed types of anomalies were treated in the literary works and compared with the abnormalities observed in real deployments. Then, the different anomaly detection strategies and evaluation methods were presented. Finally, a comparison of the founded contribution against state-of-the-art algorithms based on a set of criteria has gone through this chapter.

In many scientific fields, measurements were made over time. These observations lead to a collection of data organized in the form of time series. Time series data mining aims to extract all the meaningful knowledge in that data. Even though

humans had a natural ability to perform these tasks, it remains a complex problem that computers could perform quickly on vast amounts of data. Machine learning (ML), which experienced significant growth during this decade, was a path of improvement that was adopted in this work. This chapter presents the scientific context of this research. First, a description of the data processed was set in a form of a univariate time series. Then, the chapter explores patterns in this data and its use with machine learning.

4.2.1.1. Time Series

A time series is a set of real-valued variables collected (a series of observations) sequentially at a regular or irregular interval. These observations represent measurements associated with a timestamp indicating the time of its collection (Adhikari et al., 2012; analytics & 2015, 2015; X. Wang & Wang, 2020a, 2020b). Examples of time series include weather data such as temperature or precipitation, economic data such as stock prices, and medical data. In addition, a time series allows analysis of the effect of cyclical, seasonal, and irregular events on the measured data item (K. Choi et al., 2021; Lopez et al., 2021). Depending on the number of variables and their dependence, a time series can be univariate or multivariate.

- A univariate time series is a sequence of measurements of a single variable that depends on the time.
- A multivariate time series is a sequence of measurements of several variables. The variables are co-dependent and dependent on time (K. Choi et al., 2021; Lopez et al., 2021). In this case, the observations are Vectorial.

Figure 4. 1 Example of Univariate Time Series Corresponding to The Energy Consumption of a Building Calculated Through the Index Readings of a Meter.

Figure 4.1 is an example of a univariate time series. The Figure shows the evolution of a building's energy consumption over time. These numerical values were calculated from the readings of a counter or index managed in a research environment.

This work dealt with the type of univariate time series from different monitors and counters. It carried out experiments in a local lab, particularly on index or consumption data. These measurements are regular over time. However, they may contain missing data due to a failure in the sensor networks, which subsequently generates various anomalies. For experts, it is essential to detect inconsistent values and anomalies instead of deleting them for their analyses. The dataset that comes without missing values allows processing the time series without imputing the missing data.

4.2.1.2. Pattern Extraction

Pattern extraction or mining refers to a data mining method that involves finding existing patterns in data. The exploration of frequent or infrequent models, the sequential exploitation of models, and the extraction of a set of objects (itemset) also come under the exploration of models. Frequent patterns were subsets that occur a significant number of times (Fayyad et al., 1996; Iváncsy et al., 2006; Kinnebrew et al., 2013). Its support must be greater than or equal to a minimum threshold defined by the user to search for such patterns. This principle is widely used in association rules, such as the well-known apriori algorithm created by Agrawal and Srikant (S. Agrawal et al., 2015), to search for the most frequent patterns.

The detection of outliers, infrequent patterns, may be more important in many sequences than that in regular, more frequent analysis patterns. Changes in client behavior, unusual data flow rhythm, surprising protein sequence patterns, etc., represent aberrant patterns that may indicate abnormalities.

Time series can contain specific patterns that would be relevant for data analysis. Extracting shapes in logical time series is not new and has been approached differently. The time series is divided into segments (J Lin et al., 1997; Jessica Lin et al., 2004), and the segments are classified into model classes (Horst, 2017; Schenker et al., 2003). Classification can be based on any machine learning (ML) algorithm such as K-means, Support Vector Machine (SVM), decision trees or random forests, neural networks, etc. (A Bhatnagar et al., 2019; Hasan et al., 2019; Koushik et al., 2020). Due to their

periodicity, seasonal time series are good candidates to be analyzed via pattern extraction tools.

This field of research seeks to detect rarity. Therefore, the focus on detecting irregular patterns constitutes essential information for detecting anomalies in time series.

4.2.1.3. Sliding Windows

The sliding window algorithm is a well-known time series data segmentation method (Gürsakal et al., 2020). The sliding window temporarily approximates the actual value of time series data (Hota et al., 2017).

Figure 4.2 illustrates an example of the sliding window process with a window size of 4. Each x represents the daily observation of time series data (1,2,3...N).

The window covered from 1 to 4 (black rectangle) represents the historical data for four days, so it was used to predict the next day's value. The window (red rectangular) slides from left to right one day, a step of 1, to cover another four days (from 2 to 5) to predict the next day. The process was repeated until all data in the time series was thus segmented. This principle had been used as a pre-processing of time series to extract rules from segments (windows).

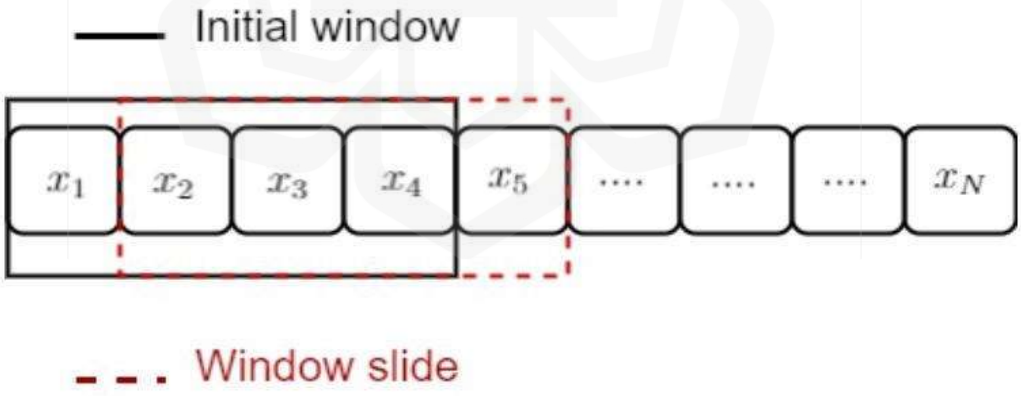


Figure 4. 2 Example of The Sliding Window Principle Process.

4.2.1.4. Machine Learning

Machine learning (ML) is a field of data analysis in which the algorithm learns from the data. ML aims to mimic human learning, reasoning, and decision-making (Koushik et al., 2020; Salcedo-Sanz et al., 2020). Unlike expert systems which are automated

systems based on human expertise, ML-based systems rely entirely on the approach to learning from data. The "intelligence" of ML algorithms comes from the iterative processing of data until the convergence of an objective function. ML makes it possible to build autonomous systems constituting artificial intelligence.

There are mainly two phases in an algorithm based on machine learning: a training phase, where the system learns from the data, and a testing phase, where the algorithm applies the acquired knowledge to new samples. The system can also continue to learn new data to increase its performance. In the following, an interest in different approaches based on machine learning for anomaly detection has been practiced.

These basic concepts constitute the perimeter field of research relating to the automatic detection of anomalies in time series. Moreover, they provide state-of-the-art anomaly detection techniques based on machine learning, as explained in the following section.

4.2.2. Anomalies Detection by Time Series

Time series are used to detect anomalies; types of anomalies dealt with in the literature are being used, and a focus is on the anomalies encountered in actual sensor deployments. Then, it presents a synthetic study of the application areas and anomaly detection techniques. Furthermore, several detailed works illustrated these techniques with the main associated concepts. At last, this chapter concluded with the methods and measures for evaluating these techniques in an experimental context.

Recent technological advances allow researchers to collect a large amount of data over time in various application domains. Observations recorded in a time-ordered fashion constitute a time series. Time series data mining aims to extract all the significant knowledge of these data, normal or abnormal behaviors over time, instituting indications of the operating conditions and abnormalities of the monitored system (Bhaskaran et al., 2020; Esling & Agon, 2012). For example, an aircraft engine rotation fault, a fault on a production line, or an increase in temperature may indicate a malfunction.

Anomaly detection appears to be the means to identify anomalous events and find patterns in the data that do not match the system's normal behavior (Chahla et al., 2019; Varun Chandola, 2009). One of the first studies on this subject was conducted by

(Fox, 1972); two types of outliers in the univariate time series have been defined as: type I, which affects a single observation, and type II, which involves both a particular observation and subsequent observations. This work was first extended to four outlier types; additive outlier (AO), innovational outlier (IO), level shift (LS), and temporary change (TC) (R. Tsay et al., 1998; R. S. Tsay, 1988); then in the case of multivariate time series (Galeano et al., 2006). Since then, many definitions of the term aberrant have appeared and many detection methods had been proposed in the literature. However, to date, there is still no consensus on the terms used (Carreño et al., 2019; Carrera et al., 2019). According to the applications and the fields of research, aberrant observations were referred to in different ways, such as anomalies, discordant observations, discords, exceptions, aberrations, surprises, peculiarities, or contaminants (Blázquez-García et al., 2021). The most common terms used in the literature include anomalies and outliers. According to (V Chandola et al., 2007, 2009), anomalies refer to "patterns in the data that do not conform to a well-defined notion.".

4.2.2.1. Definition of Normal Behavior

Other researchers define anomalies as "an observation that seems inconsistent with the rest of the data set" (Bansal et al., 2016; Hodge et al., 2004; Savage et al., 2016),. (Keogh et al., 2007; Laxhammar et al., 2013) Proposed another definition of anomalies, time series discords, which were different subsequences of all real subsequences.

The term outlier had been associated with noise, linking these observations to incorrect or inconsistent behavior (S. Agrawal et al., 2015), such as human errors appearing during data recovery (Barai et al., 2018; J. Zhang et al., 2011). In other situations, detecting instances with a significant deviation was also considered outliers. According to (Acuna et al., 2004; Hawkins, 1980), an outlier is an observation that deviates strongly from other observations. The explanation of this phenomenon can be reduced to the fact that a different mechanism generated it.

The terms anomaly and outlier convey the idea of an undesirable pattern, and in this work, it used the two terms interchangeably. There were two groups of methods for anomaly detection (Thill et al., 2019):

- Direct approaches include grouping, classification, or methods based on distance or density. Thus, anomalies are considered to be away from the centers of the clusters, to form a tiny class/cluster at fully-fledged, distant from their nearest

neighbors, or have a high probability distance.

- Indirect (or residual) approaches where normal behavior is learned and modeled. These models make predictions, and the gap between the observed value and the predicted value is used to decide whether an observation is abnormal.

Depending on the method used, the output of an anomaly detection algorithm links can be:

- Scores: Most outlier detection algorithms produce a score quantifying each data point's outlier level. This score can also be used for data points ranking in order of aberrant trends. However, this is a general form of output that does not provide a brief summary of the rareness of observations that should be considered outliers.
- Binary labels: are a classification that indicates whether a specific data point is an outlier or not. Some algorithms can directly produce binary labels, while others produce outlier scores which can be converted to binary labels by setting a threshold based on the statistical distribution of the scores. This type of output is commonly used for making decisions in practical applications.

In this research, a direct approach was used to detect anomalies allowing the generation of labels describing the abnormalities. To conclude, detecting anomalies has given rise to several research works depending on the nature of the data, the availability of labels on normality, and diverse application domains. We are approaching these concepts to explain in the following parts.

4.2.2.2. Areas of Application

Much research has been done on anomaly detection in recent years in various fields of application, in particular, intrusion detection, fraud detection, anomaly detection in medical imaging, detection of industrial damage, detection of anomalies in sensor data, image or video processing, etc. (Mehrotra et al., 2017; Mohan et al., 2017; Tran et al., 2019):

- Intrusion detection systems. Intrusion detection refers to detecting malicious activity (penetrations, break-ins, and other forms of computer misuse) about data collected about network traffic or other user actions in many computer systems. Detection of these malicious activities is essential for computer security (Ratnayake et al., 2020; Shaukat et al., 2020).

- Fraud detection. It detects fraudulent activities in commercial entities such as banks, mobile phones, insurance agencies, the stock market, etc. Among these activities, credit card fraud could be cited, such as unauthorized use of card banking, mobile phone fraud, such as high volume of calls, and auto insurance fraud, such as unauthorized and illegal claims. Organizations are interested in immediately detecting these frauds to avoid economic losses. Data in this domain typically consist of records defined on multiple variables (Bindu et al., 2016; Heydari et al., 2016; X. Ma et al., 2021).
- Medical diagnosis. Many medical applications collect data from various devices, such as Positron Emission Tomography (PET) scans. This data may be abnormal due to abnormal patient conditions, instrumentation faults, or recording faults. In addition, data in this domain can be temporal and spatial (Fahim et al., 2019; Tsehay, 2019).
- Detection of industrial damage: anomalies are related to faults in mechanical components such as engines, oil flow in pipelines, turbines, or other mechanical components. The data collected in this area has a temporal aspect (Ahmed et al., 2021).
- Sensor networks. In many real-world applications, sensors are often used to track various environmental and location parameters. Anomalies in sensor data refer to sensor faults or unforeseen events such as intrusions (Abdelrahman et al., 2020; Al-Amri et al., 2021; Alsoufi et al., 2021; Manimurugan et al., 2020; Vangipuram et al., 2020). Sensor data can be binary, discrete, continuous, audio, video, etc.
- Image and video processing applications. Anomalies in this domain correspond to the detection of rare or unknown movements, such as the detection of abnormalities in video surveillance applications, or to regions that appear abnormal on the static image, such as the analysis of imagery via satellite (Alfergani, 2021; Pérez, 2021; Yuan Wang et al., 2019).

4.2.2.3. Type of Anomalies

In the literature, a general classification of anomalies applies in several application fields and can be divided into three main types: punctual, collective, and contextual (V Chandola et al., 2007, 2009).

- **Point Anomalies**

Point (or global) anomalies are when a data point is considered an outlier because it is sufficiently different or far from the data set. Figure 4.3 shows an example of a building energy consumption time series. An observation with a very high value (overconsumption) relative to a building's usual consumption range presents a point or point anomaly.

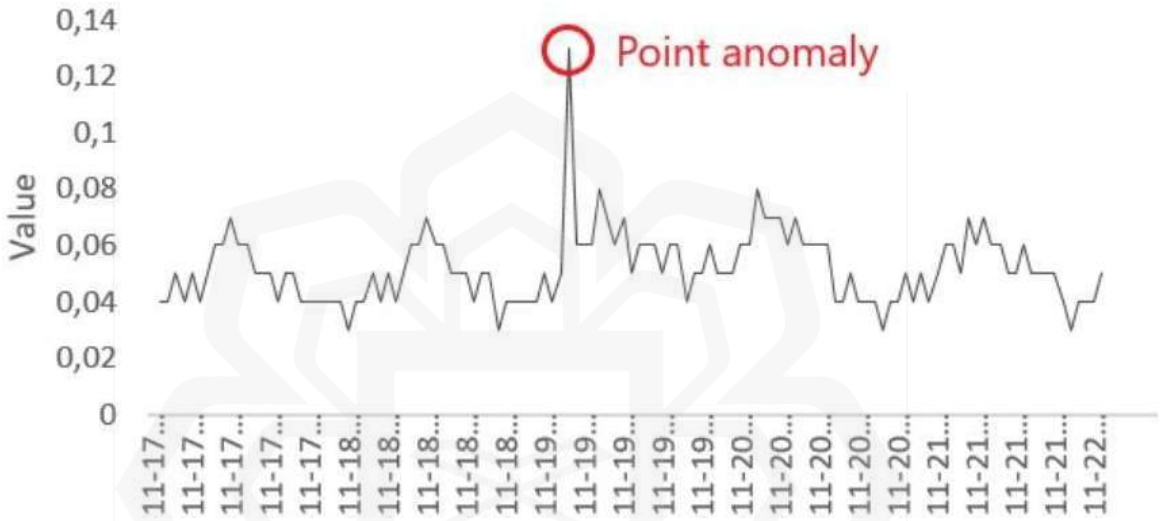


Figure 4. 3 Point Anomaly in A Building Energy Consumption Time Series

- **Contextual Anomalies**

Contextual (or local) anomalies correspond to a data point (or sequence of points) different or distant from other data points but in a specific context (spatial or temporal). For example, Figure 4.4 shows a contextual anomaly in a monthly temperature time series. A low temperature in winter at t1 is considered normal, while the same case might not be normal in high summer at t2.

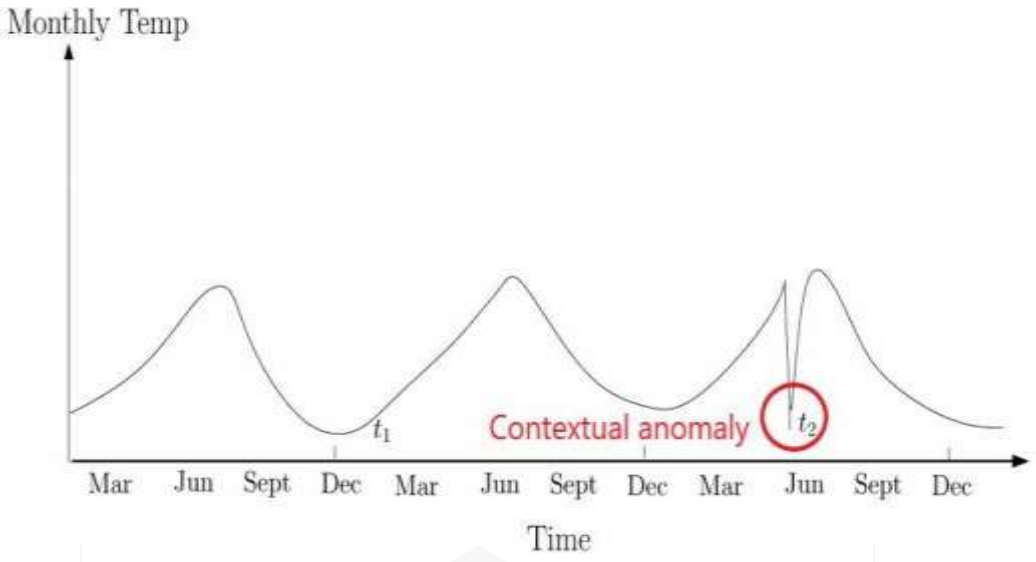


Figure 4. 4 Contextual Anomaly in A Monthly Temperature Time Series (Chandola et al., 2009).

- **Collective Abnormalities**

Collective (or sequential) anomalies correspond to a collection of observations different from the data set. Such as, Figure 4.5 shows an example of a time series containing an anomalous subseries because it differs from the set of subsequences in the time series. This may correspond to a stopped counter, which fails to upload data.

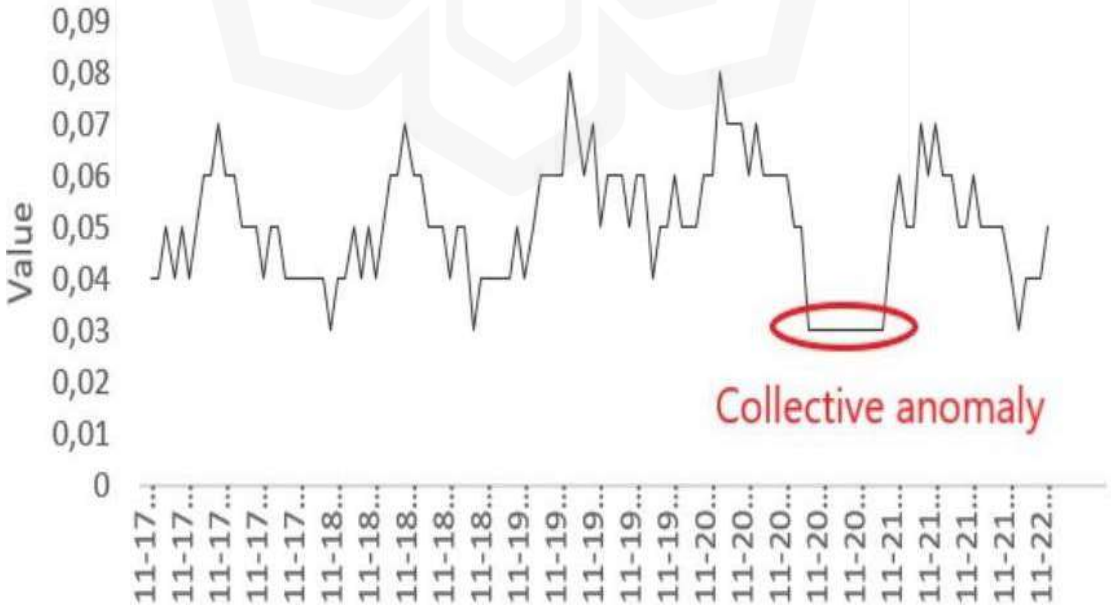


Figure 4. 5 Collective Anomaly Corresponding to A Meter Stoppage

4.2.2.4. Type of Anomalies in Actual Deployments

From state-of-the-art, Table 4.1 has been constructed, which shows the equivalence between the types of anomalies observed in real deployments, which had sought to detect, and the types of anomalies addressed in the literature.

Several terminologies exist to identify the types of anomalies, illustrated in Table 4.1. In this work, types of anomalies had been deeply searched in sensor networks, namely, anomalies caused by faulty sensor readings (e.g., damaged sensors, change of sensor) or unforeseen events such as a pure outage (e.g., the problem of communication or false alarms). Thus, we sought to detect the following anomalies:

- Positive or negative peak. It is an abrupt change in sensor readings measured between two successive samples (represented by triangles in Figure 4.6. It can associate it a point (global) anomaly and a short anomaly. (Al-khatib et al., 2020; Sharma et al., 2020) defined it as an Additional Outlier (AO).

Table 4. 1 Comparison between anomalies observed in real deployments and the anomalies detected by the algorithms of the literature

Reference\Anomalies	Peaks	Noise	Plate	Level Change
Al-khatib et al., (2020)	AO	TC	-	LS, SLS
Sharma et al., (2020)	AO	TC	-	LS, SLS
Jaganathan et al., (2019)	-	short duration	long duration	-
Owido et al., (2013)	-	short duration	long duration	-
Yeh t. al. (2016)	-	discords	-	-
Chen Zhan (2008)	-	collectives	-	-
Fermans et al. (2019)	-	contextual	-	-
Adams et al., (2019)	-	TC	-	-
Rosner(1983)	global	local	-	-
Luo et al., (2021)	-	TC	-	-

- Noise anomalies. It is an increase in the variance of the sensor readings, as shown in Figure 4.6. Unlike short anomalies that affect only one sample at a time, noise anomalies affect several successive samples. This anomaly presents the discord time series (Jaganathan et al., 2019; Owido et al., 2013). (D. (Daphne) Yao et al., 2017) defined this anomaly as a short-term anomaly in the sensor's readings. Various definitions have also been proposed, such as local anomalies (Adams et al., 2019; Luo et al., 2021) or temporary change (TC) (C. Chen & Liu, 1993; Liu & Chen, 1993).
- Constant anomaly (plate): the sensor reports a constant value for several successive samples. These are abnormal readings at a constant offset (illustrated by a rectangle in Figure 4.6). (Y. Chen et al., 2017) defined this type of outlier as a long-lasting anomaly due to a relatively long change in sensor readings. This anomaly can be considered collective (M. Ma et al., 2021; Tornai et al., 2016).
- Level change: This sudden change in the sensor measurements, represented by a cross in Figure 4.6, causes a change in permanent or temporary level in the time series by a certain amplitude from an observation (Balke, 1993). This type of anomaly has been defined as Level Shift (Fillatre, Retraint, et al., 2005) and divided into two variations (Cogranne et al., 2013, 2014; Fillatre, Processing, et al., 2005): level shifts (Level Shift (LS)) and Seasonal Level Shifts (SLS).

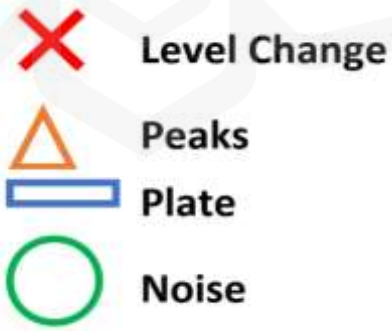


Figure 4. 6 Example of anomalies in sensor measurements.

4.2.3. Automatic Learning for Detection Anomalies

The available data influences the anomaly detection techniques that can be applied. Indeed, data instances can be labeled (there is a label at each data point giving information if the instance's class is normal or abnormal) or unlabeled. Detection can

then be done according to the three principles known in machine learning, namely: unsupervised, supervised, and semi-supervised (J.-R. Jiang et al., 2021; Yu et al., 2021).

- Unsupervised anomaly detection: unsupervised learning is used in cases of not labeled data. This approach makes it possible to determine outliers without prior knowledge of the data. Techniques that are operating in unsupervised mode do not require training data but assume that normal behavior is the most common. The advantage of this method is that no labeled data is needed, and it is widely applicable in different fields.
- Supervised anomaly detection: this approach requires a training data set that contains data labeled as normal or abnormal. The difficulty of using supervised learning is that it often takes a lot of time to assign labels to data. In addition, it is typically challenging to include all types of anomalies, which is necessary for the algorithm to work correctly.

  Its advantage is that it can be used when anomalies are more frequent than typical instances. In addition, unlike unsupervised methods, supervised methods are designed to detect application-specific anomalies.
- Semi-supervised anomaly detection: this approach assumes that the training data contains partially labeled instances, for example, for only the normal class. Finding data that spans all regular instances, like supervised mode, can be difficult.

The main challenge with supervised learning is that it requires a large amount of labeled data, which can be time-consuming to obtain. Unsupervised methods, on the other hand, do not rely on labeled data and are often used for exploratory purposes. These methods can discover outliers, but it is up to the analyst to determine their significance in the specific context of the application. This study focuses on both supervised and unsupervised methods for anomaly detection.

4.2.3.1. Taxonomy of Detection Techniques Anomalies

Most of the existing research focuses either on several application domains or on a single domain of application, as in the case of these journals (S. Agrawal et al., 2015; V Chandola et al., 2009; Hodge et al., 2004; Wu et al., 2020). The authors discussed several anomaly detection techniques in these studies depending on the application domain. Some authors have chosen methods appropriate for detecting particular types of abnormalities (Sharma et al., 2020). Thus, these authors explored the anomaly

detection techniques suitable for detecting anomalies (short, noise, and constant). Others present in their article a taxonomy for anomaly detection techniques concerning several types of datasets (simple and complex) (Bulusu et al., 2020; Sebestyen et al., 2018) provide a more recent review of advances in anomaly detection. Recent studies of anomaly detection methods in univariate (Mohan et al., 2017; Savage et al., 2016; Thill et al., 2019; D. (Daphne) Yao et al., 2017) and multivariate (Adams et al., 2019; Adhikari et al., 2012; Chowdhary, 2017) time series have been proposed.

Given the extensive literature on anomaly detection, a group of the work has been used in knowledge-based approaches, spectral decomposition analysis, information theory, clustering, regression, classification, pattern mining, nearest neighbors, and statistics, as illustrated in Figure 4.7 (V Chandola et al., 2009; García et al., 2015; Liang et al., 2020; Tao et al., 2019; Triguero et al., 2016). Figure 4.7 briefly explains the techniques and focuses on the main methods, particularly those applied to sensor data.

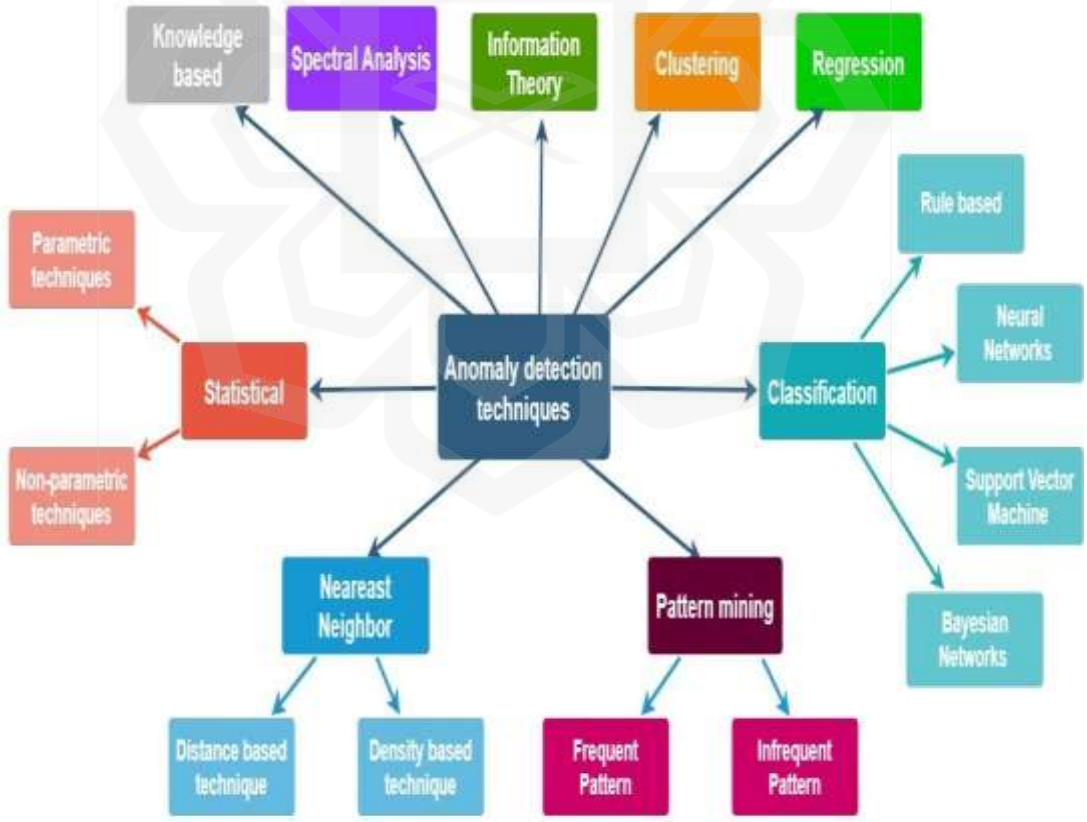


Figure 4. 7 Techniques for detecting anomalies in static data

- **Knowledge-Based Techniques**

This approach (also called an expert system) is based on human domain knowledge. It assumes that the patterns of anomalies are already known and can be defined in a way that is understandable by the machine. The process of using a knowledge-based system to detect anomalies includes three steps(Aldweesh et al., 2020; Alsoufi et al., 2021). Firstly, an expert must go through a large amount of data and recognize patterns that indicate anomalies. Secondly, these anomaly models are then programmed into an automated system. Lastly, the system can then automatically detect any anomalies in new data. The expert's knowledge can be incorporated into the system in various ways (Z Chen et al., 2021; Salvador et al., 2004; D. Shipmon et al., 2017; D. T. Shipmon et al., 2017):

- Rule-based systems: employ a set of "if-then" statements linked to particular events. They operate under the assumption of data predictability, where identical conditions consistently yield the same results. While these systems are relatively simple to execute, it is crucial to ensure full coverage of all potential rules.
- Systems that use statistical methods involve creating decision rules based on basic statistical calculations. For example, two rules can be defined for the specific types of anomalies to be detected (Sharma et al., 2020). One rule may be for short anomalies and involve analyzing a time series by comparing each time two consecutive observations.
- An anomaly is detected if the calculated standard deviation is above a certain threshold. These types of systems are based on statistical calculations and the rules are determined by comparing observations or calculating statistical values such as standard deviation. They are simple to implement, but the rules need to be comprehensive.
- Finite state machine: this approach offers a formalized manner for representing expert knowledge. It entails designing a series of events that are efficiently encoded within a straightforward graph structure, enhancing comprehensibility and usability for experts. Furthermore, the state machine can be employed to condense an extensive array of rules, enhancing manageability for experts.

- **Techniques Based on Statistics**

These techniques fit a statistical model (usually for normal behavior) to the given instances and then apply a statistical inference test to determine whether or not a new

instance belongs to that model. Based on the applied test statistic, instances with a low probability of being generated from the learned model have declared anomalies. Statistics-based approaches are categorized into parametric and non-parametric approaches (Ben Kraiem et al., 2020; L. Chen et al., 2017).

- Parametric techniques assume distribution knowledge and estimate the given instances' parameters.
- Non-parametric techniques generally do not assume knowledge of the underlying distribution and are based on building the distribution model.

The generalized Extreme Studentized Deviate (ESD) test (Ben Kraiem et al., 2020; L. Chen et al., 2017) and the change of point (Aminikhanghahi & Cook, 2017; Du et al., 2021; D. Shipmon et al., 2017) had been proposed for the detection of anomalies on united-varied data. The ESD algorithm used statistical functions such as mean and standard deviation for anomaly detection. However, ESD requires specifying an upper bound for the likely number of existing anomalies; this is impossible for all applications. (Hochenbaum et al., 2017) Therefore, it created an algorithm named Seasonal Hybrid ESD (SH-ESD), which relies on the generalized ESD test to detect anomalies. In addition, SH ESD can be used to detect global and local anomalies. This is achieved using time series decomposition and statistical metrics (median and ESD).

The Change Point method detects distribution changes (e.g., mean, variance, covariance) in sensor measurements (Rosner, 1983). This method detects each change in the form of anomalies.

- **Regression-Based Techniques**

These techniques are widely used on temporal data. This approach is based on the principle of predicting. The anomaly can be defined as a discrepancy between reality and what is expected. The principle of this approach works in the following mode. First, it estimates future data.

Then, Regression-based techniques quantify the difference between the actual and predicted values. The new data point is considered an anomaly if the deviation is large enough (more significant than a predefined threshold). To predict new values, we need an autoregressive model such as Auto Regressive Integrated Moving Average (ARIMA) proposed by (P. Chen et al., 2020). Many researchers have applied ARIMA to detect anomalies in different contexts (M. Alizadeh et al., 2021; Aadyot Bhatnagar et al., 2021; Zhipeng Chen et al., 2022; Ren et al., 2019; Salvador et al., 2004). ARIMA models are

known to be very accurate in forecast values and scalable to seasonal time series. ARIMA detects four types of anomalies: AO (Additive Outlier), IO (tool innovation), TC (Temporary Changes) or LS (Level Shift), and Seasonal Level Shift (SLS). However, this accuracy depends on the model's order selection (auto-regressive orders, differentiation, and moving average). The main weakness of ARIMA is that it does not have an efficient model update procedure. This makes ARIMA computationally expensive.

- **Techniques Based on Classification**

Most classification-based anomaly detection techniques work in a supervised or semi-supervised environment. They use a labeled dataset (training) to train a model or classifier. This model is then used to classify the new (test) points into one of the classes (normal, abnormal). As illustrated in Figure 4.7, classification-based approaches are classified into four categories: neural network-based approaches (Ozyildirim et al., 2018; Zarzour et al., 2020), Bayesian network-based approaches (R. Alizadeh et al., 2020; Rashidi et al., 2011; Scarpiniti et al., 2021), support vector machines (Hejazi et al., 2013) and rule-based approaches (Duffield et al., 2009; Zarzour et al., 2020), depending on the type of classification model they use.

- The Support Vector Machine (SVM) is a popular algorithm for classifying data in anomaly detection systems, particularly when working with numerical and time series data (Aparecido et al., 2018; J. Jiang et al., 2013; Mounce et al., 2011). SVM creates a boundary between normal and abnormal data using a core function. It is known for its high classification accuracy, especially when using nonlinear kernels such as polynomial. However, when using nonlinear kernels, SVM is prone to overfitting problems.
- Neural network classifiers were used in anomaly detection systems, such as intrusion detection (Fernandes et al., 2019; P. Singh et al., 2019). Researchers have used a combination of different neural network algorithms, such as the C-LSTM, to analyze web traffic data and detect anomalies (Kim et al., 2018). This method involves using a CNN to model the spatial information in the data and LSTM to model the temporal information. The output of the final LSTM layer is then fed into DNN layers for classification. While this approach can achieve high classification performance, the disadvantage is that the decision-making

process of the neural network is not transparent, making it difficult for humans to interpret the results.

- Bayesian Networks are a formalism that fuses probability theory and graph theory. They consist of a set of variables (network node) and a set of arcs between the variables. In this technique, the Bayesian structure is learned from the data and the parameters of the Bayesian network are estimated. This algorithm was used by (Kim et al., 2018) to extract interesting moments in Formula 1 videos. This approach uses a Bayesian network structure built by hand from knowledge of the treated domain. The problem is that knowledge about the different relationships that exist between variables is not always available. In (Rashidi et al., 2011), the authors presented a method to find anomalies in categorical or mixed datasets in an unsupervised way. This technique cannot detect or explain more complex anomalies resulting from conflicts between large sets of nodes rather than individual nodes.
- Rule-based techniques involve creating decision rules based on a set of "if-then" statements that represent the normal state of the system. The advantage of using a rule-based approach is that the rules clearly explain why a particular value is considered abnormal, similar to the way a decision tree works (Aadyot Bhatnagar et al., 2021; Dudde et al., 2014; D. Gupta et al., 2012; Jidiga et al., 2014; Manoj Gadiyar et al., 2016; Popescu et al., 2015).

Classification Rule

Car Color = Blue = Yes & Model = 2015 –> Buy

Car Color = Blue = Yes & Model = 2015 = No & Mileage < 500000KM -> Buy

Car Color = Blue = Yes & Model = 2015 = No & Mileage > 500000KM -> Do not Buy

Car Color = Blue = No & Color = Red = No –> Do not Buy

Car Color = Blue = No & Color = Red & car = Ferrari = Yes –> Buy

Car Color = Blue = No & Color = Red & car = Ferrari = No –> Do not Buy

Figure 4. 8 Classification technique based on decision tree rules

In general, systems based on decision trees are easily understood because they clearly show the sequence of tests leading to classifying a point as normal or abnormal, as shown in Figure 4.8. Its “white box” model is easy to conceptualize, visualize and interpret the result and also makes it possible to generate understandable decision rules. Although rule-based approaches are quick to test, this might not always be the case when there is a large and complex set of rules, making the generated rules less interpretable. Numerous classifiers based on decision trees have been discovered (Dudde et al., 2014) for anomaly detection, such as Best-first, Functional Tree, Decision Tree, Random Forest, Logistic Model Tree, and J48. Their study showed that the Random Forest decision tree outperformed other classifiers based on the decision tree regarding the correct classification rate.

The main problem with decision trees is the high risk of over-fitting by reducing a composite classification issue with extremely multidimensional data. To circumvent this problem, random forests were proposed by (Cho & Nam, 2019; D. Gupta et al., 2012). A random forest consists of creating several decision trees, each on a randomly selected subset from the input data set. The output of a random forest is the average of all decision trees (P. Chen et al., 2014; Liu & Chen, 1993).

- **Techniques Based on Pattern Mining**

These approaches could be classified into two categories: the exploration of frequent models and the exploration of rare models.

- Frequent pattern or pattern mining refers to extracting informative and useful patterns in data sets. The goal is to find frequent patterns in the data, serving as a template for frequently observed normal behavior (S. Abghari et al., 2018; S. Z. Abghari et al., 2018; Kuchar et al., 2017). The task of exploiting frequent patterns appears in many domains. A typical application is market basket analysis, the objective of extracting sets of items purchased frequently together in a supermarket by analyzing customer baskets. Once the extraction the frequent sets done, then allow to extract association rules among the sets of items, where a statement could be made about the probability that two sets of items occur conditionally or occur (Jr et al., 2014; Saarela et al., 2016). Many algorithms had used the extraction of frequent patterns for anomaly detection, such as Pattern-Based method for Anomaly Detection (PBAD) (Feremans, Vercruyssen, Meert, et al., 2020), Pattern based Outlier Detection (POD)

(Feremans, Vercruyssen, Cule, et al., 2020), Frequent pattern based outlier detection (FP-outlier) (He et al., 2005) and Matrix Profile (MP) proposed by (Yeh et al., 2016).

- Exploration of infrequent patterns has drawn the attention of the data mining research community, which aims to discover rare associations because infrequent patterns are more interesting than frequent patterns in anomaly detection or novelty (Otey et al., 2006; Pang et al., 2016; V Chandola et al., 2007, 2009; Varun Chandola, 2009). A multi-scale anomaly detection; Pattern Anomaly Value (PAV) algorithm based on infrequent linear models have been proposed (L. Chen et al., 2017; Xu Zhang et al., 2019). Anomaly models are infrequent models with lower support than other time series models.

- **Techniques Based on Nearest Neighbor**

These approaches are classified into two categories: techniques that use the distance of a data instance at its k^{th} nearest neighbor as an anomaly (L. Chen et al., 2017; Kraiem et al., 2019) and techniques that calculate the relative density of each instance of data to calculate its anomaly score, for example, the Local Outlier Factor (LOF) algorithm (Acuna et al., 2004). The density-based outlier detection method estimates the density of the neighborhood of each data instance. An instance in a low-density neighborhood is declared abnormal, while an instance in a dense neighborhood is declared normal. (M. Breunig et al., 2000; M. M. Breunig et al., 2000) proposed the LOF (Local Outlier Factor) algorithm. In this approach, an outlier is measured using a local outlier factor (LOF), which is the ratio of that point's local density to its nearest neighbor's local density. The data point with the high LOF value is declared as aberrant.

- **Techniques Based on Partitioning**

The partitioning-based method groups similar data instances together to form clusters. Partitioning is an unsupervised or semi-supervised technique. Partitioning-based anomaly detection techniques can be grouped into three categories based on the following assumptions:

- Data instances that are considered normal belong to a group or cluster within the data, while anomalies do not fit into any cluster.
- Normal data instances are near the nearest cluster center of gravity, while anomalies are far from the nearest cluster center of gravity.

- Normal data points are typically grouped together in large and densely populated clusters, while anomalous data points tend to belong to smaller or less dense clusters.

Many clustering algorithms have been used to detect anomalies (Hardin et al., 2004), such as DBSCAN, ROCK, K-Means, etc. One of the most widely used algorithms is K-means (Gaddam et al., 2007; Muniyandi et al., 2012; Münz et al., 2007). The principle of the K-means algorithm is as follows: it starts by selecting the centroids and creating clusters around them by assigning each data point to its nearest centroid. Then the centroids and clusters are updated repeatedly until convergence. (Gançarski et al., 2020) Propose a platform, FODOMUST, for collaborative clustering under incremental constraints of time series. It contains methods, libraries, and interfaces dedicated to clustering complex data. Although k-means is simple to implement and easy to interpret, it has some flaws, like the number of clusters that must be chosen before executing the algorithm and especially the spherical shape of the clusters it assumes (El Malki et al., 2020).

- **Techniques Based on Information Theory**

Information theory techniques analyze the information content of a dataset using different information-theoretic measures such as Kolmogorov complexity, entropy, relative entropy, etc. This approach assumes that anomalies in a data set alter its information content. Entropy is the most widely used indicator of information in information theory, which quantifies the uncertainty or randomness of data.

Many researchers have used entropy to detect anomalies (Khan et al., 2017; Martos et al., 2018; Sevil, 2020; Velarde-Alvarado et al., 2016), but the problem with this technique is its outstanding sensitivity to the presence of noise.

- **Techniques Based on Spectral Analysis**

Spectral techniques attempt to approximate the data by using a combination of attributes that capture most of the variability in the data. This approach assumes normal and abnormal points are easily separable in lower dimensional space. Consequently, it projects the data into this space. Among the techniques of spectral analysis, the quote could be of the analysis in principal components (PCA) (R. Jiang et al., 2010; Rubinstein et al., 2016; Tuor et al., 2018), Fast Fourier Transform (FFT) (M. Chen et

al., 2014), wavelet transform (Abramovich et al., 2000; S. J. Yao et al., 2000) and Hough transform (Kaur, Saxena, et al., 2010; Kaur, Technology, et al., 2010).

4.2.4. Valuation Methods

This section provides an overview of the valuation methods used to evaluate the performance of an anomaly detection algorithm. The comparison and evaluation of the results of the anomaly detection methods are based on analysis of observations that have been rightly or wrongly detected as anomalies or as normal.

4.2.4.1. Confusion Matrix

A confusion matrix is a table that gathers the results obtained from applying a machine learning algorithm (supervised or unsupervised). In the case of anomaly detection, four quantities in the confusing matrix had presented as shown in the following Table 4.2:

Table 4. 2 Confusion Matrix

		Predicted Class	
		Normal	Attack
Actual Class	Normal	True Negative (TN)	False Positive (FP)
	Attack	False Negative (FN)	True Positive (TP)

The behaviors qualified in this work as Negative are the normal behaviors, and Positive for the cases of anomalies.

- True negative (TN): a normal value correctly labeled by the algorithm.
- True positive (TP): an anomaly correctly predicted/classified by the algorithm.
- False positive (FP): a normal value wrongly considered as an anomaly.
- False negative (FN): a labeled anomaly, wrongly considered as a nor-value male.

4.2.4.2. Evaluation Metrics

Several metrics can be inferred from the confusion matrix. Among those which are used frequently, precision has been found (also called value positive predictive), recall (also known as sensitivity), and F-measure, which is a compromise (harmonic mean) between recall and precision (Alsoufi et al., 2021; Pérez, 2021).

- Accuracy: quantifies the relevance of the points detected as anomalies. It measures the probability that an observation classified as Positive is Positive.

$$P = TP/(\mathrm{TP} + \mathrm{FP}) \qquad (4.1)$$

- Recall: quantifies the ability of an algorithm to detect existing anomalies aunts. It represents the rate of True Positives, i.e., the proportion of anomalies correctly classified.

$$R = TP/(TP + FN) \qquad (4.2)$$

- F-measure (or F1 Harmonic Mean) of precision and recall gives the algorithm performance.

$$F\text{ - }measure = 2PR/(P + R) \qquad (4.3)$$

These metrics are used most to evaluate time series classification algorithms (Tatbul et al., 2018). In this thesis, these measures were used to evaluate anomaly detection approaches. System evaluation was done by verifying posteriori if the classes assigned to the observations were correct.

Table 4. 3 Commonly Used Algorithms

Techniques	Algorithm	Anomalies	Input	Exit	Learning Methods	Interpretable	Reference
Knowledge	Rules	Short, constant	time-series, numeric	label	non-supervised	yes	Sharma (2010)
near neighbors	Density (LOF)	global, local	numeric	score	non-supervised	no	Upadhyaya (2012)
	Distance (KNN)	global	numeric				Breunig (2000)
Statistics	SH-ESD	global, local	time-series	label	non-supervised	yes	Hochenbaum (2017)
	Point Changes	Level-shifts	time-series	labels	non-supervised	yes	Basseville (1993)
Regressions	ARIMA	LS, TC, AO	time-series	labels	non-supervised	yes	Chen (1993)
Classification	Decision tree	local, global	numeric/ categorical	classes	supervised	yes	Sinwar 2016)
	Random Forest					no	Breiman (2001)
	neurons network					no	Sreevidya (2014)
Exploration of motifs	PBAD	contextual	time-series	score	supervised	no	Feremans (2019)
	PAV	collective				no	Chen (2008)
	MP	discords				no	Yeh (2016)
Exploration of motifs	**ARP**	**pics, noise, constant, level shift**	**time-series**	**label, type**	**non-supervised**	**yes**	**Ben Kraiem et al. (2019)**

4.2.5. Common Algorithms Findings

The Table 4.3 presents state-of-the-art research aimed at solving the problem of anomaly detection. The comparison of these works had done according to the following criteria:

- The technique used for each detection algorithm;
- The type(s) of anomalies considered by each method;
- The input of each method corresponds to the type of data processed;
- The output describes the anomalies, which can be labels, scores, or classes;
- The learning between supervised and unsupervised according to the availability of labels;
- The interpretability indicates whether the experts understand each method's result.

As shown in Table 4.3, we can see that some algorithms simply process the data and give interpretable results through rules or even labels indicating the anomaly. Others are more complex and less easily interpretable. When it is, for example, a black box or when the result is a score that requires work to be able to interpret the result. Additionally, these approaches address a few specific anomaly types but do not cover all the anomaly types that may observe in real deployments.

This work (last row of Table 4.3) makes it possible to overcome these shortcomings through a method, ARP, which makes it possible to finely detect multiple anomalies of different types and generate the label and the type of anomaly found as a result. Also, it proposes a method for automatic labeling of time series. Moreover, based on the patterns, this method is interpretable by the experts since the patterns describe the remarkable points and potential anomalies.

4.2.6. Recap of the Section

In this section, it had been presented the main concepts necessary to understand the context and the problem addressed in this thesis.

The presented principles of anomaly detection, starting with the definition and types of anomaly by specifying the types of anomaly specific to the field of application, then the type of learning, the different detection techniques, and the evaluation methods to compare and evaluate anomaly detection methods. Then, it summarized all the work and built a summary table comparing the different literature outcomes concerning this work. The following part presents the practical implementation of detecting anomalies.

4.3. ENHANCING TECHNIQUES AIMED AT DETECTING APT ATTACKS

4.3.1. Pattern-Based Methods for Anomaly Detection

This second part of this chapter presents the proposed approaches for detecting anomalies. In achieving this, the research begins by introducing the context and motivation of the strategy and the basic notions used in this work before detailing the proposed contributions. Then, it presented the first pattern-based approach for multiple anomaly detection. Finally, a detail of the second supervised approach, rule extraction for anomaly detection, had done.

4.3.2. Context and Motivation

Sensor networks play an essential role in daily life activities (e.g., smart watches, android/iOS-based mobile devices, and smart appliances) at the campus scale and, more broadly, in a city, region, or country. Data recorded by sensors measure the operation's effect on these interconnected devices. These data include anomalies that affect supervision (e.g., false alarms, interruption requests), such as the anomalies illustrated in Figure 4.9. This work occurs within the framework of real applications with specific anomalies within the environment, namely the advanced persistent threats. These anomalies require an analysis phase by the experts and extraction of in-deep related features before applying any policies to the local digital environment. The local environment ultimately seeks to improve the dataflow circulation by setting up a posteriori anomaly detection system. A goal was to process data from local infrastructure and find a method to detect multiple anomalies of different types observed during real deployments while maximizing the number of anomalies detected and minimizing detection errors (FP, FN). So, in this context, a dealt with univariate time series posteriori.

In an actual operating situation, the supervision of sensor networks is done by experts (network admins, system admins, etc.) by observing the curves to detect remarkable points corresponding to unusual behavior. Typically, these are the nodes' measurements in this case study illustrated in Figure 4.9. These remarkable points are the unusual variations between successive points in a time series; they constitute the markers (or indices) of possible anomalies through knowledge of the history and nature of data, which is the analysis of the neighborhood of the remarkable points to decide the instances that represent a true anomaly.

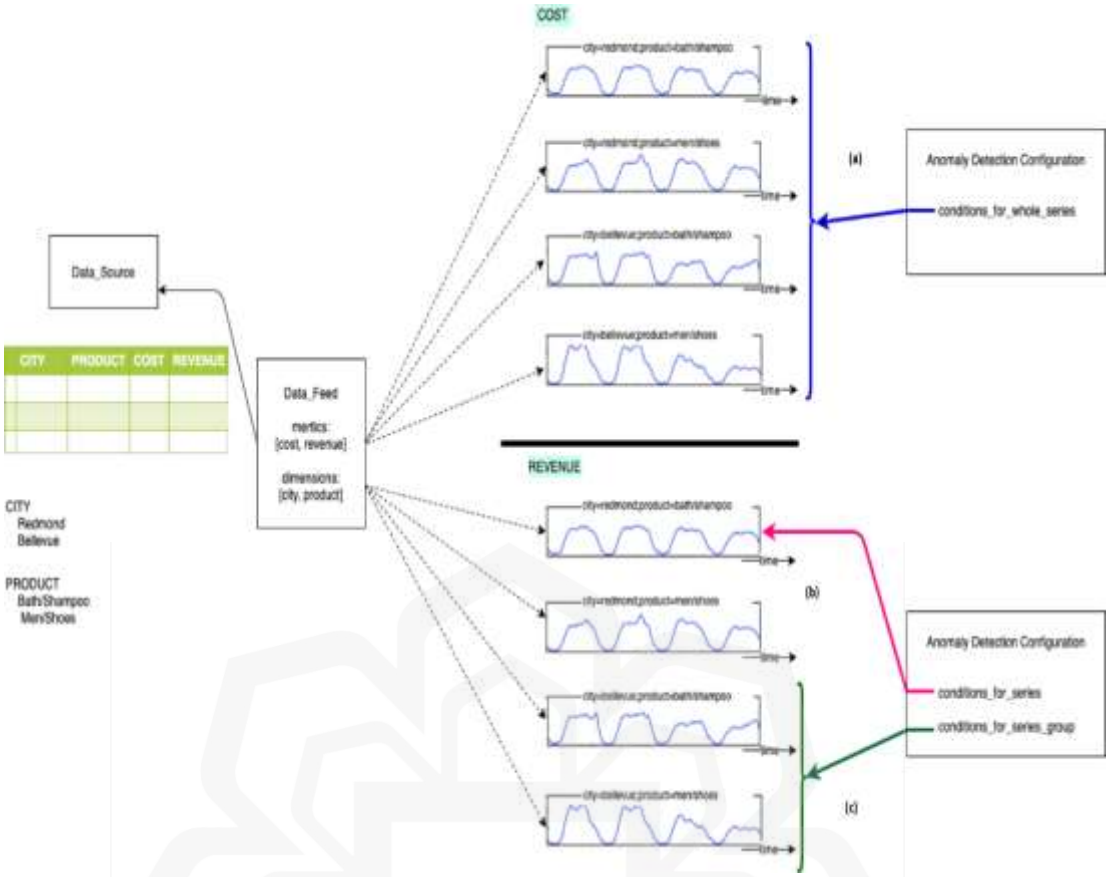


Figure 4. 9 Example of Anomalies Observed in Real Deployments

The aim is to automate this process to detect anomalies in the first place, generate classification rules to interpret in the second place, and insert an IDS database.

4.3.3. Types of Anomalies

As seen in the introduction, the choice of a technique for detecting anomalies was made in particular according to the data processing. In the context of this work, an applied anomaly detection had been chosen for monitoring data from the supervision system managed in a local environment.

For example, Figure 4.9 (a) illustrates an abrupt change in a node flow, resulting in a permanent level change. Figure 4.9 (b) shows several peaks representing reading defects related to an unforeseen event (e.g., external noise). Finally, Figure 4.9 (c) depicts a constant lag in the measurements (due to a communication problem).

This section looks for four types of anomalies observed in real deployments, as illustrated in Figure 4.9. Thus, an anomaly can manifest itself by:

- An offset or a change in level in the node flow measurements is associated with

exceeding a value threshold. This is a considerable lag between the initial values and the values following this change, generated due to an event affecting a series at a given time.

- An abnormal change or an unexpected frequency of variation in node flow measurements or peaks representing reading faults related to an unforeseen event (e.g., external noise). In this case, two types of anomalies had been looking for, namely a positive peak and a negative peak with a substantial variation compared to the other instances of data or a slight variation.
- A sudden variation or noise in the node flow measurements is linked to an unforeseen event (e.g., external noise). Moreover, this variation can affect several successive observations.
- A constant value or a constant offset in the measurements is due to a communication problem.

These constant values could be an anomaly for some successive samples with or without shift compared to the previous values.

4.3.4. Approach Notations

This part presents the basic definitions and notations used in these approaches.

Definition: A univariate time series is composed of successive observations or points collected sequentially at regular intervals.

These points represent measurements associated with a timestamp indicating the time of its collection. A time series is defined as $T_s = \{x_1, x_n\}$ where $\forall i \in [1..n]$, $x_i \in R$ such that the xi values are uniformly spaced in time and n is the size of T_s.

Definition: A point (or measurement) is composed of a value and a timestamp. It denotes as a point $xi = (t_i, v_i)$ such that t_i is the timestamp of x_i (denoted $t(x_i)$) and v_i is the value of x_i (denoted $v(x_i)$).

4.4. DEVELOP OF AUTONOMOUS APT DETECTION EFFICIENT MODEL

4.4.1. Detection Methodology Based on the Anomalies Remarkable Points

To monitor the data from the sensors, the experts analyze the curves and detect remarkable points in the time series that can be considered exciting patterns to detect outliers of different types.

The illustration seen in Figure 4.2 (mentioned earlier in this chapter) is an example of the sliding window process with a window size of 4. Each x represents the daily observation of time series data (1,2,3, N). Thus, all the possible noisy detections were classified as an anomaly from such an approach.

This section presents the pattern-based approach for detecting multiple anomalies that may occur in a time series. It makes it possible to finely identify different types of anomalies by specifying the location and the type of anomaly.

4.4.1.1. Context and Motivation

Various methods for detecting anomalies have been proposed in literature, which are classified based on their specific use or the type of anomalies they are designed to detect (V Chandola et al., 2009). However, these techniques may not always be effective in detecting all types of anomalies, thus requiring the use of multiple methods to detect different kinds of anomalies. Current approaches also have limitations in detecting different types of anomalies and have limitations when applied to multiple anomalies (Munir et al., 2017; Riedel et al., 2010; Sharma et al., 2020; X Zhang et al., 2020). The difficulty of having a robust technique to detect all the anomalies has led to defining a new configurable named **ARP** "Anomalies Remarkable Points." This approach involves identifying unusual remarkable in a time series by evaluating patterns, and then using this information to detect multiple anomalies by analyzing combinations of these remarkable points.

ARP requires application domain expertise to define labels' patterns and compositions effectively.

4.4.1.2. Anomalies Remarkable Points Description

In this thesis, it had been working with experts from the Management and Operations Department on data processing and analysis. They showed their analysis strategy to monitor sensor networks. Indeed, they analyzed the curves from different sensors and detect remarkable points.

These points are remarkable because they have unusual behaviors compared to the rest of the data. These points represent unusual variations between certain successive points in a time series. Then they analyzed the gap between these points. The deviation is important, especially for index data (meter reading) representing increasing time series. Finally, they analyzed the neighborhood of each remarkable point to decide if it represents an anomaly. This strategy is manual, but it represents the expertise and know-how of the experts who arrive by inspecting the curves to detect different types of anomalies and to analyze their causes.

In this context, relying on the experience of experts (detection of remarkable points then identification of anomalies), a proposed the **ARP** algorithm had built in two phases. The first consisted of detecting and annotating the points considered as remarkable in the time series. The second phase involved identifying anomalies from the compositions of remarkable points. It is important to note that **ARP** is generic; it is therefore easily adaptable to other situations, providing different reasons where appropriate.

4.4.1.3. Detection of Remarkable Points

The detection of remarkable points is carried out from the patterns of detection defined and decided in advance based on the literature.

Definition: A pattern p is defined by a triple p = (l, σ_a, σ_b) where l is a label of the remarkable point, σ_a and σ_b are two thresholds used to decide if a given point is remarkable. A pattern is applied to three consecutive points x_i-1,x_i,x_i+1of time series T_s.

σ_a corresponds to the difference between v(x_i-1) and v(x_i), while σ_b corresponds to the difference between v(x_i) and v(x_i+1) as illustrated in Figure 4.10. When a pattern is satisfied on x_i-1, x_i, x_i+1, the point x_i takes the label l of the pattern p. If no pattern is verified, the point x_i is considered normal and labeled with the label “Normal.”

Definition: A time series labeled T_{sL} is a time series of points on which the labels detected by the patterns are added.

Definition: A remarkable point x_i of a labeled time series is defined by a triple (t_i ,v_i , L_i) where t_i is its timestamp, v_i is its value, and L_i = {l_1, l_2, ...} is a list of labels characterizing the point.

Patterns are therefore used to detect remarkable points and associate them with one or more corresponding labels. Thus, the list of labels of a noteworthy point consists of the labels of the different patterns verified on this point.

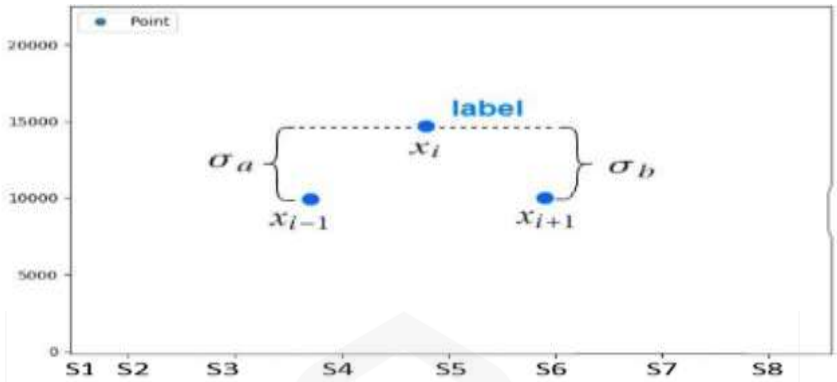


Figure 4. 10 Labeling of a remarkable point "x" *by a pattern σa and σb.*

Table 4. 4 The different patterns for labeling remarkable points in a time series.

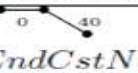
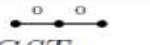
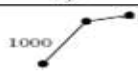
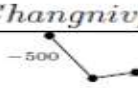
Pattern	Definition	Example
Ptpicpos	$x_{i-1} < x_i \wedge x_i > x_{i+1}$ $\sigma_a, \sigma_b \in R_+^*$	 $Ptppicpos_{80,100}$
Ptpicneg	$x_{i-1} > x_i \wedge x_i < x_{i+1}$ $\sigma_a, \sigma_b \in R_-^*$	 $Ptpicneg_{-100,-100}$
SartCstNeg	$x_{i-1} > x_i \wedge x_i = x_{i+1}$ $\sigma_a \in R_-^*, \sigma_b = 0$	 $SartCstNeg_{-40,0}$
StartCstPos	$x_{i-1} < x_i \wedge x_i = x_{i+1}$ $\sigma_a \in R_+^*, \sigma_b = 0$	 $StartCstPos_{40,0}$
EndCstNeg	$x_{i-1} = x_i \wedge x_i > x_{i+1}$ $\sigma_a = 0, \sigma_b \in R_+^*$	 $EndCstNeg_{0,-40}$
EndCstPos	$x_{i-1} = x_i \wedge x_i < x_{i+1}$ $\sigma_a = 0, \sigma_b \in R_-^*$	 $EndCstPos_{0,-40}$
CST	$x_{i-1} = x_i \wedge x_i = x_{i+1}$ $\sigma_a = 0, \sigma_b = 0$	 $CST_{0,0}$
Changnivpos	$x_{i-1} < x_i \wedge x_i < x_{i+1}$ $\sigma_a \in R_+^* \sigma_b \in R_-^*$	 $Changnivpos_{1000,-80}$
Changnivneg	$x_{i-1} > x_i \wedge x_i < x_{i+1}$ $\sigma_a, \sigma_b \in R_-^*$	 $Changnivneg_{-500,-20}$

Table 4.4 presents the patterns used to label the time series. It defines 9 reasons, namely:

- Ptpicpos: pattern to detect a remarkable positive peak point.
- Ptpicneg: pattern to detect a remarkable point of negative peak type.
- StartCstNeg: pattern to detect a remarkable point of start of constant negative.
- StartCstPos: pattern to detect a remarkable point of start of constant positive.
- EndCstNeg: pattern to detect a remarkable point of end of negative constant.
- EndCstNeg: pattern to detect a remarkable point of end of positive constant.
- CST: pattern to detect a constant remarkable point.
- Changnivpos: pattern to detect a remarkable point of level change positive.
- Changnivneg: pattern to detect a remarkable point of level change negative.

The provided patterns are sufficient for labeling time series data with various specific patterns. They cover a range of scenarios, including detecting peak points (positive and negative), the start and end of constant periods (both positive and negative), constant points, and level change points (both positive and negative). These patterns offer a comprehensive way to label and identify different behaviors within time series data.

The thresholds Ω_a and Ω_b are defined by the experts. They can be configured according to time series and application domain.

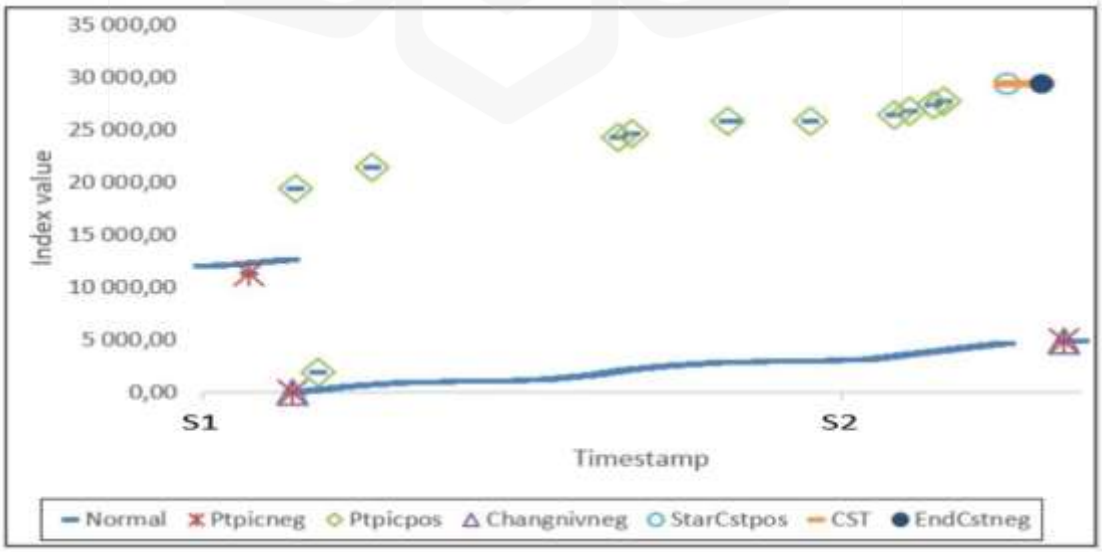


Figure 4. 11 Labeling of a series of remarkable points (algorithm 1) using predefined patterns.

Figure 4.11 illustrates an extract of a labeled time series of index data from a building calorie meter. So as stated, it includes 7 examples of labels with different patterns defined in Table 4.4. A remarkable point can be labeled by several patterns such as the point represented by the cross and the triangle in Figure 4.11 which have 2 labels (Ptpicneg, Chan gnivneg).

Algorithm 1 labeled extraction of a time series in index data

Input: xi-1, xi, xi+1, p = (l, σ_a, σ_b)
Output: Boolean

```
1:  If p. σa>0 then
2:    leftValidate ← (v(xi) ≥ v(xi-1)+p. σa? true : false)
3:  else if p. σa < 0 then
4:    leftValidate ← (v(xi) ≤ v(xi-1)+p. σa? true : false)
5:  else if p. σa = 0 then
6:    leftValidate ← (v(xi) = v(xi-1)? true : false)
7:  end if
8:  If p. σb>0 then
9:    rightValidate ← (v(xi) ≥ v(xi+1)+p. σb? true : false)
10: else If p. σb>0 then
11:   rightValidate ← (v(xi) ≤v(xi+1)+p. σb? true : false)
12: else If p. σb=0 then
13:   rightValidate ← (v(xi) =v(xi+1)? true : false)
14: end if
15: confirmedPattern ← leftValidate and rightValidate
16: return confirmedPattern
```

Figure 4. 12 labeled extraction of a time series in index data

Algorithm 1 (Figure 4.12*)*, called EvaluatePattern, evaluates a pattern using rules. This function takes as input three successive points denoted x_i-1, x_i and x_i+1, and the pattern p to be evaluated, and returns the evaluation result indicating whether the pattern is confirmed. During the evaluation, an applying of different verification rules had been done depending on the signs of σ_a and σ_b (lines 1–14).

Algorithm 2 (Figure 4.13*)* uses the EvaluatePattern function to process a time series. It takes as input the initial time series T_s and the list of patterns P and returns a new time series labeled T_sL. The processing consists of traversing the time series and the list of patterns. For each point xi and pattern pk, the Evaluate Pattern function is called to add (or not) the label l of the pattern pk to the evaluated point. In line 13, the triple (t_i, v_i, L_i) is the remarkable point formed at the point xi = (t_i, v_i) from T_s.

Algorithm 2 Definition of Remarkable Points

Input: Ts = {x1, x2, xn, …}, P={p1, p2, pn, …}
Output: Labeled Time Series T_{s_L}

```
1:  T_sL ← {}
2:  for I in range (2... | Ts | -1) do
3:      Li ← 'Normal'
4:      for k in range(1... | P | ) do
5:          if EvaluatesPattern (xi-1,xi,xi+1) then
6:              if Li = 'Normal' then
7:                  Li ← {pk.l}
8:              else
9:                  Li ← Li ∪ {pk.l}
10:             End if
11:         End if
12:     End for
13: T_sL ← T_sL ∪ {(ti, vi, Li)}
14: End for
15: return T_sL
```

Figure 4. 13 Definition if Remarkable Points

4.4.1.4. Pattern Composition

In order to detect anomalies, analyzed of the neighborhood had made for the remarkable points detected during phase 1. Thus, from a subset of points of a time series label, a construction had made by concatenating the labels L_i of these remarkable points a chain of labels. On this chain conditions' check had been established on point values. An anomaly thus recognized by a composition of labels and a check of the conditions on the points.

Definition: An anomaly is identified by examining a group of remarkable points based on their labels and the values associated with them. The composition of these labels (sequential order of the labels) and the conditions imposed on these values are used to determine if the points belong to an anomalous subset. The anomaly may be present in one or multiple points within this subset. A composition of labels can be defined using a proposed grammar, which illustrates the elements of a composition of labels as in Figure 4.14. This grammar makes it possible to specify the possible labels (one or more) on consecutive points in order to identify a specific combination of labels.

The grammar starts from the labels placed on the points (<label>). Labels can be combined on a single point with logical expressions AND, OR and NOT (<label comp> and <point-label>). For example, "l1 AND NOT l2 AND l3" designates a point labeled l1, unlabeled l2, and labeled with l3. Each combination of labels on a single point can be repeated on successive points by quantifiers: ?, + * (<label-enum>). For example, "(l1) +" means that the composition must include one or more successive points labeled l1.

The final composition of labels is created through a succession of enumeration of labels separated by "." (< composition >). For example, "l1.(l2)*.(l1 OR 13)" means a point labeled by l1 followed by none-labeled by l2 followed by a point labeled by l1 or by l3.

```
<composition> ::= <label-enum> ("." <label-enum>)*

<label-enum> ::= <label-comp>
     |           "(" <label-comp> ")" "?"
     |           "(" <label-comp> ")" "*"
     |           "(" <label-comp> ")" "+"

<label-comp> ::= <point-label> ("OR" <point-label>)*
     |           <point-label> ("AND" <point-label>)*
     |           <point-label>

<point-label> ::= <label>
     |            "NOT" <label>

<label> ::= list of words (remarkable points)
                 defined by patterns
```

Figure 4. 14 Grammar for the definition of a composition of labels.

Definition: A composition of labels making it possible to recognize an anomaly is made up of three parts:

- Composition: the composition of the labels of remarkable points, a sequence of points comprising labels defined according to the grammar presented in Figure 4.14. The same composition of labels can correspond to different anomalies.

- Condition: this is a condition between the values of the recognized points (those corresponding to the sequence of labels). This condition is created using operators (<>, <, <=, =, >, >=), allowing to compare values, and logical operators (AND/OR/NOT) allowing to combine comparisons. In order to avoid the use of the notation $v(x_i)$, we denote by v_i the value of the i^{th} point recognized by the composition, v_1 the first, and v_n the last; note that the number of points involved in the composition can be variable considering the quantifiers that can be used in the composition.
- Conclusion: the anomaly identified for which its type is specified (name of the anomaly) and the list of values (points) where the detected anomaly is located.

As an example, compositions of labels had been given to identifying very recurrent anomalies in sensor data: (i) anomaly of positive peak values. The latter is possibly recognized from two compositions of **labels, Label-composition 1** and **Label-composition 4,** because they have the same composition part. The purpose of this example is to show the usefulness of the condition part in anomaly detection, (ii) anomaly of negative peak values presented by **Label-composition 2**, and (iii) anomaly of constant values presented by **Label-composition 3**;

Label-composition 1

composition: Normal. Ptpicpos . Ptpicneg. Normal

condition: v2 > v4 and v3 > v1

conclusion: positive peak -> v2

The composition "Normal. Ptpicpos . Ptpicneg . Normal" means that there is a point labeled "Normal" followed by a point "Ptpicpos" followed by "Ptpicneg" followed by "Normal." If this composition was found in the labeled series, then a condition check must be done by comparing the point values.

In this composition triggered on four successive points, the value of the second point must be greater than the last point, and the value of the third point must be greater than the first point. If this condition is true, an anomaly is triggered.

Label-composition 2

composition: Regular. Ptpicpos .Ptpicneg. Normal

condition: v2 < v4 and v3 < v1

conclusion: negative peak -> v3

Label-composition 2 detects a negative peak anomaly. However, as is already indicated, the exact composition could detect different types of anomalies, and the condition makes it possible to apply one of the two. In this example, the "composition" parts of **Label-composition 2** and **Label-composition 1** are identical. However, the "condition" and the "conclusion" are different. Thus, during the evaluation, ARP first checks the compositions. Then it checks their conditions to see which condition is true to identify the corresponding anomaly.

Label-composition 3

composition: Startcstpos . Cst* . Endcstpos

condition: v1 == v2 and vn-1== vn

conclusion: constant -> all

In this example, "Startcstpos" means the start of a constant, which is followed by zero or several "Cst*" constants, which is followed by the end of the constant "Endcstpos." The anomaly is the set of points of the composition.

Label-composition 4

composition: Regular. Ptpicpos . Ptpicneg AND Changnivneg . Normal

condition: v2 > v4 and v3 > v1

conclusion: positive peak -> v2

In this composition, there is a point "Normal" followed by "Ptpicpos" followed by a point which has two labels at the same time "Ptpicneg AND Changniv" followed by "Normal."

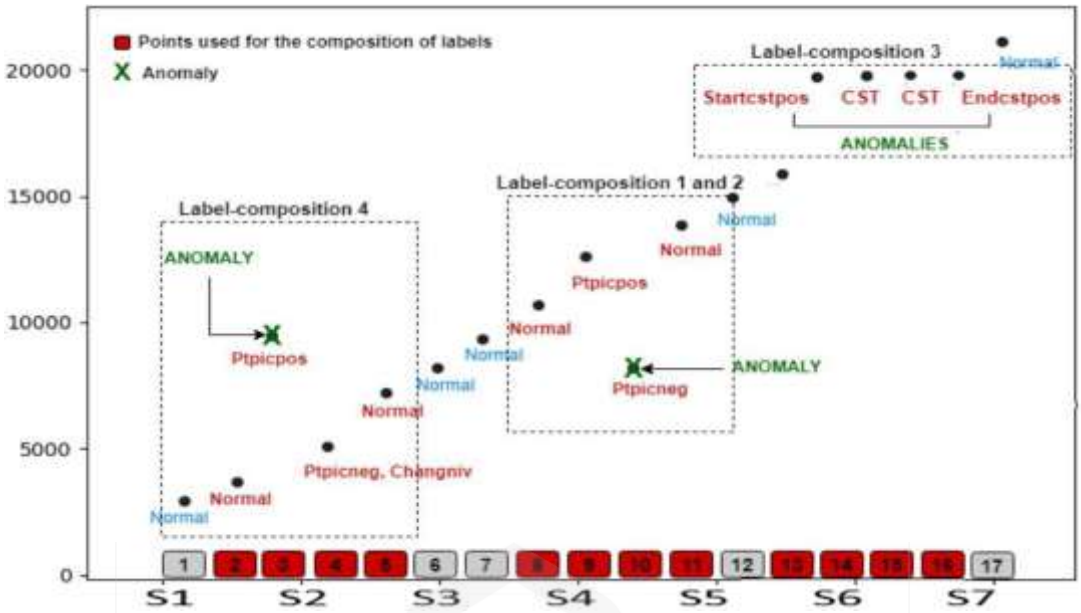


Figure 4. 15 ARP algorithm Phase 2 Results

Considering the points subsets as shows in Figure 4.15 in red. For example, points 2 to 5 in the index, give the following sequence of labels: (Normal . Ptpicpos . Ptpicneg and Changniv. Normal), which is detected by **Label-composition 4**. This composition, therefore, makes it possible to detect the positive peak anomaly at index 3 in Figure4.15.

Index points 8 to 11 trigger **Label-composition 1** and **Label-composition 2** (the same composition with different conditions and conclusions). By checking **Label composition 1**, the condition v9 > v11 and v10 > v8 is false. Therefore, **Label composition 1** is invalid. For **Label- composition 2**, the condition v9 < v11 and v10 < v8 is true. Therefore, the composition is valid, and the Peak Negative anomaly is recognized in v10. Index points 12 to 16 trigger **Label-composition 3** to detect a constant plate.

4.4.1.5. Application to Local Flow Data

An algorithm had been implemented capable of traversing a list labeled TsL and checking, from each point, which compositions of labels apply to identify the anomalies (and the corresponding points).

In Table 4.5, ARP was applied on the local network. The objective was to identify the anomalies in the daily and historical data of the meters in order to facilitate

the analysis. Using ARP could locate the location of different types of anomalies and also pinpoint the types of anomalies as shown in Figure 4.15.

Table 4. 5 Application of ARP on the local network

A	B	C	D	E	F	G
SizeOfCode	CheckSum	Timing	SizeOfImage	SizeOfInitializedData	Malware	Remarkable Points Detection
113152	2018335512	1566779476	483328	189440	1	constant
73728	0	1607501316	98304	139264	1	level shift
28672	409729	1604736516	393216	20480	1	noise
40448	2598754130	1630829316	81920	28160	1	peak
53760	375001	1631434116	458752	89600	1	noise
1667584	2380904	1631866116	2387968	15360	1	level shift
47616	2485637	1623917316	98304	282112	1	peak
12288	0	1576656516	98304	13824	1	constant
37376	51072	1577261316	69632	162816	0	begnine
16384	28252	1574496516	36864	318976	0	begnine
1183744	18898230	1603267716	18882560	20992	0	begnine
0	0	1608538116	278528	610304	0	begnine
85504	370840	1596010116	376832	841728	0	begnine
531456	610969	1595837316	606208	46080	0	begnine

4.4.1.6. Summary of the first Contribution: ARP

This section presented the contribution to “detecting anomalies” in univariate time series by modeling patterns and compositions. The approach aims to identify different types of outliers observed during real deployments. The targeted objective is to maximize the number of anomalies detected and to minimize false alerts and false positives.

Thus, this proposal, ARP, relies on pattern-based anomaly detection. First, it consists in labeling all the remarkable points which exhibit unusual behavior using patterns. Then, by label compositions, it precisely identifies the multiple anomalies present in the univariate time series.

This approach requires domain expertise to be able to define patterns effectively. Indeed, the determination of the thresholds σ_a and σ_b requires an exploration of the data in order to understand the behavior of the remarkable points which might exist. Also, the composition of labels comes from the knowledge and feedback of experts who tend to analyze the neighborhood of remarkable points to detect anomalies. Unlike the

methods in the literature presented in Table 4.1, ARP requires business expertise to be applied. Consequently, this approach may not be completely adapted to the expectations of users who prefer a more intuitive approach. However, it meets their primary need by precisely detecting multiple anomalies. Moreover, an improvement path could be based on the automatic learning of the patterns to learn the thresholds or even to learn the compositions of labels.

In conclusion, this approach offered several advantages. It could detect different anomalies types observed during real deployments. The approach also made it possible to identify the location of outliers precisely. It gave information on the type of anomaly found in univariate time series and was reliable for detecting multiple anomalies.

4.4.1.7. Recap of the Section

ARP method was composed of two steps: it marks (labels) all the remarkable points present in the time series based on detection patterns, then precisely identifies the multiple anomalies present from label compositions. This approach requires application domain expertise to define label patterns and compositions effectively. Although it requires good business expertise to be applied, ARP has the advantage of being precise in detecting different types of anomalies and locating the points where the anomaly generates few errors, as will show in part III during evaluations. This work has been published in several journal papers indexed in the Annex.

The following part presents the second contribution, which automates the labeling process and automatically generates classification rules.

4.4.2. Detection Methodology based on Composition-based Decision Tree

Human interpretable rules for anomaly detection refer to anomalous data presented in a format (rules) that is intelligible to analysts. Learning these rules is complex, and only a few works address the issue of different types of anomalies in time series.

In the previous section, the proposed method solves the challenge of detecting multiple anomalies by using patterns to label time series and pattern compositions to identify anomalies.

This section seeks to solve another issue: the automatic production of rules for detecting anomalies to help the experts decide by keeping the main idea of patterns and compositions. The main goal is to make labeling automatic and learn pattern compositions to generate rules.

The pattern-based Composition-based Decision Tree (CDT) method had been described to attain this, automatically generating a reduced set of human-understandable rules. Thus, it proposed automatic labeling of time series through patterns. Then a decision tree was built based on the pattern positions. In addition, it used Bayesian optimization to avoid manual tuning of hyper-parameters. Then, a quality measure was defined to assess the produced rules' correctness and intelligibility.

4.4.2.1. Context and Motivation

In a real context, experts analyze the anomalies they detect in the curves and manually construct decision rules to detect future occurrences of these anomalies. However, as the amount of data collected increases, the decision rules become more complex to define, which makes analysis more difficult.

The automatic extraction of rules and the timely detection of these different outliers can be of considerable interest to an expert. To overcome this challenge, rule learning algorithms had been proposed (Barakat and Diederich, 2005; Singh and Gupta, 2014). The deployment of such systems could reveal comprehensible information to users to explain the root cause of anomalies better than black box-type algorithms.

Another critical challenge is the presence of different types of abnormalities that can occur in time series. Furthermore, these differences may depend on the domain of application (Chandola et al., 2009; Aggarwal, 2015). Thus, a complex problem in anomaly detection approaches is considering the diversity of all existing anomalies, as explained in the introduction chapter.

To address these challenges, a machine learning method had proposed to generate human-interpretable rules for anomaly detection in univariate time series, called Composition-based Decision Tree (CDT).

This method uses sequences of patterns (patterns) to identify conspicuous points corresponding to multiple anomalies. Compositions (sequences) of patterns existing in time series are learned through an internally generated decision tree, then simplified using Boolean algebra to produce intelligible rules. Bayesian hyper-parameter optimization (two hyper-parameters) was used to obtain the best hyper-parameters for the method.

The approach aims to find the best compromise between high precision for anomaly detection and a minimized set of rules more easily interpretable by humans.

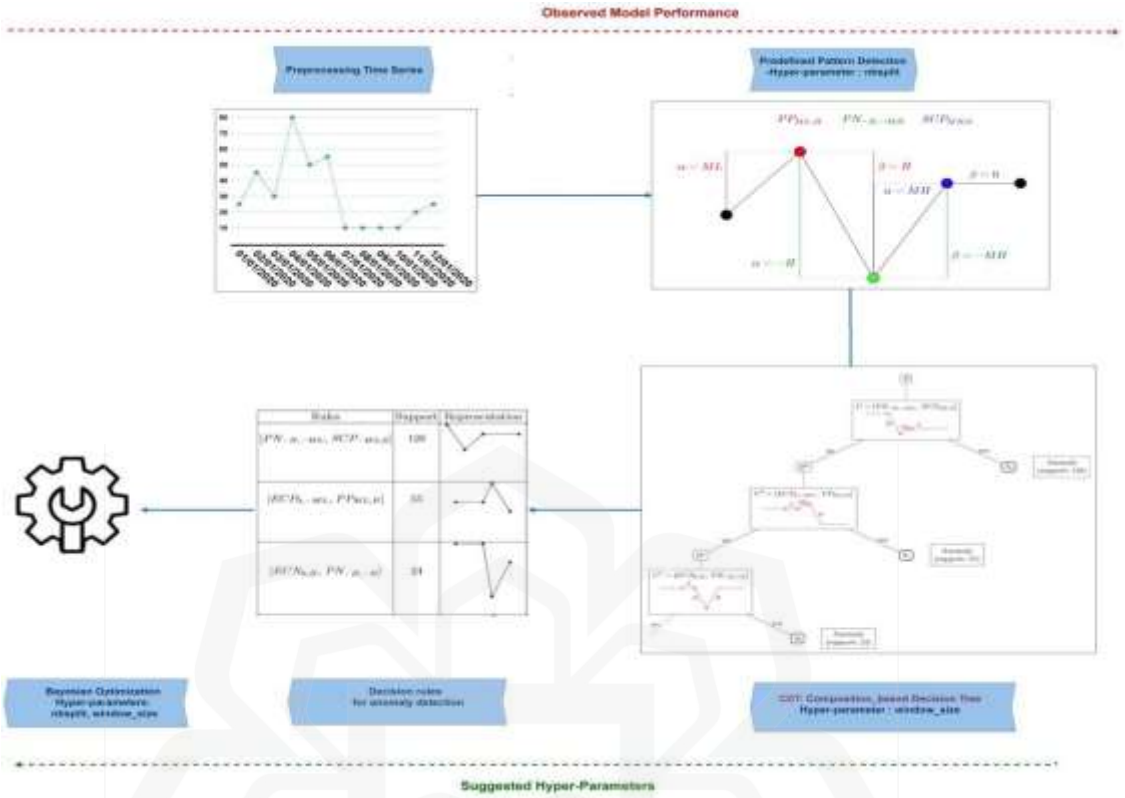
4.4.2.2. Time Series Preprocessing

The univariate time series had been preprocessed, as shown in below Figure 4.16. The time series was collected from different sensors, and the values of the measurements were over ranging different. To obtain scale and offset invariance and to ensure better model stability, a normalization for each continuous time series T_s to values between 0 and 1 had been done. The data thus obtained were normalized using the following equation:

$$Norm(xi) = \frac{x_i - \min(Ts)}{\max(Ts) - \min(T_s)} \qquad (4.4)$$

Such that x_i is a measure (point value) of a time series T_s.

Figure 4. 16 The different stages of the CDT approach.



Resampling could also be used in the analysis and exploration of time series. Downsampling is the process of reducing the data sample rate. Indeed, it consists in manufacturing a series comprising fewer samples than an original signal.

For example, suppose that a network host sends data to a supervision system every second; the expert may need the data according to an hour or a week instead of seconds. This facilitates the analysis by limiting the number of points x_i. Using down sampling, multiple data points in a time range for a single time series are aggregated using a mathematical function, such as the mean, into a single value at an aligned timestamp.

4.4.2.3. Time Series Labeling

In this section, the typical variations between successive points to formalize patterns had been sought to use, then used to label remarkable points in time series. Labeling with the patterns makes it possible to subsequently generate intelligible and more readable rules by the experts.

Consider three successive time series points: T_s denoted x_i-1, x_i, x_i+1. Three successive points make it possible to define nine possible variations as listed in Table 4.6, namely, PP (Positive peak), PN (Negative peak), SCP (Start Constant Positive), SCN (Start Constant Negative), ECP (End Constant Positive), ECN (End Constant Negative), CST (Constant), VP (Positive Variation) and VN (Negative Variation).

Given a normalized time series T_s (values between 0 and 1), it assumed that each of these variations could have different magnitudes between [-1,1].

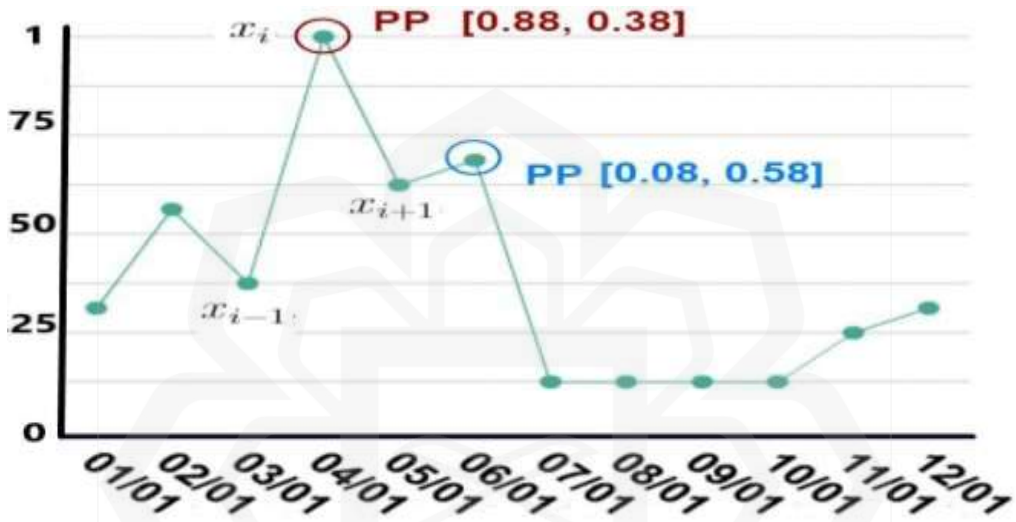


Figure 4. 17 Example of Positive Peak (PP) type variation.[3]

In order to refine the detection efficiency of this approach, the distinguishing had been used at several intervals in [-1,1] for each variation. For example, Figure 4.17 illustrates a PP variation with different magnitudes [0.88, 0.38] and [0.08, 0.58]. The values [0.88, 0.38] and [0.08, 0.58] were the variation of point x_i with respect to x_i-1 and x_i with respect to x_i-1. So, notice that the magnitude of the first variation is larger than the second variation.

Hyper-parameter (δ). As noted, δ the hyper-parameter used to distinguish the different quantities of the nine variations (PP, PN, SCP, SCN, ECP ECN, CP, VN, and CST).

[3] Example of Positive Peak (PP) type variation. There are two models labeled PP[0.88,0.38] and PP[0.08,0.58] corresponding to multiple positive peaks but with different amplitudes.

For a given δ, construct intervals had has $2\delta + 1$ (taking their limits between [-1, 1]) describing the extent of variation of the points: δ intervals for positive variations (in the interval]0, 1]), δ intervals for negative variations (in the interval [-1, 0]), and a particular case for the absence of variation (equal to 0).

This hyper-parameter allows having patterns of more or less acceptable amplitudes to capture any change of points in the series. δ will be determined automatically using Bayesian optimization.

Definition: A pattern had defined noted P = (l, α, β) where l is a name (or a label) identifying the pattern, and α and β are two possible intervals of [-1,1].

For each successive point xi-1, x_i, x_i+1, the point x_i is labeled by a pattern only if $x_i - x_i\text{-}1 \in \alpha \wedge xi - x_i\text{+}1 \in \beta$. In this case, x_i is labeled with l.

For example, with δ = 2, 5 intervals had constructed: δ]0,0.5], δ]0,5,1] , δ[-0,5,0[, δ[-1,-0.50[and δ[0,0]. For simplicity, the rest of the chapter could be only considered the notation with δ = 2 (other values of δ would only result in a greater variety of intervals and patterns), and each interval had denoted as follows:

- — δ]0.0.5]: Low (L =]0,0.5]),
- — δ]0.5,1]: High (H =]0.5,1]),
- — δ[-0.5,0[: -Low (-L = [-0.5,0[),
- — δ[-1,-0.50[: -High (-H = [-1,-0.5[),
- — δ[0,0]: special case, Zero (Z = 0).

Using these five intervals (for δ = 2) and the nine variations summarized in Table 4.6, 23 possible patterns could get as follows:

- For the PP variation, four patterns were there: PPL,H, PPH,H, PPH,L, PPL,L. The same principle was followed for the PN variation

Table 4. 6 Types of variations for labeling.

Variation	Définition	Motif	Exemple
PP	$x_{i-1} < x_i \wedge x_i > x_{i+1}$ $\alpha, \beta \in \{L, H\}$	$PP_{\alpha,\beta}$	$PP_{L,H}$
PN	$x_{i-1} > x_i \wedge x_i < x_{i+1}$ $\alpha, \beta \in \{-L, -H\}$	$PN_{\alpha,\beta}$	$PN_{-H,-H}$
SCN	$x_{i-1} > x_i \wedge x_i = x_{i+1}$ $\beta \in \{Z\}$ $\alpha \in \{-L, -H\}$	$SCN_{\alpha,\beta}$	$SCN_{-L,0}$
SCP	$x_{i-1} < x_i \wedge x_i = x_{i+1}$ $\beta \in \{Z\}$ $\alpha \in \{L, H\}$	$SCP_{\alpha,\beta}$	$SCP_{H,0}$
ECN	$x_{i-1} = x_i \wedge x_i > x_{i+1}$ $\alpha \in \{Z\}$ $\beta \in \{L, H\}$	$ECN_{\alpha,\beta}$	$ECN_{0,L}$
ECP	$x_{i-1} = x_i \wedge x_i < x_{i+1}$ $\alpha \in \{Z\}$ $\beta \in \{-L, -H\}$	$ECP_{\alpha,\beta}$	$ECP_{0,-L}$
CST	$x_{i-1} = x_i \wedge x_i = x_{i+1}$ $\alpha, \beta \in \{Z\}$	$CST_{\alpha,\beta}$	$CST_{0,0}$
VP	$x_{i-1} < x_i \wedge x_i < x_{i+1}$ $\alpha \in \{L, H\}$ $\beta \in \{-L, -H\}$	$VP_{\alpha,\beta}$	$VP_{L,-L}$
VN	$x_{i-1} > x_i \wedge x_i > x_{i+1}$ $\alpha \in \{-L, -H\}$ $\beta \in \{L, H\}$	$VN_{\alpha,\beta}$	$VN_{-L,L}$

- For the VP variation, only three patterns were obtained: $VP_{L,H}$, $VP_{H,L}$, $VP_{L,L}$.
- The $VP_{H,-H}$ pattern, could not be applied. Indeed, $VP_{H,-H}$ corresponds to VP]0.5,1],[-1,-0.5[. Since the values 0.5 and -0.5 were excluded in the H and -H intervals, any combination of values from these intervals would exceed the range of actual values of the time series between 0 and 1. The same principle was followed for the VN variation.
- The SCN variation had three possible patterns: $SCN_{-L,0,}$ and $SCN_{-H,0}$. The same principle was followed for SCP, ECN, and ECN;
- For the CST variation, it had a single pattern $CST_{0,0}$. Using these intervals could create, for example, a $PP_{L,H}$ pattern where PP was a positive peak with α =] 0,0.5] (labeled L) and β =]0.5,1] (labeled H). Conversely, it could define a $PN_{-L,-H}$ pattern in the negative peak case.

Figure 4.18 illustrates remarkable points represented by different patterns named $PP_{L,H}$, $PN_{-H,-H}$, $SCN_{H,0}$.

- $PP_{L,H}$ is a positive peak such that: PP presents the variation α = L, α =]0, 0.5] (marked L) and β =]0.5, 1] (marked H).
- $PN_{-H,-H}$ is a positive peak such that: α = -H and β = -H ;
- $SCP_{H,0}$ is the beginning of a positive constant such that α = H and β = 0.

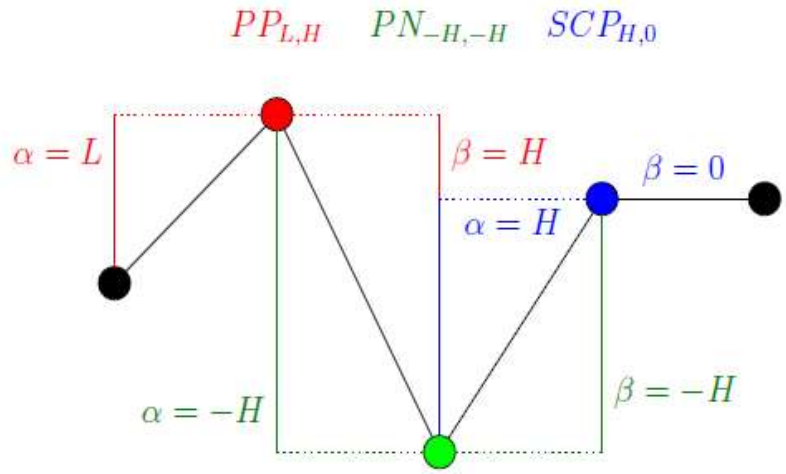


Figure 4. 18 Example of different patterns

Figure 4.19 shows a time series of building heat meter consumption data. It shows examples of different pattern sizes such as $PP_{L,H}$, $PP_{L,L}$ and $PP_{H,H}$. Using these templates could automatically label each noteworthy point in the series.

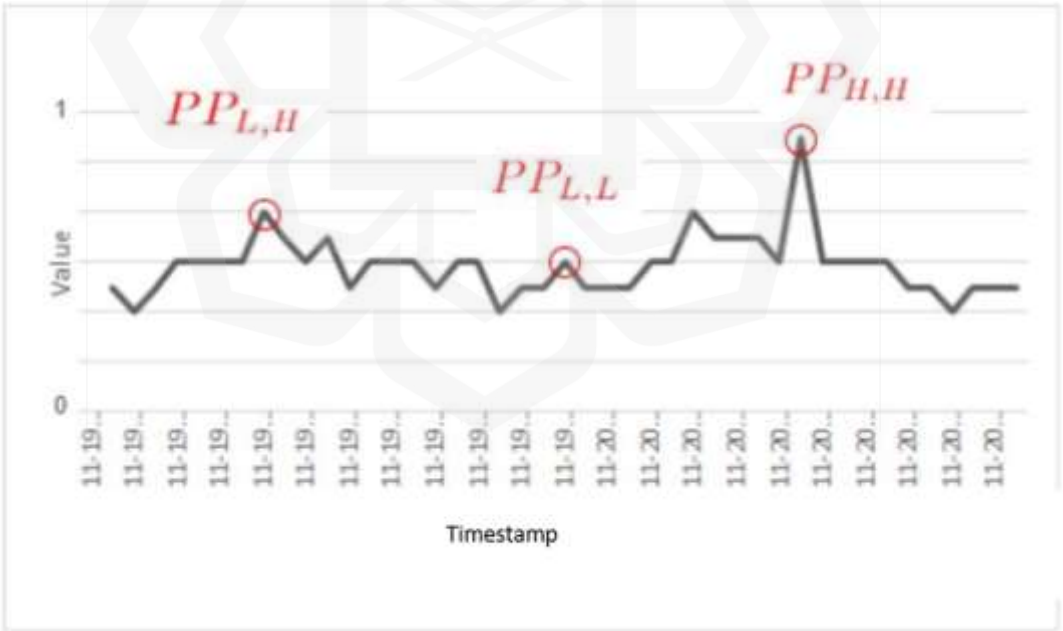


Figure 4. 19 Example of the pattern.[4]

[4] Example of the pattern. There are three patterns marked $PP_{L,H}$, $PP_{L,L}$ and $PP_{H,H}$ corresponding to positive peaks but with different amplitudes.

Definition A labeled time series denoted by T_{sb} = {l1, ..., lN } with N = n-2 where the label of its corresponding pattern replaces each point x_i of an initial time series (Ts).

Note that for this section, an adaption of notations with L,H,-H,-L had made in the general case where δ = 2, and the labels were denoted with explicit intervals. For example, if δ = 4, then for PP would be the following intervals]0,0.25], [0.25,0.50],]0.50,0.75],]0.75,1].

4.4.3. Composition-based Decision Tree

This section describes the Composition-based Decision Tree (CDT) method for rule mining and anomaly detection. First, the time series has been pre-processed by performing value normalization and possibly resampling. Then, it formalizes nine anomalous variations corresponding to the different types of anomalies in the form of patterns. Finally, predefined patterns are used to create labeled time series automatically.

Subsequently, a modified decision tree is created to build an anomaly detection classifier and generate decision rules. It simplified the produced rules, evaluated the quality of the rules, and adjusted the hyper-parameters using Bayesian optimization. A description of each step in detail in the following sections, as illustrated in Figure 4.16.

4.4.3.1. Decision Tree Weaknesses

Though, decision trees are considered to produce comprehensible rules, the case is not here, and they are not fully modified to the requirements:

- The rules can become unintelligible if the trees are tall. A large number of rules thus obtained makes them complex to interpret. Also, some rules may not make sense to the user.
- When dividing the dataset, the decision tree considers variables (features) without any order. Experts, on the other hand, always think in successive points. Thus, an adept of the principle of decision trees to take into account the order between variables. Rules-based had been built based on compositions of ordered variables similar to the grammar proposed in the first contribution but in an automated way. To solve the problems mentioned above, an adapted version of decision trees is proposed to generate interpretable rules for anomaly detection.

4.4.3.2. Decision Trees Definition

Decision trees (DT) are used in data mining as a decision support tool. They employ a hierarchical representation of the data structure in the form of sequences of decisions (tests) for predicting a variable or class.

This supervised learning algorithm is well known for its simplicity and understandability. It is induced from learning observations composed of values of variables or characteristics and a class label (target variable).

A tree is built by dividing the training data into subsets by choosing the variable that best partitions the training data according to an evaluation criterion. This criterion characterizes the homogeneity of the subsets obtained by dividing the data set. Among these criteria, we can cite Shannon's entropy and Gini's diversity index. This process is repeated recursively on each derived subset until all (or almost all) instances of a subset are in the same class (Su and Zhang, 2006).

This construction principle is called "Top-Down," i.e., the tree is built from the root to the leaves following a greedy and recursive algorithmic approach. For example, in a decision tree, each internal node (or decision node) describes a test on a learning variable, each branch (or arc) represents a result of the test, and each leaf contains the value of the target variable (a label of class).

The decision tree defines a classifier that translates into terms of decision rules, mutually exclusive and ordered (in the form of if-then-else).

4.4.3.3. Description of Composition Decision Tree

In a classic tree, each node carries out a test relating to a variable whose result indicates the branch to follow in the tree. On the contrary, in this tree, each node performs a test relating to a composition of patterns (ordered sequence of remarkable points), thus representing a sequence of ordered variables.

This section describes the CDT method. Tree construction extends classical decision tree construction such as CART (Breiman et al., 1984) and C4.5 (Quinlan, 1993).

Furthermore, moving towards a classification in the form of binary trees makes it possible to classify two categories of classes (normal and anomaly). The input to the decision tree is in the form of sliding windows of fixed size created from the labeled time series.

Definition, A set of observations noted D = {d1, d2, ..., $d_{N-\omega+1}$} represents the result of the cut of T_{sb} by sliding window of size ω, and a fixed step of 1. Thus, the different observations are:

$$D = \{\{l1, ..., l\omega\}, \{l2, ..., l\omega+1\}, ..., \{l_{N-\omega+1}, ..., lN\}\}.$$

Let M be the number of classes with which the observations are associated. In this context, two classes had been considered (M = 2): the abnormal class (observation with anomaly) or the normal class (observation without anomaly). Each observation di ∈ D is associated with a single annotated class (di).

Hyper-parameter (ω). We note ω <= N/ 2, a window size such that N represents the size of the time series labeled T_{sb}. This hyper-parameter will be determined automatically using Bayesian optimization.

In order to determine the probability of distribution of the observations on the classes of a node, several functions of heterogeneity or impurity can be defined, such as the Gini index or entropy. The opt-in this method for the Gini index (Singh and Gupta, 2014) as a measure of impurity of a subset of observations Dj ⊆ D.

Definition: The Gini impurity index, denoted G(Dj), provides a measure of the quality of the set of observations Dj according to the distribution of the observations in the classes. The impurity metric is minimal (equal to 0) when all the observations belong to the same class, and maximal (equal to 0.5) if there are as many elements of each class. The Gini impurity index is defined as:

$$G(D_j) = \sum_{k=1}^{m} pk(1 - pk) \qquad (4.4)$$

where pk is the fraction of observations belonging to class k such that pk = |di ∈ Dj/ class(di) = k|/| Dj |, from observation, the composition had defined used as a characteristic to split a node into two subnodes.

Definition A composition denoted c is a sequence of existing labels in an observation di. Using the symbol ⊆o, we denote c ⊆o di.

Example. Consider d = {l1, l2, l3, l4, l5, l6}.

- c = {l2, l3, l4} ⊆o d,
- c = {l3, l2, l4} ⊈o d,
- c = {l1, l2, l3, l4, l5, l6} ⊆o d.

Also, an introduction of additional notations had as follows: c ∈o D when ∀d ∈ D, c ⊆o d, and c ⊈o D when ∀d ∈ D, c ⊈o d.

A decision tree is constructed based on the variables with the highest information gain. During the creation of the CDT, the compositions are compared according to the information gain they provide.

Definition an Information Gain, denoted IG, allows the measurement of the score quality of a subset of compositions and the amount of "information" that a composition gives about the class.

$$IG(D_j, c) = G(D_j) - (\frac{|D_{inc}|}{|D_j|}G(D_{inc}) + \frac{|D_{exc}|}{|D_j|}G(D_{exc})) \quad (4.5)$$

Where |Dinc|, |Dexc| and |Dj| are the size of Dinc, Dexc and Dj respectively. Dinc and Dexc are two subsets of Dj where Dinc = {d ∈ Dj |c ⊆o d} and Dexc = {d ∈ Dj |c ⊈o d}.

The CDT process is described by Algorithm 3. To build a decision tree, it defined a node of the tree as a quadruplet (as shown in line 1 of Algorithm 3):

- observations: the set of observations considered in this node;
- composition: used to divide observations into child nodes;
- childTrue: the node of observations satisfying the composition;
- childFalse: the node of observations that do not satisfy the composition.

A function list_of_all_possible_compositions() was introduced to compute all compositions deduced from Dj (line 6 in algorithm 3). For each composition, it calculated the information gained for splitting a node (lines 7-15).

On line 16, if G(Dj) 6= 0 means that the set of observations of the node is impure (the observations are of different classes). Moreover, maxGain = 0 means that a composition that divides the set of observations has already been found: in this case, it

created a node Ninc which represents the positive branch of the node (c ⊆o Dj), and Nexc, which represents the negative branch (c ⊈o Dj) (line 16–25). These steps have been repeated until there are no more nodes to process (lines 3–26).

Algorithm 3 CDT : Composition-based Decision Tree

Input : $D = \{d_1, d_2, ..., d_{N-\omega+1}\}$ a set of observations
Output : N_{root} the root node of CDT

1: $N_{root} \leftarrow Node(D, null, null, null)$
2: $q \leftarrow [N_{root}]$ *// construct the queue of nodes to split*
3: **while** $q \neq \emptyset$ **do**
4: $N_j \leftarrow q.pop()$ *// dequeue the first node from the queue*
5: $D_j \leftarrow N_j.observations$
6: $C_j \leftarrow list\_of\_all\_possible\_compositions(D_j)$
7: $maxGain \leftarrow 0$
8: $c_{best} \leftarrow null$
9: *// Choose the composition that has the best Gain*
10: **for all** $c \in C_j$ **do**
11: **if** $IG(D_j, c) > maxGain$ **then**
12: $maxGain \leftarrow IG(D_j, c)$
13: $c_{best} \leftarrow c$
14: **end if**
15: **end for**
16: **if** $G(D_j) \neq 0$ and $maxGain \neq 0$ **then**
17: $D_{inc} \leftarrow \{d \in D_j | c_{best} \subseteq_o d\}$
18: $D_{exc} \leftarrow \{d \in D_j | c_{best} \not\subseteq_o d\}$
19: $N_{inc} \leftarrow Node(D_{inc}, null, null, null)$
20: $N_{exc} \leftarrow Node(D_{exc}, null, null, null)$
21: $q.append(N_{inc})$ *// enqueue child nodes*
22: $q.append(N_{exc})$ *// enqueue child nodes*
23: $N_j.composition \leftarrow c_{best}$
24: $N_j.childTrue \leftarrow N_{inc}$
25: $N_j.childFalse \leftarrow N_{exc}$
26: **end if**
27: **end while**
28: **return** N_{root}

Figure 4. 20 Example of CDT output

Figure 4.20 illustrates an example of CDT output. This tree is an extract of the actual local data. The root node, D1, represents the set of learning observations used for constructing the tree. The leaves represent class labels, and the branches represent the conjunctions of compositions that lead to these class labels. In this Figure 4.20, the tree is composed of 3 partitions (splits), building a set of 3 leaves S = {S1, S2, S3}.

4.4.3.4. Generating Rules for Anomaly Detection

This step produces a system of classification rules from the tree built via the set of paths starting from the tree's root and arriving at each of the leaves. Thus, it converted the tree from CDT into a set of decision rules. Then, it considered only "pure leaves," leading to the anomaly class.

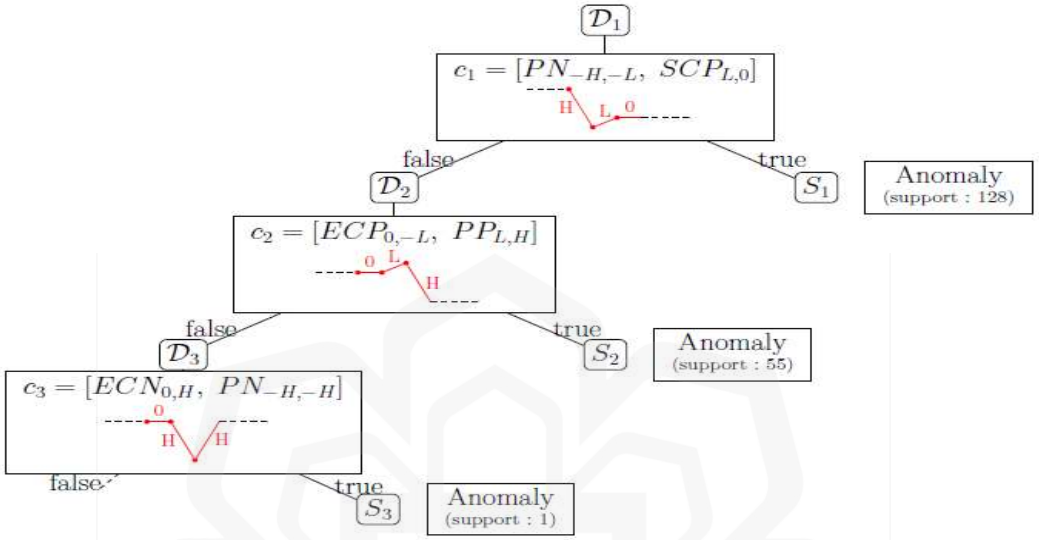


Figure 4. 21 Illustration of a tree obtained with CDT

Definition, a rule predicate, denoted Rs, is a branch of the decision tree or a path leading to the anomaly class. It is a conjunction ("logical AND") of tests encountered. It is built by combining (conjunction) the successive compositions ci or ¬ci from leaves to the root node. For each positive branch (ci $\in_o$ Dj), the positive composition ci is deduced while a negative composition ¬ci is deduced from a negative branch (ci $\notin_o$ Dj).

Example. Three rule predicates are produced from the tree of Figure 4.21.

- Rs1 : c1 = [PN-H,-L, SCPL,0]
- Rs2 : c2 ∧ ¬c1 = [ECP0,-L, PPL,H] ∧ ¬[PN-H,-L, SCPL,0]
- Rs3 : c3 ∧ ¬c2 ∧ ¬c1 = [ECN0,H, PN-H,-H] ∧ ¬[ECP0,-L, PPL,H] ∧ ¬[PN-H,-L, SCPL,0].

Abusive notations had been used, ci ∧ ¬cj, which means that for an observation d on a time series, a check ci $\subseteq_o$ d ∧ cj $\not\subseteq_o$ d. So, although incorrectly noted as logical expressions for simplicity, a predicate of the rule defines a list of compositions that an observation must or must not contain to be considered an anomaly.

Definition: A rule, denoted R, is a disjunction of rule predicates. For example, as shown in Figure 4.21: R = Rs1 ∨ Rs2 ∨ Rs3 = (c1) ∨ (c2 ∧¬c1) ∨ (c3 ∧ ¬c2 ∧ ¬c1).

4.4.3.5. Simplification of Rules

The objective of having rule predicates readable by the expert-led is to consider their length (number of compositions, number of labels). Simplifications envisaged aim to reduce the number of compositions composing the rule predicates.

This research sought to eliminate redundant compositions or did not seem good at predicting the class. This pruning makes it possible to eliminate specific tests from the conditional part of a rule predicate or an entire rule predicate. One way to minimize CDT rule predicates was to post-process the rule produced by simplifying the logical expression using the rules of Boolean algebra. Boole's algebra originated from British mathematician George Boole (Taylors, 1954). There are three basic operators:

- Not, denoted ¬a or $\underline{a}$, which inverts the value of the variable a;
- And, denoted a · b or a ∧ b which returns 1 if a and b are equal to 1, otherwise returns 0;
- Or, note a + b or a ∨ b which returns 1 if a or b is 1, otherwise returns 0

The function could be transformed logic from the Boolean algebra properties to simplify it (Taylor, 1954). Among these properties, the commutativity had been used for presented by the following equation:

$$a\ b = b\ a \text{ and } a + b = b + a \qquad (4.6)$$

Also, there are several laws of Boolean algebra, such as the lightning theorem, absorption theorem, Morgan's theorem, etc. The rules had been applied to the law relief in order to simplify them using the following rule:

$$a + a\ b = a + b \text{ and } (a + b)\ a = b\ a \qquad (4.7)$$

The logical expression can be written in the form "sum of products" and "product of sums." CDT's rules are in the form of the “sum of the products” of Boolean functions. Indeed, it is a disjunction of compositions.

So, using equations 3.5 and 3.4, can simplify the generated rule R from the tree of Figure 4.22 as follows:

$$R = Rs1 \vee Rs2 \vee Rs3$$
$$= (c1) \vee (c2 \wedge \neg c1) \vee (c3 \wedge \neg c2 \wedge \neg c1)$$
$$= (c1) \vee (\neg c1 \wedge c2) \vee (\neg c1 \wedge \neg c2 \wedge c3)$$
$$= (c1) \vee (c2) \vee (c3)$$

Example. Consider the example of the three rule predicates generated from *Figure4.22*.

$$\mathcal{R} = R_{S_1} \vee R_{S_2} \vee R_{S_3}$$
$$= [PN_{-H,-L}, SCP_{L,0}] \vee ([ECP_{0,-L}, PP_{L,H}] \wedge \neg[PN_{-H,-L}, SCP_{L,0}]) \vee$$
$$([ECN_{0,H}, PN_{-H,-H}] \wedge \neg[ECP_{0,-L}, PP_{L,H}] \wedge \neg[PN_{-H,-L}, SCP_{L,0}])$$
$$= [PN_{-H,-L}, SCP_{L,0}] \vee (\neg[PN_{-H,-L}, SCP_{L,0}] \wedge [ECP_{0,-L}, PP_{L,H}]) \vee$$
$$(\neg[PN_{-H,-L}, SCP_{L,0}] \wedge \neg[ECP_{0,-L}, PP_{L,H}] \wedge [ECN_{0,H}, PN_{-H,-H}])$$
$$= [PN_{-H,-L}, SCP_{L,0}] \vee [ECP_{0,-L}, PP_{L,H}] \vee [ECN_{0,H}, PN_{-H,-H}]$$

Figure 4. 22 Example of the three rule predicates

In this approach, this cross-industry simplification is applied until there is no more simplification to be done. This allows the minimization of the number of compositions in a rule.

4.4.3.6. Quality Measurement

In (Barakat and Diederich, 2005) the authors defined rule quality criteria to evaluate the rules extracted from the SVM algorithm, such as the rules comprehensibility and precision. Thus, the authors considered a set of rules accurate if it can correctly classify new examples. The comprehensibility of a ruleset was determined by measuring the size of the ruleset (in terms of the number of rules) and the number of antecedents per rule.

In (Daud and Corne, 2009), the authors studied the performance of classification algorithms from the literature, such as rule induction algorithms and decision trees. For their experimental results, they were interested in the percentage of correctly classified instances of the algorithms (percentage of precision) and in the readability which was calculated according to the number of rules or the size of the trees produced by the classifiers (number of node).

In this work, the aim to generate precise and understandable rules had been sought. An assumption is that a comprehensible rule should be short and contain a reduced number of different tags. For this reason, it had been defined the criteria following:

- I(c) to characterize the quality of a composition c as a function of its length and the number of patterns used;
- M(IRs) to characterize the rule predicate quality RS as a function of the number of compositions of RS and the quality of each of the compositions I(c);
- Q(R) to characterize the quality of a rule R according to the quality of the rule predicates and their support.

To calculate the interpretability of a composition, it had been using the following equation:

$$\mathcal{I}(c) = 1 - \frac{L_c.N_L}{\omega.MaxL} \qquad (4.8)$$

where Lc = |c| denotes the length of a composition c (number of labels composing c), NL is the number of unique labels used in a composition, ω is the maximum window size, and MaxL is the maximum number of labels.

Next, it calculated the average interpretability of a rule predicate (conjunction of compositions) using the following equation:

$$\mathcal{M}(\mathcal{I}_{R_S}) = \frac{1}{Nc}\sum_{k=1}^{Nc}\mathcal{I}(c_k) \qquad (4.9)$$

where Nc is the number of compositions in the rule predicate RS. The quality of the extracted rules is calculated as follows:

$$\mathcal{Q}(\mathcal{R}) = \frac{1}{S}\sum_{i=1}^{nb} S_{R_{S_i}}.\mathcal{M}(\mathcal{I}_{R_{S_i}}) \qquad (4.10)$$

where nb is the number of rule predicates in R, S_{RSi} is the support of rule predicate RSi (true positive), and S is the support of all rule predicates (true positive and true negative). Since a rule predicate with high support is more important, it multiplies the average interpretability of a rule predicate with its support.

4.4.3.7. Automatic Selection of Hyper-Parameters

As mentioned in the previous sections, CDT has two hyper-parameters: the number of divisions (δ) to specify the magnitudes of the patterns and the window length (ω), indicating the size of the CDT observations.

Manual tuning requires prior knowledge and is time-consuming, as it is a brute-force search (Snoek et al., 2012). Indeed, manual research tests sets of hyper-parameters defined by the user, who must use his knowledge to identify the parameters that will improve the desired result.

To overcome this problem, automatic search algorithms such as grid or random search have been proposed in the literature (Wu et al., 2019). The grid search is exhaustive because it calculates all possible combinations of values for the hyperparameters. Although it can give good results, its cost remains high.

Random search tries random combinations of values. It is more efficient than grid search but may not find the optimal set of hyperparameters.

In the grid and random search, the configurations were done randomly and blindly. The subsequent trial is independent of all previous attempts. On the other hand, the automatic adjustment of the hyper-parameters makes it possible to know the relation between the values of hyper-parameters and the performances of the model to create a more judicious choice for the following values hyper-parameters. The goal is to minimize the number of trials while finding a good optimum.

With this in mind, Bayesian optimization has been used (Wu et al., 2019; Xia et al., 2017) to find the best hyper-parameters for the model. The objective of optimizing hyperparameters in machine learning is to identify the most suitable parameters for a specific machine learning algorithm that can improve its performance as evaluated on a validation dataset. Indeed, it had sought to find a configuration (i.e. a set of parameters) that maximizes a performance metric or an objective function.

A Bayesian optimization is a a method for determining the optimal values of hyperparameters for a machine learning algorithm. It uses a probabilistic model, called a surrogate, to predict the performance of the algorithm based on past evaluations of the objective function. This approach allows for more efficient optimization by using previous results to guide the search for the best hyperparameters. This function is much easier to optimize than the objective function since the time spent selecting the hyper-parameters is less important than the time spent in the objective function (Shahriari et al., 2015).

The principle is to find the next set of hyper-parameters to evaluate the real objective function by selecting the hyper-parameters that perform best on the surrogate function. Thus, hyper-parameter optimization involves the following steps:

1. Define a research domain for hyper-parameters.
2. Define an objective function that takes the hyper-parameters as input and produces a score that we want to maximize (or minimize).
3. Construct a substitution probability model of the objective function.
4. Find the hyper-parameters that work best on the surrogate. A criterion, called a selection or acquisition function, to evaluate the hyper-parameters to be chosen next in the surrogate model.
5. Apply these hyper-parameters to the objective function.
6. Incorporate the latest results into the surrogate model for updating it.

There are several choices for the surrogate model, such as Gaussian Process, Random Forest Regressions, and Tree Parzen Estimators (TPE), as well as for the hyper-parameter selection function, such as Expected Improvement and Upper Confidence Bound (Snoek et al., 2012; Shahriari et al., 2015).

Hyper-parameter optimization comes in the form of the equation below (Snoek et al., 2012):

$$h^{\star} = \arg\max_{h \in H} F(h) \qquad (4.11)$$

Where F(h) represents an objective function to be maximized, h^{*} is the set of hyperparameters (δ, ω) to be optimized, and h can take any value in the search space H.

To optimize the trade-off between detection performance and good rule quality, the objective function F(h) was defined as the F-measure (detection performance) weighted by the rule quality measure Q(R).

$$F(h) = F1(h).Q(R) \qquad (4.12)$$

Where F1(h) is the F-measure (the harmonic mean of precision and recall) of the classification performance obtained with the set of parameters (h).

This work had used to calculate the substitution function, the Gaussian process. They allowed generating for each point a distribution probability characterized by a mean (the most probable value) and a standard deviation (the measure of the probable dispersion of the value around the mean). For the pick feature, it useed Upper

Confidence Bound. This function associates each point of the search space with a potential to be optimal. Therefore, finding a new point to evaluate implies evaluating the selection function in the entire search space. However, these evaluations could be much less expensive in comparison to the evaluation of the objective function.

4.4.3.8. Synthesis of the Second Practical Approach

This section presents a machine learning method called Composition-based Decision Tree (CDT) for anomaly detection. It generates human interpretable rules based on a formalization of nine variations allowing to define patterns to label time series automatically and detect remarkable points. This labeling increases the readability of the rules.

Given this time series labeling which increases the readability of the rules, a decision tree is then constructed, considering the nodes as compositions of patterns with the highest information gain.

The input to the CDT is built by creating sliding windows of fixed size, where the split and window length hyperparameters are automatically calculated via Bayesian optimization. The tree is then converted into a set of decision rules, for which a quality measure is defined. This is based on the interpretability of the composition, which considers the length of the ruler and its number of labels.

In summary, this approach offers the benefit of producing clear and understandable rules for detecting anomalies with high performance. These rules can be easily reviewed and understood by experts, who can then make decisions or adjust the rules as necessary using their own domain knowledge.

4.4.3.9. Recap of the Section

The proposed CDT method is a machine learning method that generates expertly understandable rules for detecting multiple anomalies in univariate time series. The approach is based on a tree of amended decisions. In addition, Optimized the hyperparameters used Bayesian optimization had been there to maximize the rule quality and performance. Besides, this work was published in several papers within international journals and conferences such as ITSIS'2020 (Al-Aamri., 2020), which are indexed in the Appendix.

4.5. APT DETECTION MODEL RESULTS AGAINST ML ALGORITHMS

4.5.1. Effective Experimentation

After having described the “ARP” approach in the methodology, let's experience it in this section in competition with different anomaly detection methods. These experiments were carried out on the real data of the local network and the sets of data referenced from the literature presented in the methodology.

4.5.2. Methodology and experimentation

This section provides an explanation of the methods that were used during the experiment, the protocol approach of the experiment, then presents the results obtained from the local environment monitored data of the case study and from the literature data.

4.5.2.1. Exploration of Existing Methods of Detection

This study explored five methods belonging to four techniques for detecting the anomalies said APTs.

- Rule-based method: it used two rules to detect the short abnormalities (abnormal change) and constant abnormalities (no variation) Sharma et al. (2010). The short anomaly rule deals with time series by comparing each time two successive observations: an anomaly is detected if the difference between these observations is more significant than a given threshold. This approach was based on Ramanathan et al. (2006) to determine the detection threshold. The Constant Anomaly Rule calculates the standard deviation for successive observations. If this value equals zero, all is declared an anomaly.
- A method based on the density: this approach compares the density around one point based on the thickness of its local neighbors: Breunig and para. (2000) proposed the LOF algorithm. In this method, the anomaly scores are measured using an aberrant value factor, which is the report between the local density around this point and the local density around its nearest neighbors. The point whose value LOF is high is declared as an anomaly.
- A method based on statistics: first, it used the technique S-H-ESD, which uses the decomposition of time series STL (seasonal decomposition and trend using Loess) developed by Cleveland et al. (1990) for dividing the signal of chronological series into three parts: seasonal, tendency and residue. From here,

the techniques of detecting residual anomalies are then applied to the algorithm ESD using statistics metrics. Secondly, the Exchange Point method had used to detect the level change.

- A method based on the analysis of time series: the principle of this approach is to use the model's temporal correlations and predict the time series values. It used the model Auto Regressive Integrated Moving average (ARIMA) to create a prediction model according to the approach described by Chen and Liu (1993). Then, a manual value is compared to its predicted value to determine if it is an anomaly.

There are open-source implementations for algorithms such as LOF, ARIMA, S-H-ESD, and Change Point ((Hochenbaum et al., 2017), (López-de-Lacalle, 2016), (Rosner, 1983), (Aminikhan- ghahi and Hori, 2017)) that they used for the experiments. But, on the other hand, they had implemented different approaches (short and constant rules) depending on the available sources.

Table 4.7 represents the synthesis of the methods explored to detect each kind anomalies presented in Table 4.8. However, as indicated in the introduction chapter, the algorithms cannot detect different types of APTs. For this reason, we evaluated them according to the categories of anomalies they could detect.

Table 4. 7 Comparison Between the anomalies observed in the real deployments and the anomalies detected by the literature algorithms.

Reference\Anomalies	Peaks	Noise	Plate	Level Change
Al-khatib et al., 2020	AO	TC	-	LS, SLS
Sharma et al., 2020	AO	TC	-	LS, SLS
Jaganathan et al., 2019	-	short duration	long duration	-
Owido et al., 2013	-	short duration	long duration	-
Yeh le. 2016	-	discords	-	-
Chen el. Z. 2008	-	collectives	-	-
Fermans la. 2019	-	contextual	-	-
Adams et al., 2019	-	TC	-	-
Rosner(1983)	global	local	-	-
Luo et al., 2021	-	TC	-	-

Table 4. 8 The methods of detection of anomalies studied.

Kind Methods _	of detection
Peaks	Ruler short, ARIMA, LOF, SH-ESD
ARIMA noise,	LOF, SH-ESD
Tray	Ruler constant
Change of ARIMA level,	Exchange Point

4.5.2.2. Experimental Approach

Figure 4.23 illustrates the approach to be followed in applying the algorithm ARP to the literature data. Thus, the evaluation methodology was composed of two steps. The first step consists in exploring and analyzing, with the help of experts, the remarkable points and their neighborhoods in an example of a time series. The second step consists of applying ARP to the data of the test to label the remarkable points (RP) using the patterns, then identifying anomalies from the compositions. Then, depending on the detected points, they evaluate the results using three metrics: the reminder, the accuracy, and the f-measure.

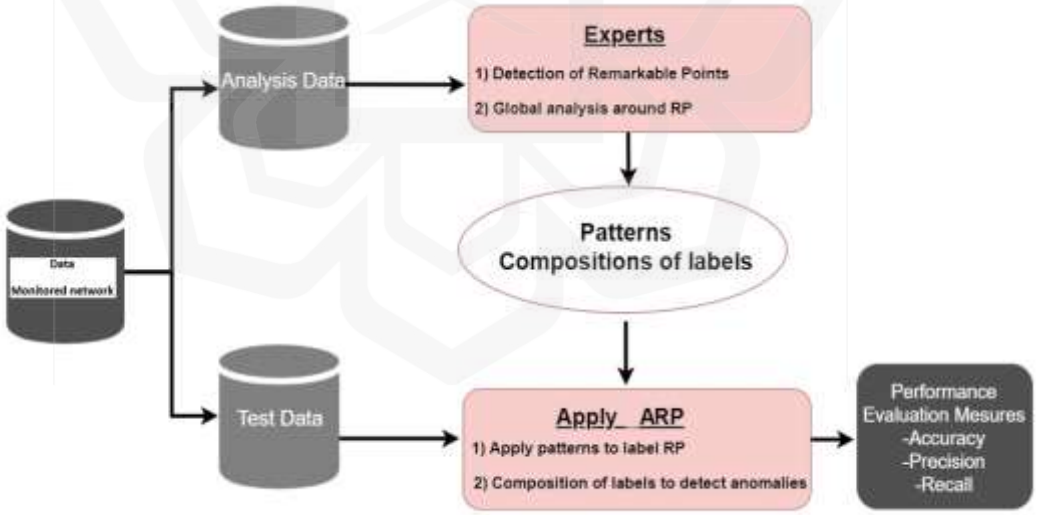


Figure 4. 23 Process evaluation of the algorithm ARP on the literature data

The experimentations were realized on a network monitor, a machine with Windows 10 professional, an Intel (Core) i5 processor, and 16 GB of RAM. They used the open distribution source Python 3.7 Anaconda to develop the algorithm, Matlab to test the tests and results visually, and the dedicated analysis of the confusion matrix.

4.5.2.3. Literature Data Experimentation (Incremental Series and Variable Series)

In this part, an evaluation of the following methods has been presented: Short Rule (denoted SR), Constant Rule (denoted CR), LOF, ARIMA, S-H-ESD, and Level Shift (denoted LS). It applied these methods by category of anomalies, as shown in Table 4.8. In order to evaluate their performance, it uses, first, the number of true positives (true anomalies detected), the number of false positives (false anomalies detected), and the number of false negatives (true anomalies non-detected). And we used the measures proposed in the methodology chapter section "Evaluation Metrics" (reminder, accuracy, F-measure).

As the methods to be evaluated are not fully automated, they had defined the values of their parameters, such as the threshold for the Short Rule, the number of neighbors, and the threshold to assess the score of the APT degree for LOF where the model type is ARIMA, etc. For the algorithm LOF, the choice of the parameter K had been enumerated, and the number of neighbors was in a range of 30 to 10 in order to evaluate its influence on the detection result (see the top of Figure 4.24B), and it had determined a threshold = 1.5 which corresponds to a standard distribution. LOF presented results in a separate graph for readability, as in Figure 4.24B.

For ARIMA, it kept the default values of the parameters (types of anomalies, model ARIMA) defined in the package.

Regarding the Short Rule (SR), it defined the threshold value using the histogram-based method described in the "rule-based detection methods" methodology. Finally, for the constant rule (CR), the choice enumerated the cut of the window sliding in a range of 30 to 10, as seen in Figure 4.24D.

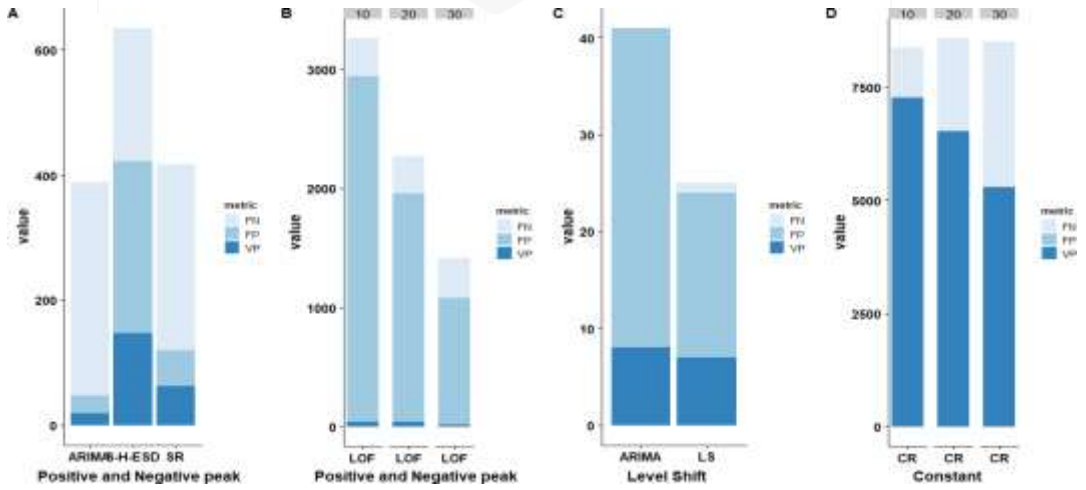


Figure 4. 24 Evaluation of methods for anomalies detection on the data of literature

Based on the results presented in the subfigures 4.24A, 4.24B, 4.24C, and 4.24D, which concern literature index data, thus, the following observations can be made: LOF is the method that generates the most prominent number of false positives. In contrast, ARIMA generates the most false negatives. The S-H-ESD method allowed the detection of more true positives by reporting to LOF, ARIMA, and SR. On the other hand, it generates many false positives and false negatives. Short Rule (SR) detects fewer abnormalities in comparison with S-H-ESD. However, the results are fewer false positives than other methods. Also, results indicated that the number of neighbors equal to 20 was the most appropriate choice to detect the enormous number of anomalies with the algorithm LOF. For the Constant Rule (CR), it was important to use a small cutoff window to handle data containing multiple constant anomalies, i.e., 10, as noted on 4.24D.

Table 4. 9 Comparison methods of the APT anomalies' detection on the literature data

	Literature Data			Local Data		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
SR	0.52	0.17	0.32	0.66	0.63	0.64
CR	1	0.8	0.88	1	0.72	0.83
LOF	0.022	0.12	0.022	0.39	0.78	0.52
S-H-ESD	0.34	0.4	0.36	0.41	0.8	0.54
ARIMA	0.3	0.07	0.11	0.66	0.25	0.36
LS	0.29	0.87	0.43	-	-	-
ARP	**0.988**	**0.988**	**0.987**	**0.94**	**0.8**	**0.83**

Table 4.9 presents the results of these methods according to precision, recall, and F-measure on the index data (incremental series) and on the literature data (variable series). Results also show the estimates of the proposed ARP method. ARP method yielded an excellent estimate with a recall of .98 and the same for the F-measure. As for the anomalies' detection, several algorithms were implemented (e.g., SR, CR, LOF, S-H-ESD, ARIMA, LS, ARP). From Table 4.9, we can see that the proposed ARP model outperformed other algorithms in terms of all evaluation methods, including precision, recall, and F-measure. This finding supports the proposed model in that the ARP model

produces a higher level of accuracy in predicting anomalies compared to other models in the context of this study.

The results in Table 4.9 suggest that the performance of the Constant Rule (CR) or the LOF method is heavily influenced by the selection of the sliding window or the number of neighbors. The Level Shift method is effective in detecting changes in the level of the time series, but its accuracy is poor in the absence of anomalies. Among the Short Rule (SR), S-H-ESD, and ARIMA, the Short Rule (SR) is the most accurate, and ARIMA is the least effective in detecting abnormal changes. The ARP algorithm, on the other hand, is able to detect multiple anomalies with better accuracy and better recall than other algorithms, and it can detect various anomalies that other algorithms are unable to detect.

The methodology used the local data from the pre-described dataset to evaluate the algorithm further and compare it to anomaly detection methods. According to the datasets, the data had been taken from network flow, from the daily evolution, unlike the literature data, which is variable. For ARP, a manual inspection of data of the same kind, a one-time series as illustrated in Figure 4.24-A (the phase from experts), to understand their changes. This way allowed the creation of the detection patterns of the remarkable points that may exist and the compositions of labels to detect the anomalies. The anomalies observed in those data were the following: peaks positive and negatives, constants anomalies, and constants anomalies starting and ending by a critical gap. Thereby, still with ugly experts, they had created nine patterns to detect remarkable points and five compositions of labels to detect the anomalies.

The results reported from algorithms on the local data are in Table 4.9 (variable series). Since the data was not stationary, the algorithm did not apply Level Shift, and thus, change levels did not exist in this data to be detected. This table shows that the algorithms were much more efficient on the local data compared with the literature data results. But even on this type of data, this approach obtained the best result of F-measure by comparing with the other algorithms. In effect, ARP detected more anomalies with fewer errors possible, with a precision equal to 0.988 and recall equal to 0.988. Note that the method results in a base of rules (SR, CR), and the method ARIMA had better precision than LOF and S-H-ESD. Finally, SR was the closest method to better results regarding reminders, with a value of 0.52. Nevertheless, it should be noted that these algorithms could not detect all the types of anomalies observed in the actual

deployments, which means that each algorithm was efficient in a specific type. The peculiarity of this method was that it could define the patterns according to the need to detect multiple anomalies with great precision and efficiency.

4.5.2.4. Local Data Experimentation (Variable Series)

To devalue the algorithm in another context, the ARIMA method's data sets were used (IPI and HIPC). Accordingly, manually analyzed a first subset of the series to specify the patterns upon defining a different pattern based on the type of anomalies to label the remarkable points in the temporal series (3 patterns). Then, it proceeded to a composition of these labels to detect anomalies (4 compositions of labels). Then, a second subset had used to comprise two-time series of these two data sets to carry out the experiments. All experiments were done by types of anomalies.

4.5.2.5. Datasets Test and Evaluation

Table 4. 10 Comparison of anomaly detection methods on the benchmark data.

	Evaluation			
Algorithm	Precision	Recall	Accuracy	F-Measure
ARIMA	**0.91**	**0.7**	**0.83**	**0.72**
LOF	0.11	0.2	0	0
SH-ESD	0.2	0.2	0.33	0.25
SR	0	0	0	0
LS	0	0	0	0
ARP	**0.94**	**0.8**	**0.85**	**0.83**

The results from Table 4.10 show that the Constant Ruler (CR) was not tested on the HIPC and IPI data sets because the anomalies present in those data did not fit that type of anomaly. The ARP, ARIMA, and LOF algorithms were applied to these data sets with a number of neighbors set to 20, as well as the S-H-ESD, Level Shift (LS), and Short Rule (SR) algorithms. The Short Ruler (SR) and Level Shift (LS) algorithms were the least accurate among these algorithms, while the new algorithm proposed in the package ARIMA was the best and was able to detect the majority of the observed

anomalies with few errors. The values were bad for the Level Shift (LS) because the algorithm did not properly identify the stop points where structural changes had been seen. Indeed, a changepoint was an instance in the time where the statistical properties before and after this time point were different. Therefore, the algorithm was efficient in case of potential changes. However, the changes in the IPI data and HIPC were not so significant that the algorithm (LS) arrived at the detection properly. For the short rule, the choice of threshold was defined by calculating the mode of the histogram values. Then the difference between each successive point was compared with the threshold. This mode of threshold definition turns out to be little adapted to the data of ARIMA.

4.5.3. Chapter Summary

This chapter presents the ARP approach based on patterns applied to univariate time series data coming from the datasets. The research method consisted of two stages: it marks all the remarkable points present in the time series based on detection patterns, then precisely identifies multiple anomalies present by compositions of labels. This approach needs expertise in the application domain to define labels' patterns and compositions effectively. Conversely, although less effective, the literature methods did not require much expertise to apply the solution.

The experimentation was based on an actual context: the data capture. Evaluation of this method was illustrated by using everything, starting with all the literature data. The algorithm was compared to five different anomaly detection techniques methods and was efficient in discovering the APT behavior and characteristics. Based on the evaluation criteria, precision, recall, and f-measure showed that this algorithm was more efficient for detecting different agent anomalies observed during deployments, leading to minimizing the false detections.

Next chapter is the results and discussion, where the findings of the research are presented and analyzed. The chapter includes a summary of the data collected and the results obtained from the analysis. It also includes a discussion of the implications of the results and how they relate to the research questions or hypotheses. It provides a clear understanding of the results and their significance to the field of study.

CHAPTER FIVE

RESULTS AND DISCUSSION

5.1. INTRODUCTION

This research studies how to apply Machine Learning technology to detect computer networks' anomalous behaviors. Anomaly detection is a vast field of action. It is usually weakly defined as a class of problems while trying to identify anomalous data points or sequences in pre-selected datasets. This study has started with a time series presentation specifically demonstrating data collected upon a time (such as a sensor in an environment or sensor's network), where the definition of anomalies needs to be considered upon temporal dependencies. Next, the univariate simple time series model will be applied to the sensor readings and the literature datasets to see how to construct an automated model that predicts future time steps in the series. Then, we can see how to use this model to identify points in time that stand out as potential anomalies.

A deep study has been set in place to limit the storage of the technology beyond APT detection using machine learning solutions. For that, we were able to draw an innovative solution using machine learning technology in a customized and enhanced manner that starts by generating the best fit from the extracted and selected features. Those features would be the ones to use for building the second contribution of the APT detection scenario. Admitting the list of features has been done after a long correlation process by using a dedicated correlation rule. The rule applied for correlation is the One-Way ANOVA Calculator.

The output of this correlation model calculator is simple enough to understand and match the needs for feature extraction and selection. Upon the round and reference to the One-Way ANOVA model, the features are fully correlated when minimal (near or less than zero).

We can see in the following Table 5.1 the correlation of five sample features considered independent variables in the dataset, which finally had 57 final non-correlated variables. This step is necessary to eliminate the highly correlated variables and keep only the non-correlated variables in the dataset.

Table 5. 1 The correlation of five sample features

SizeOfCode	-1.00	2.66	2.15	2.77	2.78
SizeOfImage	2.66	-1.00	0.51	5.44	5.44
CheckSum	2.15	0.51	-1.00	4.93	4.93
LoadConfigurationSize	2.77	5.44	4.93	-1.00	0.00
VersionInformationSize	2.78	5.44	4.93	0.00	-1.00
	SizeOfCode	SizeOfImage	CheckSum	LoadConfigurationSize	VersionInformationSize

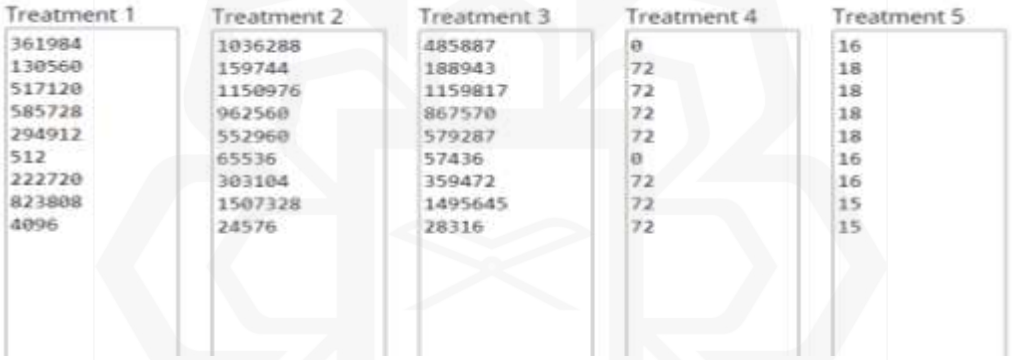


Figure 5. 1 One-Way ANOVA interface

Table 5. 2 One-way ANOVA calculator

Wise Comparisons	$Q_{.05}$ = 4.0391 $Q_{.01}$ = 4.9308
T_1:T_2	Q = 2.66
T_1:T_3	Q = 2.15
T_1:T_4	Q = 2.77
T_1:T_5	Q = 2.78
T_2:T_3	Q = 0.51
T_2:T_4	Q = 5.44
T_2:T_5	Q = 5.44
T_3:T_4	Q = 4.93
T_3:T_5	Q = 4.93
T_4:T_5	Q = 0.00

Note. T = Treatment; Q = Studentized range static;

Behind the one-way ANOVA (Figure 5.1*)* calculator, as shown in Table 5.2, there must be a couple of things to consider. First, a blue-colored value for the Q (s*tudentized range statistic*) variable indicates a significant result indicating non-correlated features. Second, it's to recognize that each Q value is the correlation of two Treatments; each is pointing to a feature value.

In brief, difficulties that could inhibit accurately detecting APT attacks may include methodological issues in the independent variables, such as multicollinearity issues. For that reason, it was mentioned above that we eliminated highly correlated independent variables.

Also, we went beyond enhancing the available standardized techniques to detect APT attacks. And in this regard, we relied on experts' experience to detect remarkable points and then identify anomalies. We proposed the ARP algorithm built in two phases.

The first phase consists of detecting and annotating the remarkable points in the time series. The second phase involves identifying anomalies from the compositions of remarkable points. It is important to note that ARP is generic; in the sense that it is easily adaptable to other situations, providing different inputs where appropriate.

To enhance standardized techniques to detect the APT attack, we propose automating the traditional approaches using times series, ARP, and CDT. For example, Figure 5.2 shows one detected pattern out of 9 possible patterns. In addition, we addressed the limitations of traditional detection methods that were carried out manually by automating the process through machine learning. For instance, Figure 5.2 shows instances of whether there was a positive peak behavior or the beginning of a positive peak. In doing so, we were able to enhance the standardized techniques to detect APT attacks.

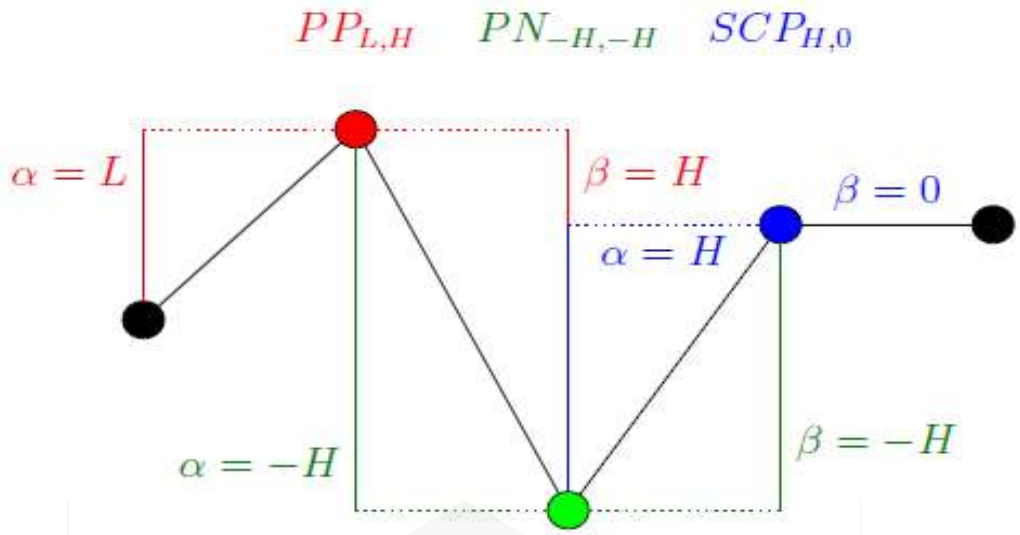


Figure 5. 2 remarkable points represented by patterns named

Figure 5.2 illustrates remarkable points represented by patterns named $PP_{L,H}$, $PN_{-H,-H}$, $SCN_{H,0}$. Those patterns are drawn from a list of nine(9) patterns provided as input to the algorithm in phase 1 of the ARP algorithm.

- $PP_{L,H}$ is a positive peak such that: PP presents the variation $\boldsymbol{\alpha}$ = L, $\boldsymbol{\alpha}$ =]0, 0.5] (marked L) and $\boldsymbol{\beta}$ =]0.5, 1] (marked H).
- $PN_{-H,-H}$ is a positive peak such that: $\boldsymbol{\alpha}$ = -H and $\boldsymbol{\beta}$ = -H ;
- $SCP_{H,0}$ is the beginning of a positive constant such that $\boldsymbol{\alpha}$ = H and $\boldsymbol{\beta}$ = 0.

This particular pattern signifies the initiation of a positive constant phase and is characterized by specific α and β values. It marks the commencement of a positive constant segment, where the α value is denoted as H (high), and the β value is set to 0. These defined patterns serve as input for the initial phase of the ARP algorithm. This phase involves the analysis and recognition of these distinctive patterns within a given dataset or time series data. Each pattern possesses distinct attributes that the algorithm leverages to discern and categorize diverse data points, encompassing features like peaks, constant intervals, and variations. By utilizing these patterns, the algorithm effectively identifies and categorizes specific data points with precision.

The contribution of this research has also identified the best customized AI automating utility to enhance ML processes in detecting up-to-date behaviors.

By exploring the math functions adopted in various Machine Learning standards and using the expertise of the on-field experts, we reached to bring an enhanced anomaly detection model, more specifically, APT malware detection. In order to identify optimized, automated utility to enhance the ML process for detecting (ab)normal behaviors, we employed a 3-step process with three algorithms: 1) an algorithm related to evaluating patterns, 2) an algorithm to examine remarkable points detections, and 3) an algorithm to generate a rule based on the classification of whether the attack is benign or malicious through composition-based decision tree (CDT). The three algorithms are summarized below:

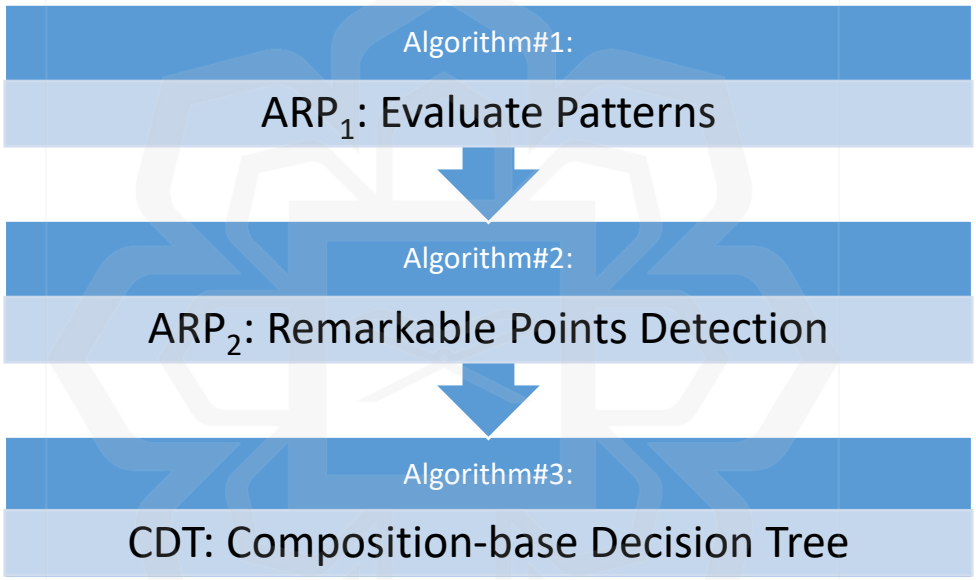


Figure 5. 3 Three Algorithms for Enhanced Model

As explained above in Figure 5.3, three stages were done for identifying the anomalies through an automated process, starting with evaluating the patterns, creating remarkable detected points, and finally identifying the patterns by examining them through a decision tree algorithm. This method has led to automating APT detection through an ML-based approach. First, assess existing patterns to achieve the proposed objective of detecting up-to-date behaviors. Pattern evaluation is based on nine different patterns. Once patterns of APT behavior are detected through time series data, the remarkable points detection algorithm is executed to find the exact pattern that reflects APT behavior. Once the remarkable points are detected, it classifies the behaviors' memberships, whether benign or malicious. The later step is carried out by using the CDT algorithm.

The previous objective aimed to enhance ML processes to detect up-to-date behaviors. This objective is necessary to enhance such behavioral patterns. However, the process remains isolated without a systematic approach based on agreed rules that enable the aforementioned enhancement process. To achieve such a systematic approach, we created several rules that functioned to regularly feed the automated detection method whenever a new use case or behavior emerges. For example, we used the SNORT application that functions as an intruder detection software that has built it rules and gets updated regularly whenever new behavior is detected. In addition to the known built-in rules that originally come with SNORT as illustrated in Figure 5.4 a contribution to the body of knowledge had done by proposing additional rules. For example, as relied on the results of objective 3 (stated above) and automatically fed the SNORT with newly formulated rules. These new rules that helped to feed SNORT were derived from the automated process in the previous step (e.g., pattern assessment, time series of ARP, and CDT). These newly added and saved rules function as a way to continuously train the detection algorithms and hence enhance the robustness of detection accuracy.

An example of existing built-in rules in SNORT was taken to demonstrate the above-stated approach. For instance, SNORT comes, by default, with a rule header and a rule option explained in Figure 5.4. A rule header includes data points such as action, protocol, source address, source port, direction, dest address, and dest port. The proposed methodology helps to continuously update the existing SNORT with newly detected actions, source and dest addresses, and other data points.

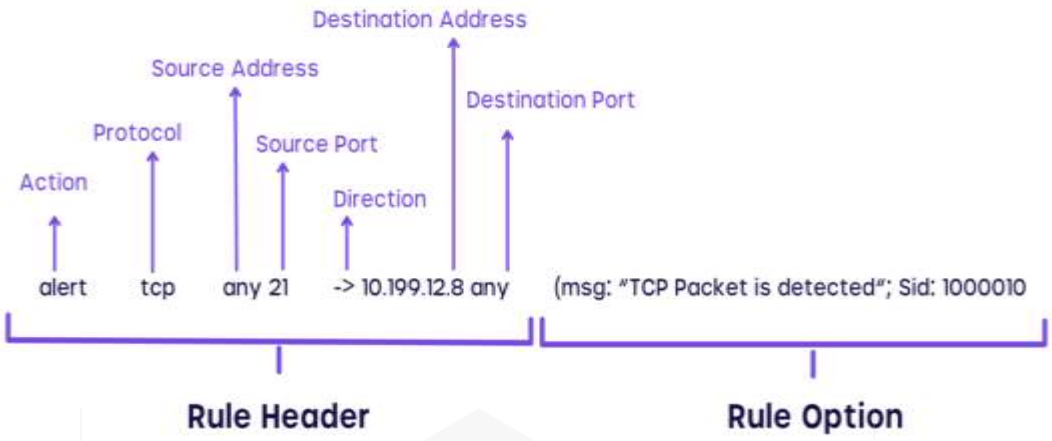


Figure 5. 4 SNORT built-in rules structure

To validate whether the proposed method is robust in detecting APT, a comparison with existing algorithms was reported in the literature. For example, the proposed method was compared against other algorithms such as J48, C4.5, Naive Bayes Tree, PRISM, JRip, and OneR. This comparison step was performed to determine whether the proposed method would outperform the existing algorithms.

Thus, the research carried out the comparison mechanism by segregating the work into two parts for evaluation purposes: the ARP and the CDT. The WEKA software was used to evaluate the approaches found and compare them with the already available standards and algorithms. Firstly, a new dataset was tested to get the research's effectiveness; besides, compared to the graphical realization, the ARP referred to the J48, C4.5, and the Naïve Bayes Tree standards.

Secondly, the PRISM, JRip, and OneR standards have been referred to compare the rule-based approach found through this research, the CDT. Before running the comparison and demonstrating the evaluation metrics, first, an offered background and description of the existing methods and standards from the literature done (e.g., classification).

5.2. CLASSIFICATION METHODS

Machine learning has two popular process phases: a learning phase, in which the classification algorithm is trained, and a classification phase, where the labeling algorithm is made for the new data. Classification is a technique for categorizing the data and mapping them into predefined groups and classes, also called supervised learning.

In this research, an alternative classifier had used to represent different recognition models, including Jrip, OneR, and J48 estimated using Waikato Environment for Knowledge Analysis (Weka) software.

5.2.1. RIPPER Algorithm

Jrip, also known as RIPPER, is a popular algorithm used for learning propositional rules through Repeated Incremental Pruning to Reduce Error. It is based on decision tree techniques and is able to create a model for anomaly detection by building a set of rules that can be used to identify new data sequences.

5.2.2. One Rule Algorithm

The OneR classification method is an abbreviation of “One Rule," a simple rule-based classification algorithm that generates one rule for each predictor in the data based on attributes. Although, it is a very effective method and widely used in machine learning applications, especially while producing simple rules for humans to interpret.

5.2.3. Java Statistical Classifier

The C4.5 algorithm, also known as J48, is one of the most well-known a supervised machine learning method used to classify data by creating decision trees based on information theory. It is an advanced version of the ID3 algorithm and is implemented in Java, known as J48 in the WEKA data mining tool. J48 has additional features such as handling missing values, pruning decision trees, handling continuous attribute values, and generating rules for classification. It is widely used in various fields for data classification, including interpreting clinical data for disease diagnosis and classifying e-governance data.

5.2.4. Naive Bayes Algorithm

The Naive Bayes algorithm is a decision tree-based classification technique that relies on Bayes' Theorem. It operates under the assumption that the predictors in the data are independent of one another, meaning that the presence of one feature does not affect the presence of any other feature.

5.2.5. PRISM Algorithm

The Prism algorithm deals with the issue of repeated sub-trees that often occur when using decision tree learning algorithms. The algorithm is designed to choose attributes based on their relevance to a particular class. It selects a specific class as the target and learns a set of rules that separates that class from the others. This process is repeated by selecting each class as the target class.

5.3. EVALUATION PROCESSES

In this first implementation, the deployment of the ARP solution was able to label anomalous points within the chosen dataset, as shown in the following Table 5.3. Also, selected random rows from the dataset of different types of malicious are presented in Table 5.4.

Table 5. 3 deployment of the ARP solution to label anomalous points[5]

	J48			C4.5			NB Tree			ARP		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
benign	0.04	0	0.02	0.04	0	0.02	0.111	0.023	0.038	0.986	0.977	0.982
malicious	0.96	1	0.979	0.96	1	0.979	0.96	0.992	0.976	0.974	0.987	0.981
AVG	.667	0.96	.667	.667	0.96	.667	0.926	0.953	0.938	0.987	0.988	0.988

[5] Dataset: GoodWare (1): https://machinelearningmastery.com/how-to-tune-a-machine-learning-algorithm-in-weka/#:~:text=Tuning%20k%2DNearest%20 Neighbour,instance%20just%2Din%2Dtime.

Table 5. 4 Sample from the used dataset in the test

A	B	C	D	E	F	G
SizeOfCode	CheckSum	Timing	SizeOfImage	SizeOfInitializedData	Malware	Remarkable Points Detection
113152	2018335512	1566779476	483328	189440	1	constant
73728	0	1607501316	98304	139264	1	level shift
28672	409729	1604736516	393216	20480	1	noise
40448	2598754130	1630829316	81920	28160	1	peak
53760	375001	1631434116	458752	89600	1	noise
1667584	2380904	1631866116	2387968	15360	1	level shift
47616	2485637	1623917316	98304	282112	1	peak
12288	0	1576656516	98304	13824	1	constant
37376	51072	1577261316	69632	162816	0	begnine
16384	28252	1574496516	36864	318976	0	begnine
1183744	18898230	1603267716	18882560	20992	0	begnine
0	0	1608538116	278528	610304	0	begnine
85504	370840	1596010116	376832	841728	0	begnine
531456	610969	1595837316	606208	46080	0	begnine

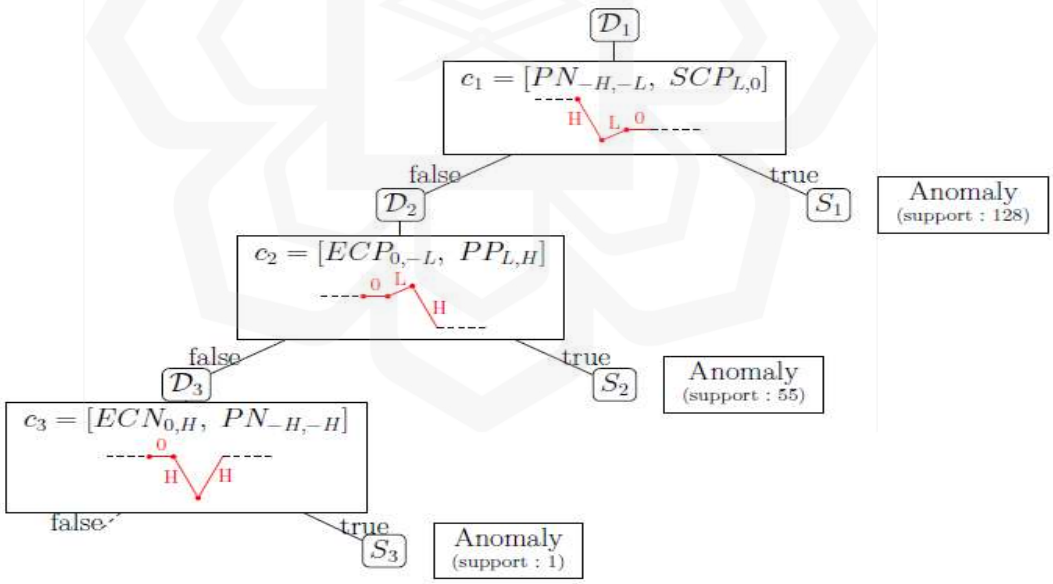


Figure 5. 5 Decision tree for patterns evaluation

Then, referred to the Weka software to evaluate available anomaly detection standards on the selected datasets samples from the local environment and the literature. Also, the CDT as shown in Figure 5.5 has tested after adopting the designed model, which had different phases summarized as follows:

- Preprocessing the data to analyze
- Labeling the prepared data points
- Applying the time series concept on the labeled data to generate a composed decision tree.
- Evaluate the final model by applying the Bayesian tree machine learning solution.

Evaluation of the rule-based algorithms[6] has to have in input the outcome the ARP solution has delivered a remarkable points composition with the view of a decision tree. And here, to this outcome, had been applying the rule-based standards to visualize the APT detection solutions' effectiveness in rules generation. Those rules are to be the input of the network filter that had to adopt for this research; the SNORT. As for the CDT, the Naive Bayes was the selected solution to generate the CDT rules.

Table 5. 5 Comparison between existing algorithms against the proposed optimized algorithm

	PRISM			JRip			OneR			CDT		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Benign	0.136	0.268	0.18	0.1	0.023	0.038	0	0	0	0.5	0.047	0.085
Malicious	0.969	0.931	0.95	0.96	0.991	0.975	0.96	0.998	0.978	0.961	0.998	0.979
AVG	0.937	0.905	0.92	0.926	0.952	0.938	0.921	0.958	0.939	0.943	0.96	0.943

As can be seen from Table 5.5 that the proposed enhanced algorithm, on average, outperformed the existing algorithms reported in the literature. For example, the precision estimate of detecting whether the attack was malicious for the proposed model (CDT) was 96%, consistent with precision estimates by the existing algorithm: PRISM 96.9%, JRip 96%, and OneR 96%. However, the proposed model outperformed the existing algorithm when detecting whether the attack was benign or not. For example, the precision of CDT in this scenario was 50% compared to 0% for OneR, 10% for JRip, and 13.6% for PRISM. Furthermore, OneR readings are 0 because it works for datasets with categorical features and relatively simple relationships between

[6] Dataset for CDT evaluation: https://github.com/GuansongPang/ADRepository-Anomaly-detection-datasets/blob/main/categorical%20data/solar-flare_FvsAll-cleaned.arff

features and classes. However, it cannot effectively handle complex data distributions or interactions between multiple features while CDTis flexible and capable of capturing complex interactions between. OneR selects the feature that results in the best separation of classes and creates a rule based on the most common class for each value of that feature.

Overall, the average score shown in Table 5.5 indicates that the proposed model outperformed the existing algorithms. For example, the average precision estimate for the proposed model was 94.3% compared to the existing algorithms, with values of 93.7%, 92.6%, and 92.1% for PRISM, JRip, and OneR, respectively.

The evaluation of the CDT algorithm (Figure 5.6) was achieved by adopting the algorithm number 3 outputs to the NB Tree standard upon the WEKA software. Algorithm number 3 of this research is described as follows:

Algorithm 3 CDT : Composition-based Decision Tree

Input : $D = \{d_1, d_2, ..., d_{N-\omega+1}\}$ a set of observations
Output : N_{root} the root node of CDT

$N_{root} \leftarrow Node(D, null, null, null)$
$q \leftarrow [N_{root}]$ *// construct the queue of nodes to split*
while $q \neq \emptyset$ **do**
  $N_j \leftarrow q.pop()$ *// dequeue the first node from the queue*
  $D_j \leftarrow N_j.observations$
  $C_j \leftarrow list\_of\_all\_possible\_compositions(D_j)$
  $maxGain \leftarrow 0$
  $c_{best} \leftarrow null$
  // Choose the composition that has the best Gain
  for all $c \in C_j$ **do**
    if $IG(D_j, c) > maxGain$ **then**
      $maxGain \leftarrow IG(D_j, c)$
      $c_{best} \leftarrow c$
    end if
  end for
  if $G(D_j) \neq 0$ and $maxGain \neq 0$ **then**
    $D_{inc} \leftarrow \{d \in D_j | c_{best} \subseteq_o d\}$
    $D_{exc} \leftarrow \{d \in D_j | c_{best} \not\subseteq_o d\}$
    $N_{inc} \leftarrow Node(D_{inc}, null, null, null)$
    $N_{exc} \leftarrow Node(D_{exc}, null, null, null)$
    $q.append(N_{inc})$ *// enqueue child nodes*
    $q.append(N_{exc})$ *// enqueue child nodes*
    $N_j.composition \leftarrow c_{best}$
    $N_j.childTrue \leftarrow N_{inc}$
    $N_j.childFalse \leftarrow N_{exc}$
  end if
end while
return N_{root}

Figure 5. 6 Composition-based Decision Tree Algorithm

5.4. EVALUATION MEASURES

The platform set for the research was heterogeneous because it had various operating systems utilized to inject the APT threat, mainly Windows- and Linux-based distributions. Thus, constraints were recorded when logging logs and generating local datasets, saying that the patterns were not homogeneous, which caused a time latency gap to move from the level of data capturing to the next until the data was set upon a homogeneous pattern.

5.5. DISCUSSION:

This research contributes to the body of knowledge in several ways. Prior to presenting the added value of this research, it is important first to understand how the currently available algorithms function. Existing algorithms, as reported in the literature, work as classification methods. For example, once trained, existing algorithms can tell whether APT attacks are benign or malicious. The limitations of these algorithms are that they only tell you that the output of the prediction model is 0 (not detected) or 1 (detected); however, these algorithms are limited in the sense that they cannot tell the breadth and depths of the nature of these attacks. In this research, we addressed this gap by proposing a three-stage model. In the first stage (Algorithm 1), we used the algorithm as implemented traditionally. For example, the algorithm will function to detect whether APT is detected or not detected. This stage is referred to as the pattern evaluation phase. The output of the first stage becomes the input for the second stage. For example, if APT is detected (malicious), the research proposed an algorithm that analyzes this attack's nature in depth. This is done by determining what is called remarkable points ARP. These remarkable points offer nine different patterns, each reflecting the nature of the attack. For example, each pattern signifies the breadth and depth of the attack (e.g., peak, noise, plate, level change). Once these outputs from the second stage (Algorithm 2) are produced in a form of patterns, stage 3 is then automatically executed. Stage 3 or Algorithm (3) serves as a robust classification problem that functions to correlate and compare among different scenarios. These scenarios are then fed (input) to the SNORT, and these scenarios or rules become like trained data that keeps accumulating as more attacks are received. Correlating and saving these rules against the patterns is carried out by using CDT. For these reasons, it can be said that this is a

Machine Learning problem because the algorithm proposed in the work continuously detects attacks, saves possible behaviors or scenarios, and is used to classify the nature of the APT attack accurately.

5.5.1. Methodological and Technological Contributions

- Previous AI/ML-based algorithms aimed at detecting APT attacks passively (not in real-time). This research contributes to the literature by proposing a framework that functions as a building block to fully implement active and autonomous enhanced algorithms that can detect APT attacks.
- In addition to the known built-in rules that initially came with SNORT, a contribution to the body of knowledge had to be made by proposing additional rules. For example, they relied on the results of objective 3 (stated above) and automatically fed the SNORT with newly formulated rules. This method helps to enhance the internationally recognized APT detection methods.
- This work also contributes to existing prediction models by offering higher detection rates of APT attacks with more accuracy levels. A framework for APT detection as a solution based on AI automation technology to achieve the best behavior.
- To design an algorithm to detect the threat while analyzing data at rest.
- To develop an intelligent self-secure network that could be an innovative technology for an excellent human state and digital security in next near future, remarkably during the interoperation of the heterogeneous frameworks.

5.5.2. Practical Contribution

- To help organizations implement the proposed APT detection methods with fewer costs and higher levels of accuracy to avoid possible negative repercussions caused by security threats.
- Help IT professionals and cybersecurity experts to be more aware of the capabilities of the emerging state-of-the-art algorithms that can be enhanced, employed, and executed automatically to detect potential security threats.
- To highlight the importance of machine learning technologies for improving organizational IT security-related practices.

- To bring to the attention of the decision-makers that industry revolution 4.0 (e.g., ML/AI) can be tailored and customized based on organizational needs (e.g., types of security threats, anomalies, organizational user behaviors).
- To provide the world's nations the ability to create effective national global secure digital environments.

5.6. Chapter Summary:

Anomaly detection is a comprehensive technology narrowed down in this research based on using artificial intelligence technological solutions. Customization is a key to succeed the limitations and shortage of available solutions. We tried to trespass the explored limits towards the APT detection solutions by customizing a detection model based on the use of different technologies. This solution has set the use of the univariate time series to generate timely frames with the capabilities of predicting and forecasting future inputs, and they customized a computerized solution to generate a list of remarkable points within the dataset in question. Finally, we were able to generate a customized decision-based tree for the best rule generation that would be the APT detection solution. Next chapter will be the conclusion and future work. The chapter is going to highlight the limitations of the study and provide suggestions for future research.

CHAPTER SIX

CONCLUSION AND FUTURE WORK

6.1. OBJECTIVES OBTAINED

This research aimed to improve the detection of advanced persistent threats (APT) using AI and machine learning (ML) techniques. It focused on identifying and addressing the following goals, which were successfully achieved:

- Goal 1: Methodological issues that could inhibit the detection of APT attacks were identified and addressed.
- Goal 2: Existing standardized techniques for detecting APT attacks were enhanced through the proposed method (ARP and CDT) for active and autonomous enhanced algorithms and the proposed new rules for SNORT.
- Goal 3: An efficient autonomous APT detection model was developed, which was able to detect APT attacks in real time and improve upon previous passive AI/ML-based algorithms.
- Goal 4: The results of the autonomous APT detection model were evaluated and compared against currently used algorithms. It was found to have improved APT attacks' accuracy and detection rates.

The proposed method for active and autonomous enhanced algorithms in this research successfully detected APT attacks in real time, showing a significant improvement over previous passive AI/ML-based algorithm. Furthermore, the proposed new rules for SNORT, a widely used intrusion detection system, were generated using the results from the autonomous APT detection model, adding to the existing body of knowledge and enhancing the internationally recognized APT detection methods.

The proposed APT detection approach based on AI automation technology also succeeded in improving the accuracy and detection rates of APT attacks. Additionally, the research focused on analyzing data at rest to detect threats and developing an intelligent self-secure network, which has the potential to serve as an innovative technology for improved human and digital security in the future. Overall, this research provides valuable contributions to the field of APT detection using AI and ML techniques. Further studies can build on these findings to enhance the efficiency and effectiveness of APT detection methods.

6.2. CONCLUSION

This chapter reviewed current studies on detection Advanced Persistent Threats using machine learning techniques, specifically focusing on papers published after 2015. A wide range of machine learning techniques, including single, hybrid, and ensemble classifiers, are considered in the review. The results of related work suggest that there is still room for further research in developing intrusion detection systems using machine learning techniques.

In conclusion, machine learning (ML) has been proposed as a promising method for detecting advanced persistent threats (APTs) due to its ability to identify abnormal network behavior. Various studies have shown the effectiveness of using different ML algorithms, such as J48, SVM, and decision tree, in detecting APTs. However, it is important to note that there are still limitations and challenges that need to be addressed in order to improve the performance and accuracy of these models. In the future, more research should be conducted to explore new techniques and tools that can be utilized to enhance the detection of APTs. Additionally, further studies are needed to evaluate the performance of these models in different types of systems and environments.

6.3. FUTURE WORK

This section highlights the limitations and deficiencies of the current research, and emphasizes the need for future work to address these shortcomings. With advancements in technology and new research tools, future studies can strive to improve and update the current methods in order to stay current with the ever-evolving field of research.

Possible areas for future research could include:

- Comparing different ensemble classifiers and hybrid classifiers in terms of prediction accuracy as the choice of a single classifier for model comparison and evaluation may not be the best option as a baseline classifier.
- Examining the design of more advanced classifiers by combining ensemble and hybrid classifiers since the goal is to achieve collaboration rather than competition between multiple classifiers.
- Investigating various feature selection approaches as the reviewed studies that consider feature selection only use one specific method, and it is not clear which method is best, especially under different classification techniques for intrusion detection.

- Developing more sophisticated and advanced ML algorithms for identifying and detecting APT attacks.
- Incorporating behavioral analysis techniques to detect APT attacks in real-time.
- Using ensemble methods to combine the strengths of multiple ML algorithms for improved detection accuracy.
- Incorporating new features into the datasets such as network flow data, system logs, and endpoint data to enhance the detection capabilities of ML models.
- Exploring the use of deep learning techniques such as convolutional neural networks and recurrent neural networks for APT detection.
- Investigating the use of transfer learning and domain adaptation techniques to adapt ML models trained on one dataset to new and unseen data.
- Incorporating explainable AI (XAI) techniques to make the decision-making process of the ML models more transparent and understandable.
- Evaluating the performance of ML-based APT detection models in real-world environments to understand their limitations and areas of improvement.
- Combining ML with other intrusion detection methods such as rule-based systems and signature-based systems to create more robust and comprehensive APT detection systems.
- Developing methods to continuously update and improve the ML models as new attack patterns and techniques are discovered.

References

Abdelrahman, O., Access, P. K.-I., & 2020, U. (2020). Assembly line anomaly detection and root cause analysis using machine learning. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/9218922/

Abghari, S., Boeva, V., … N. L.-2018 17th I., & 2018, U. (2018). A minimum spanning tree clustering approach for outlier detection in event sequences. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/8614207/?casa_token=LMi1qeO_H9oAAAAA:QJ8EW_757-8yl2iuQjEgVAS-mNeTGqE9n4VghmB7xB1FkjdfAuYwRIHUCCPQ7eACjUSMJvIegqKTXg

Abghari, S. Z., Chemical, A. I.-I. J. of C. and, & 2018, undefined. (2018). Determination of suitable operating conditions of fluid catalytic cracking process by application of artificial neural network and firefly algorithm. *Sid.Ir*, *37*(6). https://www.sid.ir/FileServer/JE/84320180616.pdf

Abramovich, F., … T. B.-J. of the R., & 2000, U. (2000). Wavelet analysis and its statistical applications. *Wiley Online Library*. https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9884.00216

Acuna, E., -, C. R., Mayaguez, U. of P. R. at, & 2004, U. (2004). A meta analysis study of outlier detection methods in classification. *Academic.Uprm.Edu*. https://academic.uprm.edu/~eacuna/paperout.pdf

Adams, C., Sens, I. J., Data, N., Adams, C., Aa, T., Bissessar, D., Brien, R., Fan, J., Hezaveh, M., & Zahed, J. (2019). Using Machine Learning to Detect APTs on a User Workstation. *International Journal of Sensor Networks and Data Communications*, *8*(3), 1–8.

Adhikari, J., Zantye, N., & Rao, P. R. (2012). *Mining and Analysis of Time-stamped Databases*. http://irgu.unigoa.ac.in/drs/bitstream/handle/unigoa/4507/adhikari_j_2012.pdf?sequence=1

Agrawal, S., Science, J. A.-P. C., & 2015, U. (2015). Survey on anomaly detection using data mining techniques. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S1877050915023479

Agrawal, V., Bhattacharyya, C., Niranjan, T., & Susarla, S. (2009). Discovering rules from disk events for predicting hard drive failures. *8th International Conference on Machine Learning and Applications, ICMLA 2009*, 782–786. https://doi.org/10.1109/ICMLA.2009.62

Ahmed, Q., Raza, S. A., & Al-Anazi, D. M. (2021). Reliability-based fault analysis models with industrial applications: A systematic literature review. *Quality and Reliability Engineering International*, *37*(4), 1307–1333. https://doi.org/10.1002/QRE.2797

Al-Amri, R., Kumar Murugesan, R., Man, M., Abdulateef, A. F., Al-Sharafi, M. A., & Alkahtani, A. A. (2021). A review of machine learning and deep learning techniques for anomaly detection in IoT data. *Mdpi.Com*, *11*(12). https://doi.org/10.3390/app11125320

Al-khatib, A., … B. M.-… D.-E. I. of, & 2020, U. (2020). A survey on outlier detection in Internet of Things big data. *Researchgate.Net*. https://doi.org/10.1049/PBPC025E_ch11

Al-Mohannadi, H., Mirza, Q., Namanya, A., Awan, I., Cullen, A., & Disso, J. (2016).

Cyber-attack modeling analysis techniques: An overview. *Proceedings - 2016 4th International Conference on Future Internet of Things and Cloud Workshops, W-FiCloud 2016*, *January 2018*, 69–76. https://doi.org/10.1109/W-FiCloud.2016.29

Al-Yaseen, W. L., Othman, Z. A., & Nazri, M. Z. A. (2017). Real-time multi-agent system for an adaptive intrusion detection system. *Pattern Recognition Letters*, *85*, 56–64. https://doi.org/10.1016/j.patrec.2016.11.018

Aldweesh, A., Derhab, A., Systems, A. E.-K.-B., & 2020, U. (2020). Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0950705119304897

Alfergani, H. (2021). *Unsupervised Learning for Anomaly Detection in Remote Sensing Imagery*. https://search.proquest.com/openview/74d7188707b2dac9c100fd293af4b6dc/1?pq-origsite=gscholar&cbl=18750&diss=y

Alizadeh, M., Hamilton, M., Jones, P., … J. M.-E. S. with, & 2021, U. (2021). Vehicle operating state anomaly detection and results virtual reality interpretation. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0957417421003699

Alizadeh, R., Beiragh, R., Economics, L. S.-E., & 2020, U. (2020). Performance evaluation of complex electricity generation systems: A dynamic network-based data envelopment analysis approach. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0140988320302346

Aljawarneh, S., Yassein, M. B., & Aljundi, M. (2019). An enhanced J48 classification algorithm for the anomaly intrusion detection systems. *Cluster Computing*, *22*, 10549–10565. https://doi.org/10.1007/s10586-017-1109-8

Alminshid, K. A. A., & Omar, M. N. (2020). A framework of APT detection based on packets analysis and host destination. *Iraqi Journal of Science*, *61*(1), 215–222. https://doi.org/10.24996/ijs.2020.61.1.24

Alshamrani, A., Myneni, S., Chowdhary, A., & Huang, D. (2019). A Survey on Advanced Persistent Threats. *IEEE Communications Surveys & Tutorials*, *PP*(8), 25. https://doi.org/10.1109/COMST.2019.2891891

Alsoufi, M. A., Razak, S., Siraj, M. M., Nafea, I., Ghaleb, F. A., Saeed, F., & Nasser, M. (2021). Anomaly-based intrusion detection systems in iot using deep learning: A systematic literature review. *Mdpi.Com*. https://doi.org/10.3390/app11188383

Aminikhanghahi, S., & Cook, D. J. (2017). A survey of methods for time series change point detection. *Knowledge and Information Systems*, *51*(2), 339–367. https://doi.org/10.1007/S10115-016-0987-Z

analytics, M. L.-M. data mining and, & 2015, undefined. (2015). Listen to the sound of data. *Springer*, 419–446. https://doi.org/10.1007/978-3-319-14998-1_19

Aparecido, L., Júnior, P., Da Costa, K. A. P., Papa, J. P., & Pontara Da Costa, K. A. (2018). FITTING MULTIVARIATE GAUSSIAN DISTRIBUTIONS WITH OPTIMUM-PATH FOREST AND ITS APPLICATION FOR ANOMALY DETECTION. *Researchgate.Net*. https://www.researchgate.net/profile/Leandro-Passos-Junior/publication/325023185_FITTING_MULTIVARIATE_GAUSSIAN_DISTRIBUTIONS_WITH_OPTIMUM-PATH_FOREST_AND_ITS_APPLICATION_FOR_ANOMALY_DETECTION/links/5af216cc458515c2837668ca/FITTING-MULTIVARIATE-GAUSSIAN-DI

Ashrafuzzaman, M., Chakhchoukh, Y., Jillepalli, A. A., Tosic, P. T., De Leon, D. C.,

Sheldon, F. T., & Johnson, B. K. (2018). Detecting Stealthy False Data Injection Attacks in Power Grids Using Deep Learning. *2018 14th International Wireless Communications and Mobile Computing Conference, IWCMC 2018*, 219–225. https://doi.org/10.1109/IWCMC.2018.8450487

Aziz, A. S. A., Hassanien, A. E., Hanaf, S. E. O., & Tolba, M. F. (2014). Multi-layer hybrid machine learning techniques for anomalies detection and classification approach. *13th International Conference on Hybrid Intelligent Systems, HIS 2013*, 215–220. https://doi.org/10.1109/HIS.2013.6920485

Bahrami, P. N., Dehghantanha, A., Dargahi, T., Parizi, R. M., Choo, K. K. R., & Javadi, H. H. S. (2019). Cyber kill chain-based taxonomy of advanced persistent threat actors: Analogy of tactics, techniques, and procedures. *Journal of Information Processing Systems*, *15*(4), 865–889. https://doi.org/10.3745/JIPS.03.0126

Balke, N. S. (1993). Detecting Level Shifts In Time Series. *Journal of Business and Economic Statistics*, *11*(1), 81–92. https://doi.org/10.1080/07350015.1993.10509934

Bansal, R., Gaur, N., … S. S.--cloud system and big data, & 2016, U. (2016). Outlier detection: applications and techniques in data mining. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/7508146/

Barai, S., Sau, B., Barai, S., Dey, A., & Sau, B. (2018). Path following of autonomous mobile robot using passive rfid tags. *Ieeexplore.Ieee.Org*. https://doi.org/10.1109/MicroCom.2016.7522573

Bari, H. (2021). *Protecting an enterprise network through the deployment of honey pot*. http://lib.buet.ac.bd:8080/xmlui/handle/123456789/5922

Ben Kraiem, I., Ghozzi, F., Peninou, A., & Teste, O. (2020). CoRP: A Pattern-Based Anomaly Detection in Time-Series. *Lecture Notes in Business Information Processing*, *378 LNBIP*, 424–442. https://doi.org/10.1007/978-3-030-40783-4_20

Berrada, G., Cheney, J., Benabderrahmane, S., Maxwell, W., Mookherjee, H., Theriault, A., & Wright, R. (2020). A baseline for unsupervised advanced persistent threat detection in system-level provenance. *Future Generation Computer Systems*, *108*, 401–413. https://doi.org/10.1016/j.future.2020.02.015

Bhadane, A., Security, S. M.-I. I., & 2018, undefined. (2019). Detecting Lateral Spear Phishing Attacks in Organizations. *IET*, *13*(2), 133–140. https://doi.org/10.1049/iet-ifs.2018.5090

Bhaskaran, P., … M. C.-… of L. P. in the, & 2020, U. (2020). Future prediction & estimation of faults occurrences in oil pipelines by using data clustering with time series forecasting. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0950423020304903

Bhatnagar, A, … S. S.-2019 9th I., & 2019, U. (2019). Machine learning techniques to reduce error in the internet of things. *Ieeexplore.Ieee.Org*. https://doi.org/10.1109/CONFLUENCE.2019.8776619

Bhatnagar, Aadyot, Kassianik, P., Liu, C., Lan, T., Yang, W., Cassius, R., Sahoo, D., Arpit, D., Subramanian, S., Woo, G., Saha, A., Jagota, A. K., Gopalakrishnan, G., Singh, M., Krithika, K. C., Maddineni, S., Cho, D., Zong, B., Zhou, Y., … Wang, H. (2021). Merlion: A Machine Learning Library for Time Series. *Arxiv.Org*. http://arxiv.org/abs/2109.09265

Bindu, P., Applications, P. T.-J. of N. and C., & 2016, U. (2016). Mining social networks for anomalies: Methods and challenges. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S1084804516300029

Blázquez-García, A., Conde, A., … U. M.-A. C. S., & 2021, U. (2021). A review on outlier/anomaly detection in time series data. *Dl.Acm.Org*. https://dl.acm.org/doi/abs/10.1145/3444690

Bodström, T., & Hämäläinen, T. (2019a). A novel deep learning stack for APT detection. *Applied Sciences (Switzerland)*, *9*(6). https://doi.org/10.3390/app9061055

Bodström, T., & Hämäläinen, T. (2019b). A novel deep learning stack for APT detection. *Applied Sciences (Switzerland)*, *9*(6). https://doi.org/10.3390/app9061055

Boutaba, R., Salahuddin, M. A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F., & Caicedo, O. M. (2018). A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, *9*(1). https://doi.org/10.1186/s13174-018-0087-2

Breunig, M., Kriegel, H., Ng, R., 2000, J. S.-P. of the, & 2000, undefined. (2000). LOF: identifying density-based local outliers. *Dl.Acm.Org*. https://dl.acm.org/doi/abs/10.1145/342009.335388

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). PyNomaly: Anomaly detection using Local Outlier Probabilities (LoOP). *Joss.Theoj.Org*. https://joss.theoj.org/papers/10.21105/joss.00845.pdf

Brogi, G. (2018). Real-time detection of Advanced Persistent Threats using Information Flow Tracking and Hidden Markov Models. In *PhD Thesis*. http://www.theses.fr/2018CNAM1167

Brungard, C., Boettinger, J., Duniway, M., Geoderma, S. W.-, & 2015, undefined. (n.d.). Machine learning for predicting soil classes in three semi-arid landscapes. *Elsevier*. Retrieved July 4, 2022, from https://www.sciencedirect.com/science/article/pii/S0016706114003516

Bulusu, S., Kailkhura, B., Li, B., Access, P. V.-I., & 2020, U. (2020). Anomalous example detection in deep learning: A survey. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/9144212/

Caldwell, T. (2013). Spear-phishing: How to spot and mitigate the menace. *Computer Fraud and Security*, *2013*(1), 11–16. https://doi.org/10.1016/S1361-3723(13)70007-1

Cardenas, A., Privacy, P. M.-… S. &, & 2013, U. (2013). Big data analytics for security. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/6682971/

Carreño, A., Inza, I., Review, J. L.-A. I., & 2020, U. (2019). Analyzing rare event, anomaly, novelty and outlier detection terms under the supervised classification framework. *Springer*. https://link.springer.com/article/10.1007/s10462-019-09771-y

Carrera, D., Rossi, B., Fragneto, P., Recognition, G. B.-P., & 2019, U. (2019). Online anomaly detection for long-term ECG monitoring using wearable devices. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0031320318304102

Caviglione, L., Choras, M., Corona, I., Janicki, A., Mazurczyk, W., Pawlicki, M., & Wasielewska, K. (2021). Tight Arms Race: Overview of Current Malware Threats and Trends in Their Detection. *IEEE Access*, *9*, 5371–5396. https://doi.org/10.1109/ACCESS.2020.3048319

Chae, H., Jo, B., Choi, S., & Park, T. (2013). Feature Selection for Intrusion Detection using NSL-KDD. *Recent Advances in Computer Science 20132*, 184–187.

Chahla, C., Snoussi, H., Merghem, L., ICPRAM, M. E.-, & 2019, U. (2019). A Novel Approach for Anomaly Detection in Power Consumption Data. *Scitepress.Org*.

https://doi.org/10.5220/0007361704830490

Chamola, V., Garg, T., Raj, M., Gupta, S., Elhence, A., Atiquzzaman, M., & Niyato, D. (2021). A survey on the role of Internet of Things for adopting and promoting Agriculture 4.0. *Elsevier*, *187*, 1084–8045. https://doi.org/10.1016/j.jnca.2021.103107

Chandola, V, Banerjee, A., (CSUR), V. K.-A. computing surveys, & 2009, undefined. (2009). Anomaly detection: A survey. *Dl.Acm.Org*. https://dl.acm.org/doi/abs/10.1145/1541880.1541882

Chandola, V, Banerjee, A., Surveys, V. K.-A. C., & 2007, U. (2007). Outlier detection: A survey. *Researchgate.Net*. https://www.researchgate.net/profile/Vipin-Kumar-54/publication/242403027_Outlier_Detection_A_Survey/links/0deec52d83600bb86c000000/Outlier-Detection-A-Survey.pdf

Chandola, Varun. (2009). *Anomaly detection for symbolic sequences and time series data*. https://search.proquest.com/openview/59deed3486e0112b8ad59520b720d764/1?pq-origsite=gscholar&cbl=18750

Chatterjee, S. R., Chakraborty, M., & Chakraborty, J. (2011). Cognitive Radio Sensor Node Empowered Mobile Phone for Explosive Trace Detection. *International Journal of Communications, Network and System Sciences*, *04*(01), 33–41. https://doi.org/10.4236/ijcns.2011.41004

Chen, C., & Liu, L. -M. (1993). Forecasting time series with outliers. *Journal of Forecasting*, *12*(1), 13–35. https://doi.org/10.1002/FOR.3980120103

Chen, L., Gao, S., and, X. C.-I. J. of C., & 2020, undefined. (2017). Research on real-time outlier detection over big data streams. *Taylor & Francis*, *42*(1), 2017. https://doi.org/10.1080/1206212X.2017.1397388

Chen, M., Computing, B. C.-A. S., & 2014, U. (2014). Online fuzzy time series analysis based on entropy discretization and a Fast Fourier Transform. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S1568494613002780

Chen, P., Desmet, L., & Huygens, C. (2020). LNCS 8735 - A Study on Advanced Persistent Threats. In *Springer*. https://link.springer.com/chapter/10.1007/978-3-662-44885-4_5

Chen, P., Desmet, L., & Huygens, C. (2014). A study on advanced persistent threats. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8735 LNCS*, 63–72. https://doi.org/10.1007/978-3-662-44885-4_5

Chen, Y., Liu, X., Li, X., Liu, X., Yao, Y., Hu, G., … X. X.-L. and U., & 2017, U. (2017). Delineating urban functional areas with building-level social media data: A dynamic time warping (DTW) distance based k-medoids method. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0169204616302675

Chen, Z, Peng, Z., Zou, X., Annual, H. S.-C. C. S., & 2021, U. (2021). Deep Learning Based Anomaly Detection for Muti-dimensional Time Series: A Survey. *Library.Oapen.Org*. https://doi.org/10.1007/978-981-16-9229-1

Chen, Zhipeng, Peng, Z., Zou, X., & Sun, H. (2022). Deep Learning Based Anomaly Detection for Muti-dimensional Time Series: A Survey. *Communications in Computer and Information Science*, *1506 CCIS*, 71–92. https://doi.org/10.1007/978-981-16-9229-1_5

Cho, D. X., & Nam, H. H. (2019). A method of monitoring and detecting APT attacks based on unknown domains. *Procedia Computer Science*, *150*, 316–323.

https://doi.org/10.1016/j.procs.2019.02.058

Choi, J., Choi, C., You, I., & Kim, P. (2015). Polymorphic malicious javascript code detection for APT attack defence. In *Journal of Universal Computer Science* (Vol. 21, Issue 3). https://doi.org/10.3217/jucs-021-03-0369

Choi, K., Yi, J., Park, C., Access, S. Y.-I., & 2021, U. (2021). Deep learning for anomaly detection in time-series data: review, analysis, and guidelines. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/9523565/

Chowdhary, A. (2017). OVERVIEW ON DATA MINING WITH CLOUD COMPUTING. *Researchgate.Net*, *4*, 12. https://www.researchgate.net/profile/Sameer-Mohammad-4/publication/354372253_OVERVIEW_ON_DATA_MINING_WITH_CLOUD_COMPUTING/links/61348ea0c69a4e48797d8ac2/OVERVIEW-ON-DATA-MINING-WITH-CLOUD-COMPUTING.pdf

Chu, W. L., Lin, C. J., & Chang, K. N. (2019). Detection and classification of advanced persistent threats and attacks using the support vector machine. *Applied Sciences (Switzerland)*, *9*(21). https://doi.org/10.3390/app9214579

Cinque, M., Cotroneo, D., & Pecchia, A. (2023). Mining Dependability Properties from System Logs: What We Learned in the Last 40 Years. *Springer Series in Reliability Engineering*, 221–238. https://doi.org/10.1007/978-3-031-02063-6_12

Cogranne, R., Conference, F. R.-2013 I. I., & 2013, U. (2013). A new tomography model for almost optimal detection of anomalies. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/6738300/?casa_token=ouyMkWjLxvIAAAAA:zvCJyNnywGCamcnQQ6Lm0jmcTyw55Iagy-CFaJ2PrXs8pMln8st6BS-yJWwEVB4GVY8zgs5EkH0dTQ

Cogranne, R., Processing, F. R.-S., & 2014, U. (2014). Statistical detection of defects in radiographic images using an adaptive parametric model. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0165168413003599

Conti, M., Dehghantanha, A., Franke, K., & Watson, S. (2018). Internet of Things security and forensics: Challenges and opportunities. *Future Generation Computer Systems*, *78*, 544–546. https://doi.org/10.1016/j.future.2017.07.060

Dalmaz, H., Erdal, E., & Ünver, H. M. (2021). Machine Learning Approaches in Detecting Network Attacks. *Proceedings - 6th International Conference on Computer Science and Engineering, UBMK 2021*, 522–527. https://doi.org/10.1109/UBMK52708.2021.9558930

Du, H., Duan, Z., & Zheng, Y. (2021). CPMAN: Change point detection approach in time series based on the prediction of multi-stage attention networks. *International Journal on Artificial Intelligence Tools*, *30*(5). https://doi.org/10.1142/S0218213021500263

Dudde, N., Advanced, S. A.-I. J. of, & 2014, U. (2014). Performance Analysis On Decision Tree Algorithms. *Search.Ebscohost.Com*. https://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=09765697&AN=99652805&h=O14Q95rF9K%2Fi30EqgCE4RgO3WJGUX3fXTwsLeneie%2FIADsLwli1DxP%2F7JJEt8LeyOYVrllO0S6aqS6b%2Fb94zqQ%3D%3D&crl=c

Duffield, N., Haffner, P., 2009, B. K.-… I., & 2009, U. (2009). Rule-based anomaly detection on IP flows. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/5061947/?casa_token=XiWobaKVa3oAAAAA:zn8ohi64kdnOPK5oubmYz_ppEl1imbseL086xkvRwQU8R5ZOHWZP9hMiUVLcx37okgJd-unmYguhkQ

El Malki, N., Cugny, R., Teste, O., & Ravat, F. (2020). DECWA: Density-Based Clustering using Wasserstein Distance. *International Conference on Information and Knowledge Management, Proceedings*, 2005–2008. https://doi.org/10.1145/3340531.3412125

Esling, P., & Agon, C. (2012). Time-series data mining. *ACM Computing Surveys*, *45*(1). https://doi.org/10.1145/2379776.2379788

Extension, F., & Brandao, P. R. (2021). Advanced Persistent Threats (APT)-Attribution-MICTIC. *Journal of Computer Science*, *17*(5), 470–479. https://doi.org/10.3844/jcssp.2021.470.479

Fahim, M., Access, A. S.-I., & 2019, U. (2019). Anomaly detection, analysis and prediction techniques in iot environment: A systematic literature review. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/8733806/

Fayyad, U., Piatetsky-Shapiro, G., KDD, P. S.-, & 1996, undefined. (1996). Knowledge Discovery and Data Mining: Towards a Unifying Framework. *Aaai.Org*. https://www.aaai.org/Papers/KDD/1996/KDD96-014.pdf?utm_campaign=ml4devs-newsletter&utm_medium=email&utm_source=Revue newsletter

Feremans, L., Vercruyssen, V., Cule, B., Meert, W., & Goethals, B. (2020). Pattern-Based Anomaly Detection in Mixed-Type Time Series. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11906 LNAI*, 240–256. https://doi.org/10.1007/978-3-030-46150-8_15

Feremans, L., Vercruyssen, V., Meert, W., Cule, B., & Goethals, B. (2020). A framework for pattern mining and anomaly detection in multi-dimensional time series and event logs. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11948 LNAI*, 3–20. https://doi.org/10.1007/978-3-030-48861-1_1

Fernandes, G., Rodrigues, J. J. P. C., Carvalho, L. F., Al-Muhtadi, J. F., & Proença, M. L. (2019). A comprehensive survey on network anomaly detection. *Telecommunication Systems*, *70*(3), 447–489. https://doi.org/10.1007/S11235-018-0475-8

Fillatre, L., Processing, I. N.-I. transactions on signal, & 2007, U. (2005). Non-bayesian detection and detectability of anomalies from a few noisy tomographic projections. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/4063545/?casa_token=4Tqq9TnwvdoAAAAA:dfYLx4yNbF-_pF0nv0OylxdxKe767VDfH2Ui_3wpdeP7fj77zTKkDnhO1ZX4eKhCFgKTVscwa-jTjw

Fillatre, L., Retraint, F., Volumes, I. N.-I. P., & 2005, U. (2005). Detectability of anomalies from a few noisy tomographic projections. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S1474667016361857

Finder, I., Sheetrit, E., & Nissim, N. (2022). A time-interval-based active learning framework for enhanced PE malware acquisition and detection. *Computers and Security*, *121*, 102838. https://doi.org/10.1016/j.cose.2022.102838

Fox, A. J. (1972). Outliers in Time Series. *Journal of the Royal Statistical Society: Series B (Methodological)*, *34*(3), 350–363. https://doi.org/10.1111/J.2517-6161.1972.TB00912.X

Gaddam, S., … V. P.-I. transactions on, & 2007, U. (2007). K-Means+ ID3: A novel method for supervised anomaly detection by cascading K-Means clustering and

ID3 decision tree learning methods. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/4072746/?casa_token=Ln5sQ1n4AGsAAAAA:L43ZGrFlSxHFivEj1u1660qczul1u346Dod6vVFjgTlrvPwCNhzpMzceXmiLz8IYVYbfc62q6FqgUg

Galeano, P., Peña, D., & Tsay, R. S. (2006). Outlier detection in multivariate time series by projection pursuit. *Journal of the American Statistical Association*, *101*(474), 654–669. https://doi.org/10.1198/016214505000001131

Gançarski, P., Dao, T.-B.-H., Crémilleux, B., Forestier, G., & Lampert, T. (2020). Constrained Clustering: Current and New Trends. *A Guided Tour of Artificial Intelligence Research*, 447–484. https://doi.org/10.1007/978-3-030-06167-8_14

García, S., Luengo, J., Systems, F. H.-K.-B., & 2016, U. (2015). Tutorial on practical tips of the most influential data preprocessing algorithms in data mining. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0950705115004785

Gauthama Raman, M. R., Somu, N., Jagarapu, S., Manghnani, T., Selvam, T., Krithivasan, K., & Shankar Sriram, V. S. (2020). An efficient intrusion detection technique based on support vector machine and improved binary gravitational search algorithm. In *Artificial Intelligence Review* (Vol. 53, Issue 5). Springer Netherlands. https://doi.org/10.1007/s10462-019-09762-z

Ghafir, I., Hammoudeh, M., Prenosil, V., Han, L., Hegarty, R., Rabie, K., & Aparicio-Navarro, F. J. (2018). Detection of advanced persistent threat using machine-learning correlation analysis. *Future Generation Computer Systems*, *89*, 349–359. https://doi.org/10.1016/j.future.2018.06.055

Ghafir, I., Kyriakopoulos, K. G., Lambotharan, S., Aparicio-Navarro, F. J., Assadhan, B., Binsalleeh, H., & Diab, D. M. (2019). Hidden markov models and alert correlations for the prediction of advanced persistent threats. *IEEE Access*, *7*, 99508–99520. https://doi.org/10.1109/ACCESS.2019.2930200

Gliozzi, V., Mayor, J., Hu, J. F., & Plunkett, K. (2009). Labels as features (not names) for infant categorization: A neurocomputational approach. *Cognitive Science*, *33*(4), 709–738. https://doi.org/10.1111/J.1551-6709.2009.01026.X

Grayver, E., & Utter, A. (2020). *Extreme Software Defined Radio – GHz in Real Time*. 1–10. https://doi.org/10.1109/aero47225.2020.9172605

Gu, J., Kong, R., Sun, H., Zhuang, H., Pan, F., & Lin, Z. (2022). A novel detection technique based on benign samples and one-class algorithm for malicious PDF documents containing JavaScript. *Spiedigitallibrary.Org*, 62. https://doi.org/10.1117/12.2637518

Guerra-Manzanares, A., Medina-Galindo, J., Bahsi, H., & Nõmm, S. (2020). MedBIoT: Generation of an IoT Botnet Dataset in a Medium-sized IoT Network. *Scitepress.Org*, 207–218. https://doi.org/10.5220/0009187802070218

Gupta, D., Malviya, A., Computer, S. S.-I. J. of, & 2012, undefined. (2012). Performance analysis of classification tree learning algorithms. *Citeseer*, *55*(6), 975–8887. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.244.9217&rep=rep1&type=pdf

Gupta, M., & K. Shrivastava, S. (2015). *Intrusion Detection System based on SVM and Bee Colony*. International Journal of Computer Applications. https://doi.org/10.5120/19576-1377

Gürsakal, N., Yilmaz, F., Analytics, E. U.-. A. in A. D., & 2020, U. (2020). Finding opportunity windows in time series data using the sliding window technique: The

case of stock exchanges. *Ceeol.Com*. https://www.ceeol.com/search/article-detail?id=910386

Haixiang, G., Yijing, L., Shang, J., Applications, G. M.-… with, & 2017, U. (2017). Learning from class-imbalanced data: Review of methods and applications. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0957417416307175

Han, W., Xue, J., Wang, Y., Liu, Z., & Kong, Z. (2019). MalInsight: A systematic profiling based malware detection framework. *Journal of Network and Computer Applications*, *125*, 236–250. https://doi.org/10.1016/j.jnca.2018.10.022

Hardin, J., Analysis, D. R.-C. S. & D., & 2004, undefined. (2004). Outlier detection in the multiple cluster setting using the minimum covariance determinant estimator. *Elsevier*, *44*, 625–638. https://www.sciencedirect.com/science/article/pii/S0167947302002803

Hasan, M., Islam, M., Zarif, M., Things, M. H.-I. of, & 2019, U. (2019). Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S2542660519300241

Hasani, S. R., Othman, Z. A., & Kahaki, S. M. M. (2014). Hybrid feature selection algorithm for intrusion detection system. *Journal of Computer Science*, *10*(6), 1015–1025. https://doi.org/10.3844/jcssp.2014.1015.1025

Hawkins, D. (1980). *Identification of outliers*. https://link.springer.com/content/pdf/10.1007/978-94-015-3994-4.pdf

He, Z., Xu, X., Huang, Z., Information, S. D.-C. S. and, & 2005, U. (2005). FP-outlier: Frequent pattern based outlier detection. *Doiserbia.Nb.Rs*. http://www.doiserbia.nb.rs/Article.aspx?ID=1820-02140501103H

Hejazi, M., Intelligence, Y. S.-A. A., & 2013, undefined. (2013). One-class support vector machines approach to anomaly detection. *Taylor & Francis*, *27*(5), 351–366. https://doi.org/10.1080/08839514.2013.785791

Herløw, L., & Hansen, S. J. (2015). Detection and Prevention of Advanced Persistent Threats. *Imm.Dtu.Dk*. http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/7057/pdf/imm7057.pdf

Heydari, A., Tavakoli, M., Applications, N. S.-E. S. with, & 2016, U. (2016). Detection of fake opinions using time series. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0957417416301129

Hochenbaum, J., Vallis, O. S., & Kejariwal, A. (2017). *Automatic Anomaly Detection in the Cloud Via Statistical Learning*. http://arxiv.org/abs/1704.07706

Hodge, V., review, J. A.-A. intelligence, & 2004, undefined. (2004). A survey of outlier detection methodologies. *Springer*, *22*(2), 85–126. https://doi.org/10.1023/B:AIRE.0000045502.10941.a9

Horng, S. J., Su, M. Y., Chen, Y. H., Kao, T. W., Chen, R. J., Lai, J. L., & Perkasa, C. D. (2011). A novel intrusion detection system based on hierarchical clustering and support vector machines. *Expert Systems with Applications*, *38*(1), 306–313. https://doi.org/10.1016/j.eswa.2010.06.066

Horst, C. (2017). *Predicting Medical Diagnoses from Pharmaceutical Claims Data*. https://digital.lib.washington.edu/researchworks/handle/1773/40423

Hota, H., Handa, R., of, A. S.-I. J., & 2017, undefined. (2017). Time series data prediction using sliding window based RBF neural network. *Ripublication.Com*, *13*(5), 1145–1156. http://www.ripublication.com/ijcir17/ijcirv13n5_46.pdf

Ingre, B., & Yadav, A. (2015). Performance analysis of NSL-KDD dataset using ANN.

International Conference on Signal Processing and Communication Engineering Systems - Proceedings of SPACES 2015, in Association with IEEE, 92–96. https://doi.org/10.1109/SPACES.2015.7058223

Iváncsy, R., Hungarica, I. V.-A. P., & 2006, U. (2006). Frequent pattern mining in web log data. *Citeseer*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.4559&rep=rep1&type=pdf

Jagadale, S. K., Khot, K. V., Bagade, C. S., Mane, R. M., Ghanwat, V. B., Mane, R. K., Mali, S. S., Hong, C. K., & Bhosale, P. N. (2017). Novel synthetic route for the synthesis of ternary Cd(SSe) photoelectrode and their photoelectrochemical application. *Journal of Materials Science: Materials in Electronics*, *28*(3), 2984–2995. https://doi.org/10.1007/s10854-016-5884-4

Jaganathan, K., Panagiotopoulou, S., Cell, J. M.-, & 2019, U. (2019). Predicting splicing from primary sequence with deep learning. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0092867418316295

Jiang, J.-R., Kao, J.-B., Li, Y.-L., Jiang, C. :, Kao, J.-R. ;, & Li, J.-B. ; (2021). Semi-supervised time series anomaly detection based on statistics and deep learning. *Mdpi.Com*. https://doi.org/10.3390/app11156698

Jiang, J., Cyber, L. Y.-2013 I. C. on, & 2013, U. (2013). Anomaly detection via one class svm for protection of scada systems. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/6685663/

Jiang, R., Fei, H., (SensorKDD'10), J. H.-D. from S. D., & 2010, undefined. (2010). Anomaly localization by joint sparse pca in wireless sensor networks. *Core.Ac.Uk*. https://core.ac.uk/download/pdf/52394519.pdf#page=97

Jidiga, G., Conference, P. S.-2014 I. I., & 2014, U. (2014). Anomaly detection using machine learning with a case study. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/7019260/

Joshi, J., Rinal, D., & Patel, J. (2014). Diagnosis and Prognosis Breast Cancer Using. *International Journal of Engineering Research and General Science*, *2*(6), 315–323. www.ijergs.org

Jr, W. M., Algorithms, M. Z.-F. C. and, & 2014, U. (2014). Data mining and analysis. *Academia.Edu*. https://www.academia.edu/download/54692577/Data_Mining_and_Analysis__Fundamental_Concepts_and_Algorithms_.pdf

Kaur, G., Saxena, V., International, J. G.-2010 2nd, & 2010, U. (2010). Anomaly detection in network traffic and role of wavelets. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/5485392/?casa_token=fCjhwMqt-WYAAAAA:-iH6krbk3ApjyjdXQMfUr_QrVxerTvyAnasaf0BTyzQqZStuuyAm6za0OoPsVeKVVVfL1kNkk3I_RA

Kaur, G., Technology, V. S.-I. J. of I., & 2010, U. (2010). An Analysis of Anomaly Detection in Network Traffic and Role of Wavelets. *Search.Ebscohost.Com*. https://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=09732896&asa=Y&AN=48722496&h=q5OPbpw3uPuEi1LApBqcsTOvvE3W%2FEL1bXvcfiUiV71mocgEiq2OPu5M5wTLqSZw2sp6PQXZ5rRCaq%2BwxH4%2FKg%3D%3D&crl=c

Kaveh, A., & Dadras, A. (2018). Structural damage identification using an enhanced thermal exchange optimization algorithm. *Engineering Optimization*, *50*(3), 430–451. https://doi.org/10.1080/0305215X.2017.1318872

Keogh, E., Lonardi, S., … C. R.-D. M. and, & 2007, U. (2007). Compression-based data mining of sequential data. *Springer*. https://link.springer.com/article/10.1007/s10618-006-0049-3

Khan, S., Gani, A., Wahid, A., Kumar Singh, P., Sci Eng, A. J., & Wahid Abdul Wahab, A. (2017). Feature selection of denial-of-service attacks using entropy and granular computing. *Springer*, *43*(2), 499–508. https://doi.org/10.1007/s13369-017-2634-8

Kim, T., Applications, S. C.-E. S. with, & 2018, undefined. (2018). Web traffic anomaly detection using C-LSTM neural networks. *Elsevier*, *106*, 66–76. https://doi.org/10.1016/j.eswa.2018.04.004

Kinnebrew, J., Loretz, K., Mining, G. B.-J. of E. D., & 2013, U. (2013). A contextualized, differential sequence mining method to derive students' learning behavior patterns. *ERIC*. https://eric.ed.gov/?id=EJ1115377

Koushik, A. N. P., Manoj, M., & Nezamuddin, N. (2020). Machine learning applications in activity-travel behaviour research: a review. *Transport Reviews*, *40*(3), 288–311. https://doi.org/10.1080/01441647.2019.1704307

Kraiem, I. Ben, Ghozzi, F., Peninou, A., & Teste, O. (2019). *Pattern-based method for anomaly detection in sensor networks*. https://doi.org/10.5220/0007736701040113

Kuchar, J., Anomaly, V. S.-K. 2017 W. on, & 2018, undefined. (2017). Spotlighting anomalies using frequent patterns. *Proceedings.Mlr.Press*, *71*, 33–42. http://proceedings.mlr.press/v71/kuchar18a.html

Lavicza, Z., Fenyvesi, K., Lieban, D., Park, H., Hohenwarter, M., Mantecon, J. D., & Prodromou, T. (2021). A Novel Method for Detecting APT Attacks by Using OODA Loop and Black Swan Theory. *Business and Society*, *60*(2), 420–453.

Laxhammar, R., Pattern, G. F.-I. transactions on, & 2013, U. (2013). Online learning and sequential anomaly detection in trajectories. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/6598676/?casa_token=dpl-ebLVA6cAAAAA:sfAWLh-9ceXqkbeNSfcipher7okeraCzKaV6BJHEzYWZCk6gskRXrm-CDxqN1TC5xBbuHCYCN1CSPg

Li, P., Yang, X., Xiong, Q., Wen, J., & Tang, Y. Y. (2018). Defending against the Advanced Persistent Threat: An Optimal Control Approach. *Security and Communication Networks*, *2018*. https://doi.org/10.1155/2018/2975376

Li, X., & Jiang, H. (2017). Artificial intelligence technology & engineering applications. *Applied Computational Electromagnetics Society Journal*, *32*(5), 381–388.

Li, Y., Dai, W., Bai, J., Gan, X., Wang, J., & Wang, X. (2019). An Intelligence-Driven Security-Aware Defense Mechanism for Advanced Persistent Threats. *IEEE Transactions on Information Forensics and Security*, *14*(3), 646–661. https://doi.org/10.1109/TIFS.2018.2847671

Li, Z., Liu, P., Wang, W., Prevention, C. X.-A. A. &, & 2012, undefined. (n.d.). Using support vector machine models for crash injury severity analysis. *Elsevier*. Retrieved July 4, 2022, from https://www.sciencedirect.com/science/article/pii/S0001457511002363

Liang, N., Yang, Z., Li, Z., Sun, W., Systems, S. X.-K.-B., & 2020, U. (2020). Multi-view clustering by non-negative matrix factorization with co-orthogonal constraints. *Elsevier*.

https://www.sciencedirect.com/science/article/pii/S0950705120300642

Lin, J, Keogh, E., visualization, S. L.-I., & 2005, undefined. (1997). Visualizing and discovering non-trivial patterns in large time series databases. *Journals.Sagepub.Com*, *4*(2), 61–82. https://doi.org/10.1057/palgrave.ivs.9500089

Lin, Jessica, Keogh, E., Lonardi, S., Lankford, J. P., & Nystrom, D. M. (2004). Visually mining and monitoring massive time series. *KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 460–469. https://doi.org/10.1145/1014052.1014104

Ling, C. X., Huang, J., & Zhang, H. (2006). An introduction to ROC analysis. *Elsevier*, *2006*, 861–874. https://www.sciencedirect.com/science/article/pii/S016786550500303X

Liu, C. C., & Chen, F. C. (1993). Adaptive control of non-linear continuous-time systems using neural networks—general relative degree and MIMO cases. *International Journal of Control*, *58*(2), 317–335. https://doi.org/10.1080/00207179308923005

Long, M., Wang, J., Ding, G., Shen, D., & Yang, Q. (2014). Transfer learning with graph co-regularization. *IEEE Transactions on Knowledge and Data Engineering*, *26*(7), 1805–1818. https://doi.org/10.1109/TKDE.2013.97

Lopez, J., Hernández, S., … A. U.-C. &, & 2021, U. (2021). Effect of missing data on short time series and their application in the characterization of surface temperature by detrended fluctuation analysis. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0098300421000960

Luo, Y., Xiao, Y., Cheng, L., Peng, G., & Yao, D. D. (2021). Deep Learning-based Anomaly Detection in Cyber-physical Systems: Progress and Opportunities. *ACM Computing Surveys*, *54*(5). https://doi.org/10.1145/3453155

Ma, M., Zhang, S., Chen, J., Xu, J., Li, H., Lin, Y., Nie, X., Zhou, B., Wang, Y., & Pei, D. (2021). Jump-starting multivariate time series anomaly detection for online service systems. *2021 USENIX Annual Technical Conference*, 413–426. https://www.usenix.org/conference/atc21/presentation/ma

Ma, X., Wu, J., Member, S., Xue, S., Yang, J., Zhou Quan Sheng, C. Z., Xiong, H., & Akoglu, L. (2021). A Comprehensive Survey on Graph Anomaly Detection with Deep Learning. *IEEE Transactions on Knowledge and Data Engineering*, 1. https://doi.org/10.1109/TKDE.2021.3118815

Mahbub, U., Komulainen, J., … D. F.-I. T. on, & 2019, U. (2019). Continuous authentication of smartphones based on application usage. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/8721521/?casa_token=vywRyYv_0YoAAAAA:OqiKaZT1KAaHiuKxtA-kNwUMs0m_mbGJlidqWhUy7p00L7KeWHdg-hjjLYTtA8oUXcj11ln5YhRXYw

Manimurugan, S., Al-Mutairi, S., … M. A.-I., & 2020, U. (2020). Effective attack detection in internet of medical things smart environment using a deep belief neural network. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/9057709/

Manoj Gadiyar, H. T., Goudar, K., S, T. G., & Jeevan Kurdekar, M. (2016). DETECTING ANOMALOUS ACTIVITIES IN SPATIO-TEMPORAL REGION. *International Journal of Engineering Applied Sciences and Technology*, *6*, 441–444. https://ijeast.com/papers/441-444,Tesma601,IJEAST.pdf

Marchetti, M., Guido, A., Pierazzi, F., & Colajanni, M. (2016). Countering Advanced Persistent Threats through security intelligence and big data analytics. *International Conference on Cyber Conflict, CYCON*, *2016-Augus*, 243–261. https://doi.org/10.1109/CYCON.2016.7529438

Martos, G., Hernández, N., Muñoz, A., Entropy, J. M.-, & 2018, U. (2018). Entropy measures for stochastic processes with applications in functional anomaly detection. *Mdpi.Com*. https://www.mdpi.com/252676

Mehrotra, M., Sci, N. J.-I. J. E., & 2017, U. (2017). Anomaly Detection in Temporal data Using Kmeans Clustering with C5. 0. *Academia.Edu*. https://www.academia.edu/download/53418848/L0605017781.pdf

Mijalkovic, J., & Spognardi, A. (2022). Reducing the False Negative Rate in Deep Learning Based Network Intrusion Detection Systems. *Algorithms*, *15*(8). https://doi.org/10.3390/a15080258

Mohan, C., OF, K. M.-I. J., & 2017, U. (2017). Anomaly detection in banking operations. *Idrbt.Ac.In*. https://idrbt.ac.in/IssueNo.1_Sep2017.pdf#page=24

Mounce, S., Mounce, R., hydroinformatics, J. B.-J. of, & 2011, undefined. (2011). Novelty detection for time series data analysis in water distribution systems using support vector machines. *Iwaponline.Com*. https://doi.org/10.2166/hydro.2010.144

Munir, A., Kansakar, P., Electronics, S. K.-I. C., & 2017, U. (2017). IFCIoT: Integrated Fog Cloud IoT: A novel architectural paradigm for the future Internet of Things. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/7948854/?casa_token=EOlioljRfqoAAAAA:KsTy0GnSla74WuEgGPsoc5DhfFNs8UroewGunvYm-ZSuFFKHBzwyvHf_ijYT-X_AL1cWt6wPRm86

Munivara Prasad, K., Rama Mohan Reddy, A., & Venu Gopal Rao, K. (2018). An experiential metrics-based machine learning approach for anomaly based real time prevention (ARTP) of App-DDoS attacks on web. *Advances in Intelligent Systems and Computing*, *668*, 99–112. https://doi.org/10.1007/978-981-10-7868-2_10

Muniyandi, A., Rajeswari, R., Engineering, R. R.-P., & 2012, U. (2012). Network anomaly detection by cascading k-Means clustering and C4. 5 decision tree algorithm. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S1877705812008594

Münz, G., Li, S., MMBnet, G. C.-G. W., & 2007, U. (2007). Traffic anomaly detection using k-means clustering. *Net.in.Tum.De*. https://www.net.in.tum.de/projects/dfg-lupus/files/muenz07k-means.pdf

Neuschmied, H., Winter, M., Stojanović, B., Hofer-Schmitz, K., Božić, J., & Kleb, U. (2022). APT-Attack Detection Based on Multi-Stage Autoencoders. *Applied Sciences (Switzerland)*, *12*(13), 1–18. https://doi.org/10.3390/app12136816

Olson, L., Eckert, Y., S Manne - US Patent 9, 524,164, & 2016, undefined. (2016). Specialized memory disambiguation mechanisms for different memory read access types. In *Google Patents*. https://patents.google.com/patent/US9524164B2/en

Otey, M. E., Ghoting, A., & Parthasarathy, S. (2006). Fast distributed outlier detection in mixed-attribute data sets. *Data Mining and Knowledge Discovery*, *12*(2–3), 203–228. https://doi.org/10.1007/S10618-005-0014-6

Owido, P., Onyuma, S., Finance, G. O.-R. J. of, & 2013, U. (2013). A GARCH approach to measuring efficiency: a case study of Nairobi securities exchange.

Researchgate.Net. https://www.researchgate.net/profile/Samuel-Onyuma/publication/275581274_A_Garch_Approach_to_Measuring_Efficiency_A_Case_Study_of_Nairobi_Securities_Exchange/links/553f9f470cf2736761c03fc3/A-Garch-Approach-to-Measuring-Efficiency-A-Case-Study-of-Nairobi-S

Ozyildirim, B., Systems, M. A.-2018 I. in I., & 2018, U. (2018). Handwritten Digits Classification with Generalized Classifier Neural Network. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/8553999/

Paganoni, S., Berry, J. D., Quintana, M., Macklin, E., Saville, B. R., Detry, M. A., Chase, M., Sherman, A. V., Yu, H., Drake, K., Andrews, J., Shefner, J., Chibnik, L. B., Vestrucci, M., Cudkowicz, M. E., Macklin, E., Quintana, M., Saville, B., Detry, M. A., … Torti, M. (2022). Adaptive Platform Trials to Transform Amyotrophic Lateral Sclerosis Therapy Development. *Annals of Neurology*, *91*(2), 165–175. https://doi.org/10.1002/ana.26285

Palmieri, F., Fiore, U., & Castiglione, A. (2014). A distributed approach to network anomaly detection based on independent component analysis. *Concurrency and Computation: Practice and Experience*, *26*(5), 1113–1129. https://doi.org/10.1002/CPE.3061

Pang, G., Cao, L., & Chen, L. (2016). *Outlier detection in complex categorical data by modeling the feature value couplings*. https://ink.library.smu.edu.sg/sis_research/7146/

Patil, P. (2016). Research in Computer Applications and Robotics Recommended Artificial. *INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS Www.Ijrcar.Com*, *4*(8), 16–19.

Pérez, A. G. (2021). *Automatic detection of anomalies in the wire fence*. https://oa.upm.es/id/eprint/66165

Perez, D., Astor, M. A., Abreu, D. P., & Scalise, E. (2017). Intrusion detection in computer networks using hybrid machine learning techniques. *2017 43rd Latin American Computer Conference, CLEI 2017*, *2017-Janua*(8), 1–10. https://doi.org/10.1109/CLEI.2017.8226392

Poola, I. (2017). The Best of the Machine Learning Algorithms Used in Artificial Intelligence. *International Journal of Advanced Research in Computer and Communication Engineering*, *6*(10), 187–194. https://doi.org/10.17148/IJARCCE.2017.61032

Popescu, V., Nedevschi, S., Danescu, R., & Marita, T. (2015). A Lane Assessment Method Using Visual Information Based on a Dynamic Bayesian Network. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, *19*(3), 225–239. https://doi.org/10.1080/15472450.2013.856724

Prasad, K. M., Reddy, A. R. M., & Rao, K. V. (2020). BARTD: Bio-inspired anomaly based real time detection of under rated App-DDoS attack on web. *Journal of King Saud University - Computer and Information Sciences*, *32*(1), 73–87. https://doi.org/10.1016/j.jksuci.2017.07.004

Prodanovic, V., Chan, H. W., Mane, A. U., Elam, J. W., Graaf, H. V.D., & Sarro, P. M. (2017). Ultra-thin ALD MGO membranes as mems transmission dynodes in a timed photon counter. *Proceedings of the IEEE International Conference on Micro Electro Mechanical Systems (MEMS)*, 740–743. https://doi.org/10.1109/MEMSYS.2017.7863514

Rad, B. B., Nejad, M. K. H., & Shahpasand, M. (2018). Malware classification and detection using artificial neural network. *Journal of Engineering Science and Technology*, *13*(Special Issue on ICCSIT 2018), 14–23.

Rakha, T., Construction, A. G.-A. in, & 2018, undefined. (2018). Review of Unmanned Aerial System (UAS) applications in the built environment: Towards automated building inspection procedures using drones. *Elsevier*. https://doi.org/10.1016/j.autcon.2018.05.002

Rashidi, L., Hashemi, S., Artificial, A. H.-I. C. on, & 2011, undefined. (2011). Anomaly detection in categorical datasets using bayesian networks. *Springer*, *7003 LNAI*(PART 2), 610–619. https://doi.org/10.1007/978-3-642-23887-1_78

Ratnayake, R., Computer, H. U.-M. J. on, & 2020, U. (2020). A Novel Hybrid Approach for Network Intrusion Detection using Extreme Gradient Boosting and Long Short-Term Memory Networks. *Search.Proquest.Com*. https://search.proquest.com/openview/d0bf0e4637ff640fe5c102a429999eb7/1?pq-origsite=gscholar&cbl=2037358

Ren, H., Xu, B., Wang, Y., Yi, C., Huang, C., Kou, X., Xing, T., Yang, M., Tong, J., Zhang, Q., & Kou, X.-A. (2019). Time-series anomaly detection service at microsoft. *Dl.Acm.Org*, *19*, 3009–3017. https://doi.org/10.1145/3292500.3330680

Riedel, S., Yao, L., & McCallum, A. (2010). Modeling relations and their mentions without labeled text. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6323 LNAI*(PART 3), 148–163. https://doi.org/10.1007/978-3-642-15939-8_10

Rosner, B. (1983). Percentage points for a generalized esd many-outlier procedure. *Technometrics*, *25*(2), 165–172. https://doi.org/10.1080/00401706.1983.10487848

Rubinstein, B. I. P., Nelson, B., Huang, L., Joseph, A. D., Lau, S.-H., Rao, S., Taft, N., & Tygar, J. D. (2016). PCA-based multivariate statistical network monitoring for anomaly detection. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0167404816300116

S, A. T., Nagakishore, S. B., & Swetha, A. (2012). Design , ASIC Implementation and Verification of Synchronous and Asynchronous FIFO. *Proceedings of International Conference on Innovation in Electronics and Communications Engineering*.

Saarela, M., Yener, B., Zaki, M., and, T. K.-J. W., & 2016, undefined. (2016). Predicting math performance from raw large-scale educational assessments data: a machine learning approach. *Jyx.Jyu.Fi*, 48. https://jyx.jyu.fi/handle/123456789/52562

Sahabandu, D., Moothedath, S., Bushnell, L., Poovendran, R., Aller, J., Lee, W., & Clark, A. (2019). A game theoretic approach for dynamic information flow tracking with conditional branching. *Proceedings of the American Control Conference*, *2019-July*, 2289–2296. https://doi.org/10.23919/acc.2019.8814596

Salama, M. A., Eid, H. F., Ramadan, R. A., Darwish, A., & Hassanien, A. E. (2011). Hybrid intelligent intrusion detection scheme. *Advances in Intelligent and Soft Computing*, *96 AISC*, 293–303. https://doi.org/10.1007/978-3-642-20505-7_26

Salcedo-Sanz, S., Ghamisi, P., Piles, M., … M. W.-I., & 2020, U. (2020). Machine learning information fusion in Earth observation: A comprehensive review of methods, applications and data sources. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S1566253520303171

Salvador, S., Chan, P., & Brodie, J. (2004). Learning states and rules for time series anomaly detection. *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004*, *1*, 306–311. https://www.aaai.org/Papers/FLAIRS/2004/Flairs04-055.pdf

Savage, D., Zhang, X., Yu, X., Chou, P., networks, Q. W.-S., & 2014, undefined. (2016). Anomaly detection in online social networks. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0378873314000331

Scarpiniti, M., Colasante, F., Di Tanna, S., Ciancia, M., Lee, Y. C., & Uncini, A. (2021). Deep Belief Network based audio classification for construction sites monitoring. *Expert Systems with Applications*, *177*. https://doi.org/10.1016/j.eswa.2021.114839

Schenker, A., Last, M., … H. B.-… C. on D., & 2003, U. (2003). Classification of web documents using a graph model. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/1227666/?casa_token=i-H5LsFYQp4AAAAA:rTGj0bHrA5RpNkf3ryNfHiSH6Zc69oUm35I2a2fuUkMB7Tq50niGua4AcjymqBEK9wQy7ukbvuDTIg

Schmidhuber, J. (2015). Deep Learning in neural networks: An overview. *Neural Networks*, *61*, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Sebestyen, G., Hangan, A., … Z. C.-2018 I. I., & 2018, U. (2018). A taxonomy and platform for anomaly detection. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/8402710/?casa_token=EV4DghlgfO8AAAAA:MZmhEQgJFrvbyfERKkq7sQmWWUVPiXfK9Xtkr9ZVfI0dxlWCgoUydbGTBlNkHCfociAZxCeN2tvUzw

Sevil, H. E. (2020). Anomaly detection using parity space approach in team of uavs with entropy based distributed behavior. *AIAA Scitech 2020 Forum*, *1 PartF*. https://doi.org/10.2514/6.2020-1625

Sharma, A., Jain, A., Gupta, P., Access, V. C.-I., & 2020, U. (2020). Machine learning applications for precision agriculture: A comprehensive review. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/9311735/

Shaukat, K., Luo, S., Varadharajan, V., Hameed, I. A., Chen, S., Liu, D., & Li, J. (2020). Performance comparison and current challenges of using machine learning techniques in cybersecurity. *Mdpi.Com*. https://doi.org/10.3390/en13102509

Shenwen, L., Yingbo, L., & Xiongjie, D. (2015). Study and research of APT detection technology based on big data processing architecture. *ICEIEC 2015 - Proceedings of 2015 IEEE 5th International Conference on Electronics Information and Emergency Communication*, *2012*, 313–316. https://doi.org/10.1109/ICEIEC.2015.7284547

Shi, F., Chen, L., Han, J., & Childs, P. (2017). A Data-Driven Text Mining and Semantic Network Analysis for Design Information Retrieval. *Journal of Mechanical Design, Transactions of the ASME*, *139*(11). https://doi.org/10.1115/1.4037649

Shipmon, D., Gurevitch, J., … P. P. preprint arXiv, & 2017, U. (2017). Time series anomaly detection; detection of anomalous drops with limited features and sparse examples in noisy highly periodic data. *Arxiv.Org*. https://arxiv.org/abs/1708.03665

Shipmon, D. T., Gurevitch, J. M., Piselli, P. M., & Edwards, S. (2017). *Time series anomaly detection: Detection of anomalous drops with limited features and sparse examples in noisy periodic data*. https://research.google/pubs/pub46283/

Siddiqui, S., Khan, M. S., Ferens, K., & Kinsner, W. (2016). Detecting advanced persistent threats using fractal dimension based machine learning classification. *IWSPA 2016 - Proceedings of the 2016 ACM International Workshop on Security and Privacy Analytics, Co-Located with CODASPY 2016*, 64–69. https://doi.org/10.1145/2875475.2875484

Singh, K., Guntuku, S., Thakur, A., Sciences, C. H.-I., & 2014, U. (2014). Big data analytics framework for peer-to-peer botnet detection using random forests. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0020025514003570

Singh, P., Krishnamoorthy, S., Nayyar, A., Luhach, A. K., & Kaur, A. (2019). Soft-computing-based false alarm reduction for hierarchical data of intrusion detection system. *Journals.Sagepub.Com*, *15*(10). https://doi.org/10.1177/1550147719883132

Singh, R., Kumar, H., & Singla, R. K. (2015). An intrusion detection system using network traffic profiling and online sequential extreme learning machine. *Expert Systems with Applications*, *42*(22), 8609–8624. https://doi.org/10.1016/j.eswa.2015.07.015

Singh, S., Sharma, P. K., Moon, S. Y., Moon, D., & Park, J. H. (2019). A comprehensive study on APT attacks and countermeasures for future networks and communications: challenges and solutions. *Journal of Supercomputing*, *75*(8), 4543–4574. https://doi.org/10.1007/s11227-016-1850-4

Society, C. T.-J. of the O. R., & 2015, U. (2015). A better measure of relative prediction accuracy for model selection and model estimation. *Springer*. https://link.springer.com/article/10.1057/jors.2014.103

Tao, X., Wang, R., Chang, R., Li, C., Liu, R., Systems, J. Z.-K.-B., & 2019, U. (2019). Spectral clustering algorithm using density-sensitive distance measure with global and local consistencies. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0950705119300310

Tatbul, N., Lee, T., Zdonik, S., … M. A.-A. in neural, & 2018, U. (2018). Precision and recall for time series. *Proceedings.Neurips.Cc*. https://proceedings.neurips.cc/paper/7462-precision-and-recall-for-time-series

Thill, M., Däubener, S., 2019), W. K.-… (ITAT, & 2019, U. (2019). Anomaly detection in electrocardiogram readings with stacked LSTM networks. *Scholarlypublications* …. https://scholarlypublications.universiteitleiden.nl/handle/1887/85757

Tornai, K., Olah, A., Alm´an, K. ´, Tornai, A. A., Andr´, A., Ol´ah, A., Ol´ah, O., & Levendovszky, J. ´A. (2016). Monitoring Scheme for Event and Danger Detection in Wireless Sensor Networks. *Researchgate.Net*. https://www.researchgate.net/profile/Kalman-Tornai/publication/291355271_Monitoring_Scheme_for_Event_and_Danger_Detection_in_Wireless_Sensor_Networks/links/58ba889792851c471d4f5cac/Monitoring-Scheme-for-Event-and-Danger-Detection-in-Wireless-Sensor-Networ

Tran, K., Nguyen, H. Du, IFAC-PapersOnLine, S. T.-, & 2019, undefined. (2019). Anomaly detection using long short term memory networks and its applications in supply chain management. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S240589631931554X

Trifonov, R., Manolov, S., Yoshinov, R., Tsochev, G., & Pavlova, G. (2017). Artificial Intelligence Methods for Cyber Threats Intelligence. *International Journal of Computers*, *2*, 129–135. https://www.iaras.org/iaras/filedownloads/ijc/2017/006-0020(2017).pdf

Triguero, I., Del Río, S., López, V., Bacardit, J., Benítez, J. M., & Herrera, F. (2016). ROSEFW-RF: the winner algorithm for the ECBDL'14 big data competition: an extremely imbalanced big data bioinformatics problem. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0950705115002130

Tsay, R., Pena, D., Biometrika, A. P.-, & 2000, undefined. (1998). Outliers in

multivariate time series. *Academic.Oup.Com*. https://academic.oup.com/biomet/article-abstract/87/4/789/232743

Tsay, R. S. (1988). Outliers, level shifts, and variance changes in time series. *Journal of Forecasting*, *7*(1), 1–20. https://doi.org/10.1002/FOR.3980070102

Tsehay, Y. (2019). *Social Media Mining Algorithms Big Data Analysts Need to Improve Infectious Disease Predictive Model for Malaria in Ethiopia*. https://search.proquest.com/openview/d06b6f895cc6f20b372bd2c80458fe99/1?pq-origsite=gscholar&cbl=18750&diss=y

Tuor, A., Baerwolf, R., Knowles, N., … B. H.-W. at the thirty, & 2018, U. (2018). Recurrent neural network language models for open vocabulary event-level cyber anomaly detection. *Aaai.Org*. https://www.aaai.org/ocs/index.php/WS/AAAIW18/paper/view/17039/15579

Ussath, M., Jaeger, D., Cheng, F., & Meinel, C. (2016). Advanced persistent threats: Behind the scenes. *2016 50th Annual Conference on Information Systems and Sciences, CISS 2016*, 181–186. https://doi.org/10.1109/CISS.2016.7460498

Vangipuram, R., Gunupudi, R. K., Puligadda, V. K., & Vinjamuri, J. (2020). A machine learning approach for imputation and anomaly detection in IoT environment. *Expert Systems*, *37*(5). https://doi.org/10.1111/EXSY.12556

Velarde-Alvarado, P., Vargas-Rosales, C., Martinez-Pelaez, R., Toral-Cruz, H., Martinez-Herrera, A. F., & Martinez-Herrera afmartinezphdmty, A. F. (2016). An unsupervised approach for traffic trace sanitization based on the entropy spaces. *Springer*, *61*(3), 609–626. https://doi.org/10.1007/s11235-015-0017-6

Wang, X., & Wang, C. (2020a). Time Series Data Cleaning with Regular and Irregular Time Intervals. *Arxiv.Org*. http://arxiv.org/abs/2004.08284

Wang, X., & Wang, C. (2020b). *Time Series Data Cleaning with Regular and Irregular Time Intervals*. http://arxiv.org/abs/2004.08284

Wang, Yu, Miao, Q., Ma, E. W. M., Tsui, K. L., & Pecht, M. G. (2013). Online anomaly detection for hard disk drives based on mahalanobis distance. *IEEE Transactions on Reliability*, *62*(1), 136–145. https://doi.org/10.1109/TR.2013.2241204

Wang, Yuan, Zhang, D., Liu, Y., Dai, B., & Lee, L. H. (2019). Enhancing transportation systems via deep learning: A survey. *Transportation Research Part C: Emerging Technologies*, *99*, 144–163. https://doi.org/10.1016/j.trc.2018.12.004

Wang, Z., Inference, R. M.-J. of S. P. and, & 2020, U. (2020). Model-free posterior inference on the area under the receiver operating characteristic curve. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0378375820300379

Weems, C. F., Ahmed, I., Richard, G. G., Russell, J. D., & Neill, E. L. (2018). Susceptibility and resilience to cyber threat: Findings from a scenario decision program to measure secure and insecure computing behavior. *PLoS ONE*, *13*(12), 1–18. https://doi.org/10.1371/journal.pone.0207408

Wichlund, S. (2012). Software Defined Radio Prototyping with Visual C++ Express and Code Composer Studio. *Wireless Engineering and Technology*, *03*(02), 52–62. https://doi.org/10.4236/wet.2012.32009

Wu, Z., Liu, S., Zhao, D., Yang, L., Xu, Z., Yang, Z., Zhou, W., He, H., Huang, M., Liu, D., Li, R., & Ding, D. (2020). Neural Network Classification of Ice-Crystal Images Observed by an Airborne Cloud Imaging Probe. *Atmosphere - Ocean*, *58*(5), 303–315. https://doi.org/10.1080/07055900.2020.1843393

Xing, K., Li, A., Jiang, R., International, Y. J.-2020 I. F., & 2020, U. (2020). A Review of APT Attack Detection Methods and Defense Strategies. *Ieeexplore.Ieee.Org*.

https://ieeexplore.ieee.org/abstract/document/9172866/

Xiong, C., Li, Z., Chen, Y., Zhu, T., Wang, J., Yang, H., & Ruan, W. (2022). Generic, efficient, and effective deobfuscation and semantic-aware attack detection for PowerShell scripts. *Frontiers of Information Technology and Electronic Engineering*, *23*(3), 361–381. https://doi.org/10.1631/FITEE.2000436

Yan, G., Li, Q., Guo, D., & Meng, X. (2020a). Discovering suspicious APT behaviors by analyzing DNS activities. *Sensors (Switzerland)*, *20*(3). https://doi.org/10.3390/s20030731

Yan, G., Li, Q., Guo, D., & Meng, X. (2020b). Discovering suspicious APT behaviors by analyzing DNS activities. *Sensors (Switzerland)*, *20*(3), 1–17. https://doi.org/10.3390/s20030731

Yao, D. (Daphne), Shu, X., Cheng, L., & Stolfo, S. J. (2017). Anomaly Detection as a Service: Challenges, Advances, and Opportunities. *Synthesis Lectures on Information Security, Privacy, and Trust*, *9*(3), 1–173. https://doi.org/10.2200/S00800ED1V01Y201709SPT022

Yao, S. J., Song, Y. H., Zhang, L. Z., & Cheng, X. Y. (2000). Wavelet transform and neural networks for short-term electrical load forecasting. *Energy Conversion and Management*, *41*(18), 1975–1988. https://doi.org/10.1016/S0196-8904(00)00035-2

Yeh, C., Herle, H. Van, 16th, E. K.-2016 I., & 2016, U. (2016). Matrix profile III: the matrix profile allows visualization of salient subsequences in massive time series. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/7837882/?casa_token=RvmSSwv2e_gAAAAA:Cy3iwcagNtiaqnzBiX35Pyz1k738HEU1gDBJ8Xp0Dq37cGPQDCII7jZsnRhdnFI3Ix2JWQQZtVYxHg

Young, H., Vliet, T. van, Ven, J. van de, & Jol, S. (2017). Understanding Human Factors in Cyber Security. *Springer*, *1*(October 2017), 244–254. https://doi.org/10.1007/978-3-319-60585-2

Yu, K., Lin, T., Ma, H., Li, X., Processing, X. L.-M. S. and S., & 2021, U. (2021). A multi-stage semi-supervised learning approach for intelligent fault diagnosis of rolling bearing using data augmentation and metric learning. *Elsevier*. https://www.sciencedirect.com/science/article/pii/S0888327020304295

Zarzour, H., … M. A.-A.-… C. on S., & 2020, U. (2020). A convolutional neural network-based reviews classification method for explainable recommendations. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/9336529/?casa_token=wu7mC9gzBVEAAAAA:llGKeNGSbED2cQ4iMoA-aXzN5vmHdCw49gFHuk79OiFqervy0zUqaciLPoSr7hAaixMsRBhf_aYpzQ

Zhang, H., Li, Y., Lv, Z., Sangaiah, A. K., & Huang, T. (2020). A real-Time and ubiquitous network attack detection based on deep belief network and support vector machine. *IEEE/CAA Journal of Automatica Sinica*, *7*(3), 790–799. https://doi.org/10.1109/JAS.2020.1003099

Zhang, J., Wang, F., Wang, K., … W. L.-I. T. on, & 2011, U. (2011). Data-driven intelligent transportation systems: A survey. *Ieeexplore.Ieee.Org*. https://ieeexplore.ieee.org/abstract/document/5959985/?casa_token=vOBkpByY-ugAAAAA:4TEdUpIkmlfibAEuWEuNDNXDRrzTvKsOHIc0GVJx6wpJd-h7L8fGVmiClxnfQbpjYk46iizwaGjiCw

Zhang, R., Huo, Y., Liu, J., Communication, F. W.-S. and, & 2017, U. (2017). Constructing APT attack scenarios based on intrusion kill chain and fuzzy

clustering. *Hindawi.Com*. https://www.hindawi.com/journals/scn/2017/7536381/

Zhang, X, Zhuo, Y., Luo, Q., Wu, Z., Midya, R., … Z. W.-N., & 2020, U. (2020). An artificial spiking afferent nerve based on Mott memristors for neurorobotics. *Nature.Com*. https://www.nature.com/articles/s41467-019-13827-6

Zhang, Xu, Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., Xie, C., Yang, X., Cheng, Q., Li, Z., Chen, J., He, X., Yao, R., Lou, J. G., Chintalapati, M., Shen, F., & Zhang, D. (2019). Robust log-based anomaly detection on unstable log data. *ESEC/FSE 2019 - Proceedings of the 2019 27th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, *19*, 807–817. https://doi.org/10.1145/3338906.3338931

Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., … A. G. &, & 2013, undefined. (2017). Botnet detection based on traffic behavior analysis and flow intervals. *Elsevier*. https://doi.org/10.1016/j.cose.2013.04.007

Zhao, M. J., Driscoll, A. R., Sengupta, S., Fricker, R. D., Spitzner, D. J., & Woodall, W. H. (2018). Performance evaluation of social network anomaly detection using a moving window–based scan method. *Quality and Reliability Engineering International*, *34*(8), 1699–1716. https://doi.org/10.1002/qre.2364

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, *115*, 213–237. https://doi.org/10.1016/j.ymssp.2018.05.050