REPUBLIC OF CAMEROON
PEACE-WORK-FATHERLAND
****
UNIVERSITY OF DSCHANG
****
POST GRADUATE SCHOOL

RÉPUBLIQUE DU CAMEROUN
PAIX-TRAVAIL-PATRIE
****
UNIVERSITÉ DE DSCHANG
****
ÉCOLE DOCTORALE

DSCHANG SCHOOL OF SCIENCE AND TECHNOLOGY

FUNDAMENTAL COMPUTER SCIENCE, ENGINEERING, AND APPLICATIONS RESEARCH UNIT (URIFIA)

TOPIC :

MACHINE LEARNING BASED ANOMALY DETECTION FOR DNS TRANSACTIONS

Thesis written for a public defense of a Masters degree in Computer Science

Option : Computer Science

Specialty : Artificial Intelligence

By

CHAH Promise ngang

Registration Number : CM-UDS-CM-UDS-19SCI0237

Master of Science in Computer Science

SUPERVISOR

Pr KENGNE Vianney .T

Associate Professor Professor

Year 2024

DECLARATION OF AUTHORSHIP

CERTIFICATION

I, the undersigned, CHAH Promise ngang, a student in Fundamental Computer Science, Registration Number CM-UDS-19SCI0237, option Artificial Intelligence, Department of Mathematics and Computer Science, Faculty of Sciences of the University of Dschang, certify that this dissertation entitled DNS abnomal attack detection using Machine Learning approach carried out at the Fundamental Computer Science, Engineering and Applications Research Unit (URIFIA) under the supervision of Pr Kengne vianney .T, associate Proffessor of the University of Dschang (Faculty of Sciences) is the fruit of my own research work. This dissertation is authentic and has never been presented for the award of any University Diploma or Degree.

AUTHOR

CHAH Promise ngang

SUPERVISOR

Pr Kengne vianney .T
(Associate Professor)
HEAD OF DEPARTMENT

Pr NKENLIFACK Marcellin

(Associate Professor)

DEDICATIONS

TO my father CHAH JOHNSON TOYANG *and my mother* KAMDOUM ANNETTE CHANTAL.

ACKNOWLEDGEMENTS

The achievements presented in this report would not be possible without God and the precious assistance of many people. Among these people I would like to particularly thank:

- Pr KENGNE TCHENDJI Vianney and Mr Tangie yaweni my supervisor for their invaluable patience and feedback.They always supported and oriented me in the right direction. Their comments and suggestions were very helpful in the realization of this work;
- The teaching staff of the Department of Mathematics and Computer Science. They taught me the fundamentals which allowed me to increase my knowledge;
- The jury members of this report for taking the time to give useful advice that I gladly embrace for the improvement of this work;
- My parents and siblings. I would be remiss in not mentioning my family for the physical, emotional, moral and financial support whatever the circumstances. They were smart enough to remind me of my goals when I let myself down in ease;
- I also express my thanks to my academic elders for their precious advice and help;
- I would like to express my sincere thanks to my classmates, particularly to Tiyo melong Rivaldes for the many exchanges we had in order to improve this work;
- Those whose name are not here and who contributed to this work.

TABLE OF CONTENTS

LIST OF SYMBOLS

$\mathbb{DNS}$	Domain Name System;
DBSCAN	Density based scan.
DNS	Domain name server.

List of Acronyms

P2P	Peer to Peer;
BPMN	Business Process Model and Notation.

List of Tables

List of Figures

RÉSUMÉ

Dans le domaine de la cybersécurité, le système de noms de domaine (DNS) est crucial pour traduire les noms de domaine lisibles par l'homme en adresses IP, mais son rôle essentiel en fait également une cible fréquente des cyberattaques. Les modèles de classification binaire traditionnels, qui classent le trafic DNS comme « bénin » ou « malveillant », sont limités dans leur capacité à fournir des renseignements détaillés sur les menaces, nécessaires à une atténuation efficace des menaces. Ce travail vise à résoudre ces limitations en utilisant les techniques Ml pour transformer le problème de classification binaire en un problème de classification multi-classes, permettant ainsi une détection plus nuancée de divers types d'attaques basées sur le DNS. Les principaux objectifs de cette recherche sont de prétraiter l'ensemble de données DNS, de mener une ingénierie approfondie des fonctionnalités grâce à une analyse de corrélation, d'appliquer des algorithmes d'apprentissage automatique (ML) tels que des algorithmes de clustering (DBSCAN) pour créer de nouvelles étiquettes de classe et de développer des modèles de classification multi-classes. Les modèles sont rigoureusement évalués à l'aide de mesures telles que l'exactitude, la précision, le rappel et le score F1. Ce modèle pourrait ensuite être déployé dans un environnement réel pour garantir son efficacité. Nos contributions incluent l'amélioration de la granularité de la classification du trafic DNS, l'amélioration des processus d'ingénierie des fonctionnalités et l'avancement de la recherche sur la cybersécurité. En tirant parti de techniques de clustering avancées et en créant de nouvelles étiquettes de classe, notre approche fournit des renseignements détaillés sur les menaces, permettant ainsi des stratégies de réponse plus efficaces et ciblées. De plus, en mettant nos méthodologies et nos résultats à disposition sous forme de ressources open source, nous favorisons la collaboration et accélérons le développement de solutions de cybersécurité efficaces. En conclusion, cette recherche transforme la classification binaire du trafic DNS en un problème de classification multi-classes, offrant des renseignements sur les menaces plus détaillés et plus exploitables. Cette approche globale améliore considérablement la détection et l'atténuation des attaques basées sur le DNS, renforçant ainsi la sécurité globale des systèmes réseau.

Mots clés: Cybersécurité, système de noms de domaine (DNS), trafic DNS, cyberattaques, classification binaire, classification multiclasse, apprentissage automatique (ML), ingénierie des fonctionnalités, analyse de corrélation, algorithmes de

clustering DBSCAN, renseignement sur les menaces ; Exactitude, précision, rappel, score F1, attaques basées sur DNS Sécurité réseau.

MACHINE LEARNING BASED ANOMALY DETECTION FOR DNS TRANSACTIONS

ABSTRACT

In the field of cybersecurity, the Domain Name System (DNS) is crucial for translating human-readable domain names into IP addresses, but its essential role also makes it a frequent target for cyberattacks. Traditional binary classification models, which categorize DNS traffic as either "benign" or "malicious," are limited in providing detailed threat intelligence necessary for effective threat mitigation. This work aims to address these limitations by using Ml technics to transform the binary classification problem into a multi-class classification problem, thereby enabling more nuanced detection of various types of DNS-based attacks. The primary objectives of this research are to preprocess the DNS dataset, conduct thorough feature engineering through correlation analysis, apply machine learning(ML) algorithms like clustering algorithms(DBSCAN) to create new class labels, and develop multi-class classification models. The models are rigorously evaluated using metrics such as accuracy, precision, recall, and F1-score. This model could be then deployed in a real-world environment to ensure its effectiveness. Our contributions include enhancing the granularity of DNS traffic classification, improving feature engineering processes, and advancing the state of cybersecurity research. By leveraging advanced clustering techniques and creating new class labels, our approach provides detailed threat intelligence, enabling more effective and targeted response strategies. Furthermore, by making our methodologies and findings available as open-source resources, we foster collaboration and accelerate the development of effective cybersecurity solutions. In conclusion, this research transforms the binary classification of DNS traffic into a multi-class classification problem, offering more detailed and actionable threat intelligence. This comprehensive approach significantly improves the detection and mitigation of DNS-based attacks, ultimately enhancing the overall security of network systems.

Keywords: Cybersecurity, Domain Name System (DNS), DNS Traffic , Cyberattacks, Binary Classification, Multi-class Classification, Machine Learning (ML), Feature Engineering, Correlation Analysis, Clustering Algorithms DBSCAN ,Threat Intelligence; Accuracy, Precision, Recall, F1-score,DNS-based Attacks Network Security.

GENERAL INTRODUCTION

CONTENTS

Context of work

In the rapidly evolving landscape of cybersecurity, DNS (Domain Name System) plays a pivotal role as it is responsible for translating human-friendly domain names into IP addresses. This process is fundamental to the functioning of the internet. However, the ubiquity and criticality of DNS have also made it a frequent target for cyber-attacks. Attackers exploit DNS vulnerabilities to conduct a variety of malicious activities, including phishing, distributed denial-of-service (DDoS) attacks, and data exfiltration. As a result, monitoring and analyzing DNS traffic for signs of attack is essential for maintaining network security. In this context, machine learning has emerged as a powerful tool for enhancing DNS security. By analyzing patterns in DNS traffic, machine learning models can effectively distinguish between benign and malicious activities. Traditionally, these models have focused on binary classification, categorizing traffic as either "attack" or "benign." However, the complexity of modern cyber threats demands a more nuanced approach. This work seeks to extend the capabilities of DNS traffic analysis by transforming a binary classification problem into a multi-class classification problem, thereby enabling more granular detection and response to different types of DNS-based attacks.

Problem statement

The core problem addressed in this work is the limitation of binary classification models in providing detailed threat intelligence from DNS traffic data. While these models can indicate whether traffic is benign or malicious, they do not offer insights into the specific type of attack. This lack of granularity can impede effective response and mitigation efforts, as different attacks may require different handling strategies.

To address this issue, this work proposes transforming the binary classification of DNS traffic into a multi-class classification problem. This involves not only detecting whether traffic is malicious but also identifying the specific type of attack. Achieving this requires a detailed analysis of DNS traffic features, the identification of relevant patterns and correlations, and the application of clustering techniques to uncover natural groupings within the data. These groupings can then be used to create new class labels, enhancing the classification model's ability to provide actionable insights.

Research hypothesis

The research hypothesis for this work is that transforming a binary DNS traffic classification model into a multi-class classification model will significantly improve the granularity and utility of threat detection and response. Specifically, it posits that by identifying and leveraging correlations between features within the DNS dataset, it is possible to create new, meaningful classes that enhance the model's ability to distinguish between different types of attacks. This, in turn, will provide more detailed threat intelligence and improve the overall effectiveness of cybersecurity measures.

Objectives

The general objective of this research is to develop an anomaly detection method for DNS transactions using multi-class classification and DBSCAN clustering to improve the precision,accuracy and reliability of detection compared to traditional binary classification. Specifically, the following are the specific objectives of this work:

- Analyze the limitations of binary classification in DNS anomaly detection.
- Apply the DBSCAN clustering algorithm on the DNS transaction dataset to identify patterns and anomaly clusters.
- Perform feature engineering based on clustering results to enrich the data and better characterize transactions.
- Develop and train multi-class classification models to detect anomalies.
- Compare the performance of multi-class models with binary models using appropriate metrics (precision, recall, F1-score).
- Review the results against existing literature on DNS anomaly detection.

Our contribution

Our contributions lies on a good number of points which are listed below: There are several works on DNS detection and classification going from traditional to machine and deep learning techniques. As for machine learning techniques, we have for example, Dharmaraj Patil et al.[15] who proposed a binary phishing detection method using lexical and string complexity analysis of URLs, employing Confidence Weighted (CW) and Adaptive Regularization of Weight Vectors (AROW) classifiers to achieve high accuracy and efficiency. Again,Hanghang Liu et Haoran Zhang[16] proposed a method of detecting DNS tunelling traffic using binary classification model using SVM,RF and obtained high accuracy with SVM 99.66%. On the other hand,M.Bakro et al [17]Proposed a hybrid feature selection technic combining GOA and GA algorithm to proper select features in other to have proper informations about all features where they later applied their method on three rescent datasets to test the performance of their model. Our contribution include:

- Improving anomaly detection by proposing and architecture that intergrates DBSCAN clustering which reduces false positive and negative rates and improves classification accuracy on several dataset as compared to existing

works.

- New insights for feature engineering are also promoted since DBSCAN clustering helps identify structures in the dataset that are not visible hence enriching feature engineering and data for more precise analysis.
- Methodological approach by combining clustering and multi-class classification offers a new methodology for anomaly detection in environments with limited feature information, applicable to other cybersecurity and data analysis fields.
- Contribution to Literature by providing a detailed comparison between binary and multi-class approaches for DNS anomaly detection enriches existing literature and offers perspectives for future research in the field. Finally,establishing a correlation among fatures in the binary classification and optimizing hyparameters leading to a high performant model of accuracy 98.34%

Work outline

The rest of this document is organised as follows:

Chapter I:Domain name system and machine learning: In this chapter, we will give an overview on DNS, its definition,how DNS works, types of DNS transactions (queries/responses) and servers. Then we will define ML, give a brief review of ML based techniques used in DNS security, some application of ML in cybersecurity. Finally, we will take the case of DNS.

Chapter II:State Of the arts on anomaly detection in DNS transactions: In this chapter, we will present DNS anomalies,starting from the traditional anomaliy techniques to Ml based techniques in binary and multi-class detection. Then we will present some supervised and unsupervised global ML approaches used in the literature to detect DNS anomalies. Finally, we will present some solutions present in the literature to face binary and multi-class anomaly detection in dns transactions as well as their advantages and disadvantages.

Chapter III:Contribution to machine learning-based anomaly detection for DNS transactions: This chapter is dedicated to the presentation of our proposed

solutions as well as their implementations. We start by describing our datasets for both binary and multi-class classification. Next, designing the model architecture and description of each component. Again,we will preprocess the data then select our model for training. Next, we will evaluate the results obtained after training based on performance metrics. Finally, we will compare the performance of both architectures together and with related works in a tabular form and discussions.
Conclusion and future works: This section is the conclusion of our work which presents the overall functioning of our proposed models and approaches, as well as future works.

I

CHAPTER

DOMAIN NAME SYSTEM AND MACHINE LEARNING

CONTENTS

I.1. Introduction

The Domain Name System (DNS) is a hierarchical and decentralized naming system for devices connected to the internet or a private network. It translates human-friendly domain names like www.example.com into IP addresses like 192.0.2.1 that computers use to identify each other on the network. Without DNS, users would need to remember and use IP addresses to access websites or online services. In the other hand,artificial intelligence (AI) is the simulation of human intelligence processes by machines, especially computer systems. These processes include learning (the acquisition of information and rules for using the information), reasoning (using rules to reach approximate or definite conclusions), and self-correction. AI encompasses a variety of subfields and techniques aimed at enabling machines to perform tasks that typically require human intelligence. A major sub field of AI is Machine Learning (ML), which focuses on the development of algorithms that allow computers to learn from and make predictions based on data. Machine Learning is divided into several categories which ware going to be discussed later in this chapter.

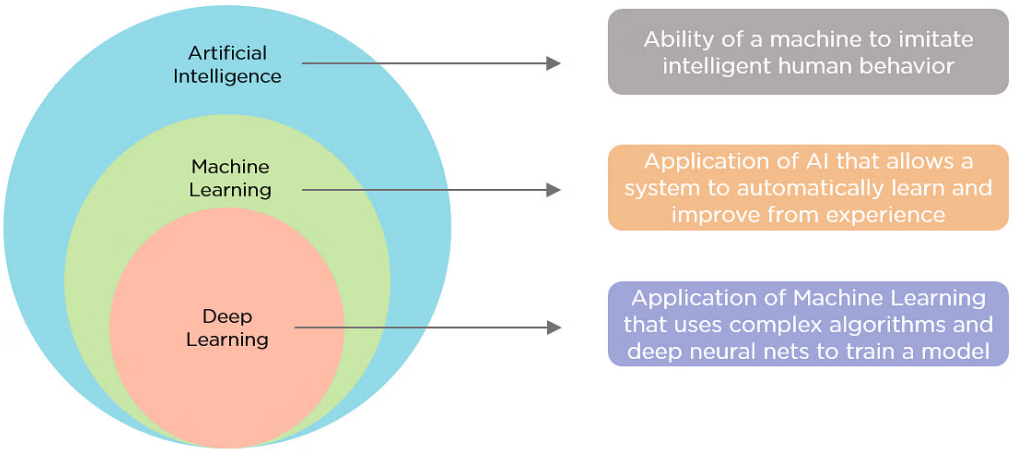


Figure 1 – *The domains of AI [1].*

I.2. Overview of DNS

The Domain Name System (DNS) is an essential component of the internet infrastructure, functioning as the internet's directory. DNS translates human-friendly domain names like www.example.com into numerical IP addresses like 192.0.2.1 that computers use to identify each other on the network. This process is crucial because, while domain names are easy for people to remember, computers use IP addresses to locate and communicate with each other.

I.2.1. Basic concepts

1. Domain Names: A domain name is a string that identifies a realm of administrative autonomy, authority, or control on the internet. Examples include example.com, openai.com.

2. IP Addresses: These are numerical labels assigned to each device connected to a computer network that uses the Internet Protocol for communication. They serve two main functions: identifying the host or network interface and providing the host's location.

I.2.2. DNS hierarchy:

The DNS is organized in a hierarchical structure:

- Root Level: The top of the DNS hierarchy, represented by a dot (.). It contains information that points to the top-level domain (TLD) servers.

- Top-Level Domains (TLDs): These are the highest level of domain names in the root zone. Common TLDs include .com, .org, .net, as well as country code TLDs like .uk, .jp.
- Second-Level Domains: These are directly below TLDs and are typically what people purchase from domain registrars (e.g., example in example.com).
- Subdomains: These are domains that are part of a larger domain (e.g., blog.example.com where blog is a subdomain of example.com).

I.2.3. DNS-components

- DNS Resolvers: These are responsible for initiating and sequencing the queries that lead to a full resolution of the requested resource, usually managed by ISPs or enterprise networks.
- Root Name Servers: These servers know where to direct queries for each top-level domain (TLD).
- TLD Name Servers: These servers store information for each domain within the TLD and direct queries to the appropriate authoritative name servers.
- Authoritative Name Servers: These servers contain the actual DNS records for a domain and provide answers to queries about those records.

I.2.4. DNS records:

DNS records are used to store data about domain names. Key types include:

- A Record: Maps a domain to an IPv4 address.
- AAAA Record: Maps a domain to an IPv6 address.
- CNAME Record: Canonical Name record; maps an alias domain name to the canonical domain name.

- MX Record: Mail Exchange record; specifies the mail server responsible for receiving email for the domain.

- NS Record: Name Server record; indicates which DNS server is authoritative for that domain.

- PTR Record: Pointer record; used for reverse DNS lookups to map an IP address to a domain name.

- TXT Record: Text record; used to store arbitrary text data, often for verification and security purposes.

How DNS works

Lets Consider the following figure:

How DNS Works

Lets consider the workflow of DNS in FIGURE **??** where a user enters a url through his or her web browser till when obtained the corresponding ip address. Lets explains detailly the scenario from ppoint 1 to 13:

1. **Query for the iP address:** When a user types a URL (e.g., http://mypage.com) into their web browser, the process of resolving the domain name to an IP address begins.

2. **Local cache check:** The first step is to check if the IP address is stored in the local cache, which includes the browser cache, OS cache, and router cache. If the IP address is found here, the process stops, and the website is loaded from the cached IP address. If not, it moves to the next step.

3. **Host files:** If the IP address is not found in the local cache, the system checks the host files. These files can manually map domain names to IP addresses. If the IP address is found in the host files, the process stops, and the website is loaded.

4. **DNS resolver query:** If the IP address is not found in the local cache or host files, the query is sent to the DNS resolver, also known as a recursive

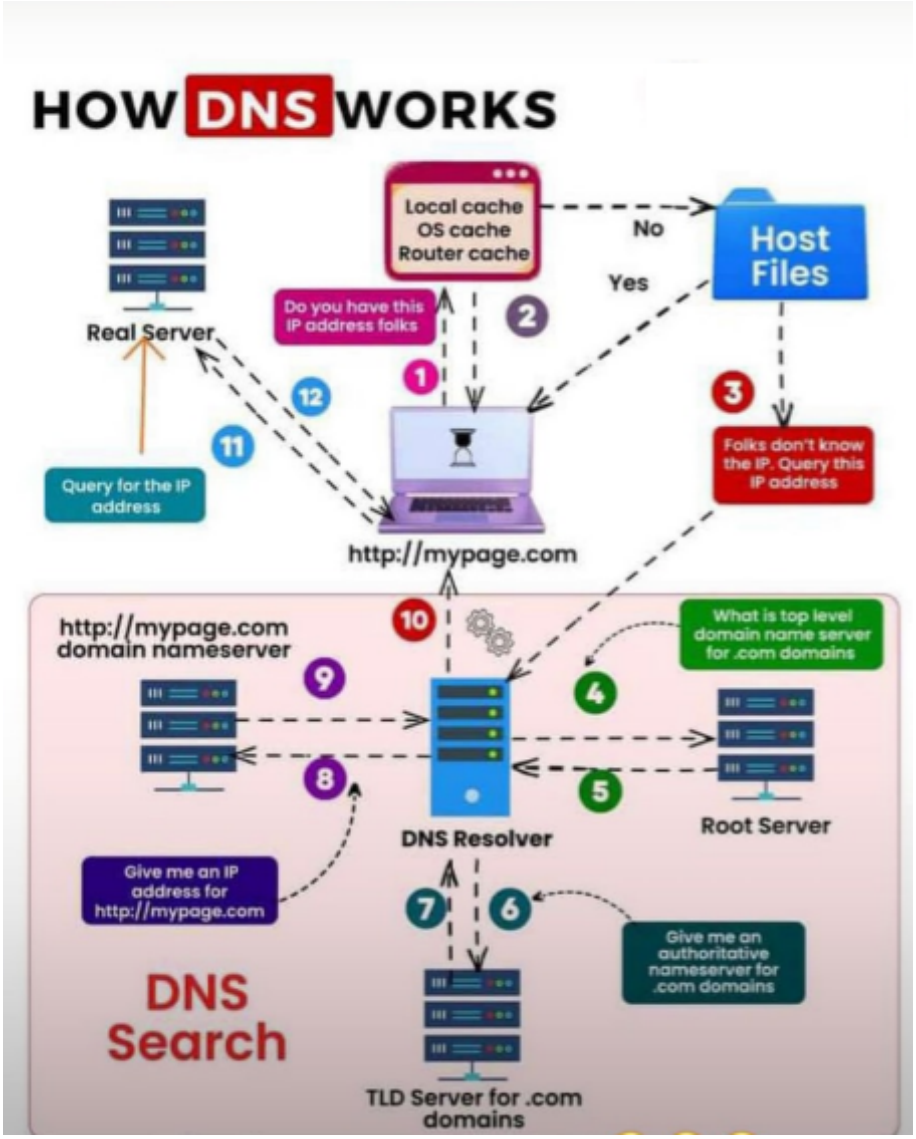


Figure 2 – *how dns resolution works*

resolver. This resolver is typically provided by the user's Internet Service Provider (ISP).

5. **Root server query:** The DNS resolver then queries a root server. There are 13 sets of root servers worldwide, and they contain information on top-level domain (TLD) servers (such as .com, .org, .net).

6. **TLD server query:** The root server responds with the address of a TLD server responsible for the queried domain. For example, if the query is for a .com domain, the root server provides the address of the .com TLD server.

7. **Authoritative name server query:** The DNS resolver then queries the TLD server for the authoritative name server that holds the actual DNS records for the domain (mypage.com in this case).
8. **Domain Nameserver Query:** The TLD server responds with the address of the domain's authoritative nameserver.
9. **Authoritative nameserver response:** The DNS resolver queries the authoritative nameserver for the specific IP address of the domain (mypage.com).
10. **IP address response:** The authoritative nameserver responds with the IP address of the requested domain.
11. **Local cache update:** The DNS resolver stores the IP address in its local cache for future queries. This helps speed up the resolution process for subsequent requests for the same domain.
12. **Query completion:** The DNS resolver sends the IP address back to the user's computer.
13. **Website access:** With the IP address obtained, the user's computer can now contact the real server hosting the website (mypage.com) and load the webpage.

I.2.5. DNS anomalies

Anomalies are patterns in data that deviate from the expected norm, arising from various abnormal activities such as credit card fraud, mobile phone fraud, and cyberattacks. These deviations are crucial for data analysts to identify. One important aspect of anomaly detection is understanding the nature of the anomaly, which can be categorized as follows:

I.2.5.1. Point anomaly

When an individual data instance significantly deviates from the typical pattern of the dataset, it is identified as a point anomaly. For instance, if a person typically uses five liters of fuel per day for their car but suddenly uses fifty liters on a random day, this would be considered a point anomaly.

I.2.5.2. Contextual anomaly

This type of anomaly occurs when a data instance is anomalous within a specific context, also known as a conditional anomaly. For example, credit card expenditures tend to be higher during festive periods such as Christmas or New Year. While high spending is normal for these contexts, it might be anomalous at other times of the year. Conversely, equally high expenditure during a non-festive month could be deemed a contextual anomaly.

I.2.5.3. Collective anomaly

When a collection of similar data instances behaves anomalously with respect to the entire dataset, the group of data instances is termed a collective anomaly. For example, in a human's Electrocardiogram (ECG) output, the existence of low values for a long period of time indicates an outlying phenomenon corresponding to an abnormal premature contraction, whereas one low value by itself is not considered anomalous.

I.2.5.4. Seasonal anomaly

Deviations from expected seasonal patterns. For instance, an unexpected spike in electricity consumption during a typically low-demand season could be a seasonal anomaly.

I.2.5.5. Spatial anomaly

Anomalies that occur in a specific spatial context. For example, unusual traffic congestion in a normally low-traffic area could indicate a spatial anomaly.

I.2.5.6. Behavioral anomaly

When an entity behaves differently than its usual pattern. For example, a user's sudden change in login times or locations could be considered a behavioral anomaly.

I.2.6. DNS Attacks

DNS attacks exploit vulnerabilities in the DNS protocol and infrastructure to disrupt normal operations or intercept data.

I.2.6.1. Man-in-the-middle (MitM) attack

Overview

In a MitM attack, the attacker secretly intercepts and relays messages between two parties who believe they are communicating directly with each other.

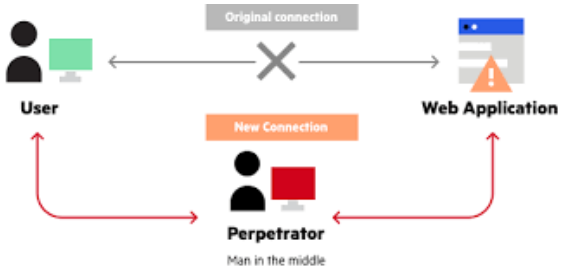


Figure 3 – *Man in the middle attack [2]*

How it Works

1. **Interception:** The attacker intercepts the communication channel between the client and the DNS server.
2. **Relay:** The attacker can modify or eavesdrop on the messages.
3. **Consequences:** This can lead to data theft, insertion of malicious content, or redirection to fake websites.

I.2.6.2. DNS Spoofing (DNS Cache Poisoning)

Overview

DNS Spoofing, or DNS Cache Poisoning, involves corrupting a DNS server's cache with false information.

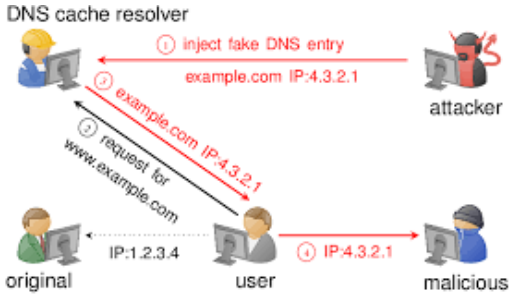


Figure 4 – *Cache poisoning [3]*

How it Works

1. **Injection:** The attacker sends forged DNS responses to a DNS resolver.
2. **Poisoning:** These false responses are cached by the DNS resolver.
3. **Redirection:** Future queries for the poisoned domain are directed to the attacker's IP address.
4. **Consequences:**

Users can be redirected to malicious websites designed to steal information or infect devices with malware.

I.2.6.3. DoS and DDos attacks

A DoS attack aims to make a DNS server unavailable by overwhelming it with a large number of requests. In the other hand,a DDoS attack is a more powerful form of DoS, where multiple systems (often part of a botnet) are used to flood the DNS server with traffic.

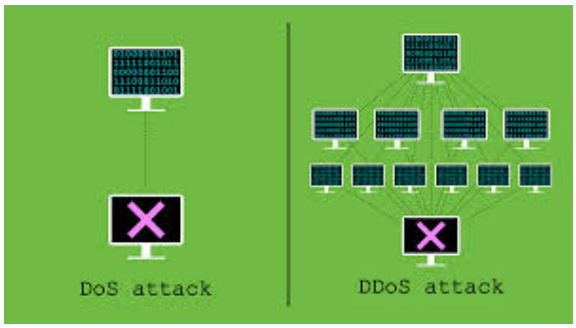


Figure 5 – *DoS and DDoS attack [4]*

How it Works

1. **Flooding:** The attacker sends a high volume of requests to the DNS server.
2. **Overloading:** The server becomes overwhelmed and cannot process legitimate requests.

Consequences: Legitimate users are unable to resolve domain names, leading to service disruption.

I.2.6.4. DNS Tunneling

Overview

DNS Tunneling exploits DNS to tunnel malware and other data through a network firewall.

How it Works

1. **Encoding Data:** Data is encoded into DNS queries.
2. **Transmission:** These queries are sent to an external server controlled by the attacker.
3. **Decoding Data:** The external server decodes the data from the DNS queries.

Consequences This technique can be used to exfiltrate sensitive information or establish command and control channels for malware.

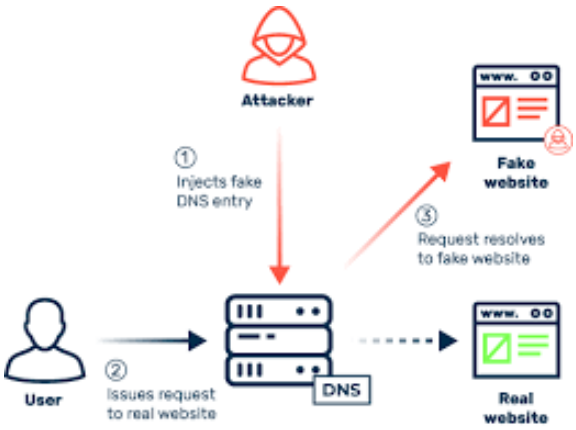


Figure 6 – *DNS tunneling [5]*

I.2.6.5. Cache Poisoning

Overview

Cache poisoning is a specific type of DNS Spoofing that targets DNS caches, causing them to store incorrect information.

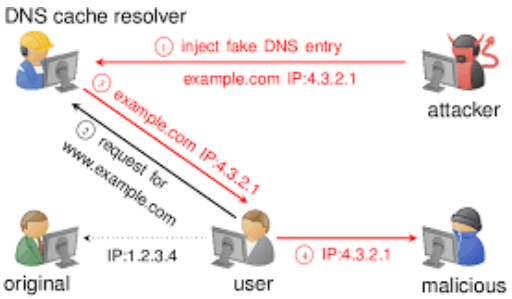


Figure 7 – *cache poisonning[3]*

How it Works

1. **Forged Responses:** The attacker sends forged DNS responses to the DNS server.
2. **Cache Storage:** These forged responses are stored in the DNS server's cache.
3. **Misdirection:** Subsequent queries for the poisoned domain are directed to the attacker's IP address.

Consequences: Similar to DNS Spoofing, users are redirected to malicious sites, facilitating phishing, malware distribution, and other attacks.

I.3. Types of Artificial Intelligence

Artificial intelligence, so far, represents the ability of a program or a machine to solve complex problems by imitating human cognitive processes. It is also capable of performing intellectual tasks at a level often superior to humans in most cognitive domains. There are mainly two types of AI: strong artificial intelligence (AGI or strong AI) and weak artificial intelligence (weak AI).

I.3.1. Weak Artificial Intelligence

Weak AI, also called narrow AI, focuses on solving specific tasks, such as solving chess problems. Unlike general AI, this type of artificial intelligence does not exhibit creativity or the ability to learn autonomously in a universal way.
Its learning capabilities rely on pattern detection (machine learning) or comparing large amounts of data. The benefits of weak AI are particularly notable in automation and process control, as well as in speech recognition and processing. Among its applications are text and image recognition, speech recognition, text translation, and navigation systems, among others.

I.3.2. Strong or General Artificial Intelligence

General AI represents an artificial intelligence model designed to surpass human cognitive abilities in almost every field. Unlike weak AI, which is limited to specific tasks, general AI has the ability to analyze, interpret, and solve complex problems that arise in daily life. This includes advanced creativity skills, a deep and nuanced understanding of human emotions, and the ability to make autonomous decisions and move independently. This form of AI could, for example, compose original music, write novels, or diagnose diseases with greater accuracy than the best human experts. It could also interact empathetically with humans, understand their emotional needs, and respond appropriately, while performing complex physical tasks without supervision [18]. However, implementing general AI requires considerable investments in several areas. This includes massive financial resources for research and development, advanced computing infrastructures to manage large-scale data processing and storage, and a highly skilled workforce to design, program, and maintain these complex systems.

I.4. Machine Learning

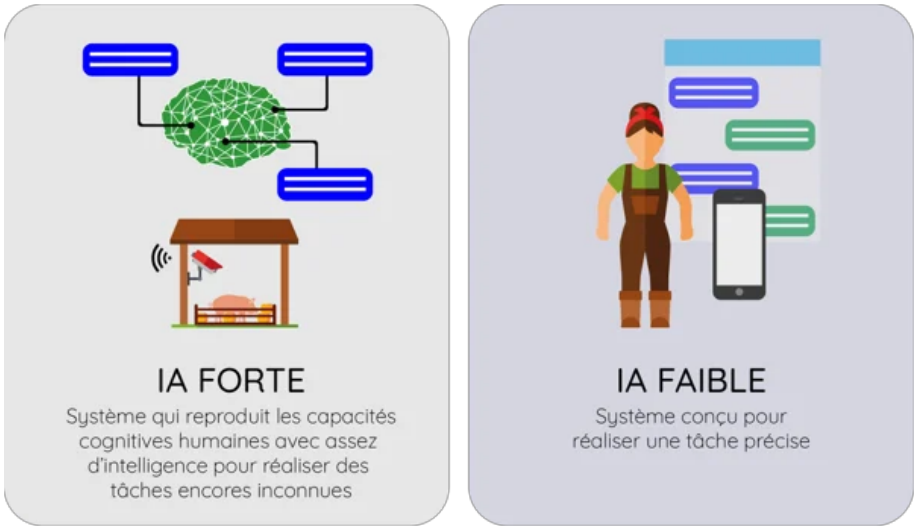


Figure 8 – *Weak AI vs.Strong AI*

Machine Learning is an AI technology that aims to iteratively improve the performance of algorithms through data processing. In other words, Machine Learning allows computer systems to learn without being explicitly programmed for each task. Practically, this modern discipline allows for the discovery of patterns and predictions from data, relying on statistical methods, data mining, pattern recognition, and predictive analytics [6]. Machine Learning is used in many fields such as speech recognition, fraud detection, consumer behavior prediction, and many more. Machine learning tasks are primarily organized into four categories: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.

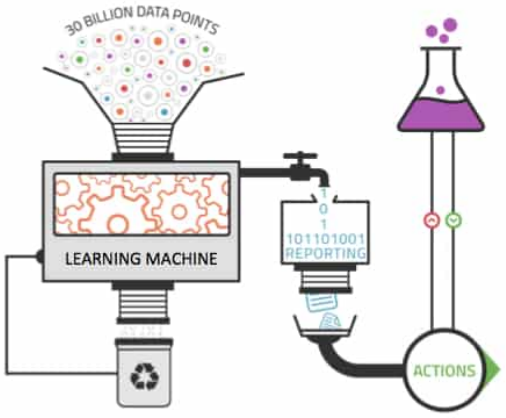


Figure 9 – *Applications of Machine Learning [6].*

I.4.1. Supervised Learning

Supervised learning is one of the most commonly used methods in the field of Machine Learning. It is a learning framework where an algorithm is trained on a labeled dataset. Each training example consists of an input and the corresponding output, allowing the algorithm to make predictions or classifications on new data based on the knowledge acquired during training, as shown in Figure 10. Supervised learning requires algorithms to train on a labeled dataset, and the goal is to map new input data to the desired target output value [19].

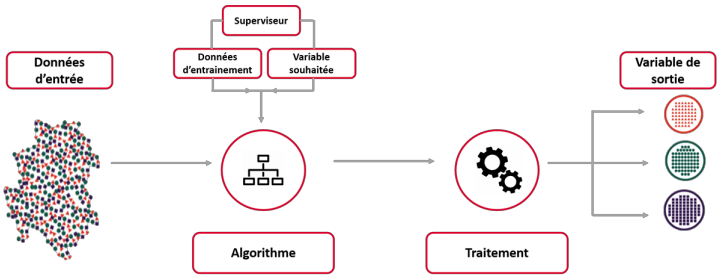


Figure 10 – *Supervised Learning [7].*

Mathematically, problems in this type of learning are generally defined or formatted as follows: from a set of n observations $\{\vec{x}1,\vec{x}_2,\ldots,\vec{x}_n\} = \{\vec{x}_i\} i = 1,\ldots,n$ belonging to the set X, and their labels $\{y_1,y_2,\ldots,y_n\} = \{y_i\}_{i=1,\ldots,n}$ described in a set Y, we aim to estimate the dependencies between the set X and Y to derive a model defined by a function $y_i = f(\vec{x}_i) + \varepsilon_i$, where ε_i is a random noise. The algorithm, once trained with the provided labeled data, becomes capable of predicting labels on new unlabeled data.

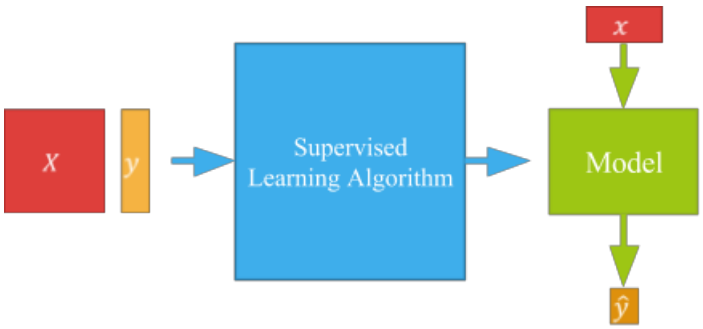


Figure 11 – *Supervised Learning*

In machine learning, the problems addressed are generally of two types: **Regression problems** and **Classification problems**.

1. **Regression Problems:** These involve predicting continuous or real values based on input variables.

2. **Classification Problems:** This involves predicting a discrete output variable, often called class labels, from a set of input data.

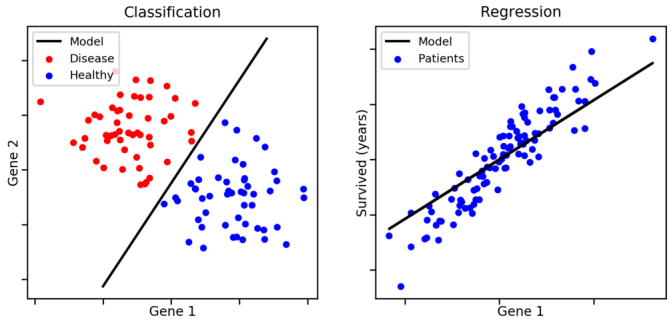


Figure 12 – *regression Vs.classification[8]*

Over time, several algorithms have been developed to address the various problems of supervised learning. Among the main ones are:

- **Linear Regression:** Used to predict the value of a dependent variable based on an independent variable . For example, predicting annual sales based on the education level or experience of a salesperson.

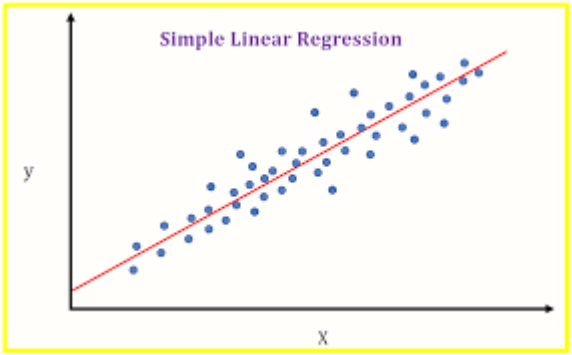


Figure 13 – *linear regression[9]*

- **Logistic Regression:** Suitable when dependent variables are binary.SVM is a relevant alternative when dependent variables are more difficult to classify.

- **Decision Trees:** Allow recommendations based on decision rules derived from classified data. For example, recommending a football team to bet on based on data like players' ages or the team's win percentage.

- **Random Forest:** A collection of decision trees, where each tree is based on a bootstrap sample [10]. The judgment is obtained by majority vote (classification) or by averaging results (regression).

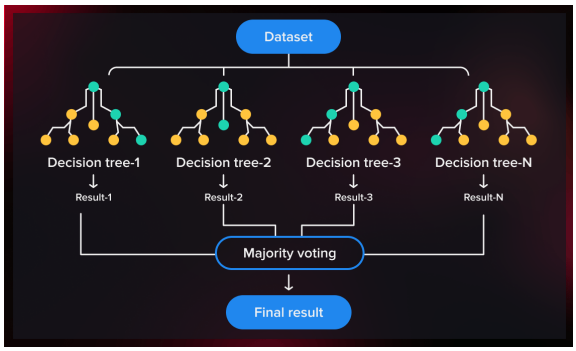


Figure 14 – *random forest[10]*

- **SVM:** Used for classification or regression. It uses a technique called the kernel trick to transform data and find an optimal boundary between possible outputs [20].

- **Neural Networks:** Generally performed using a computer, reproducing or predicting the behavior of processes based on the factors that determine them.

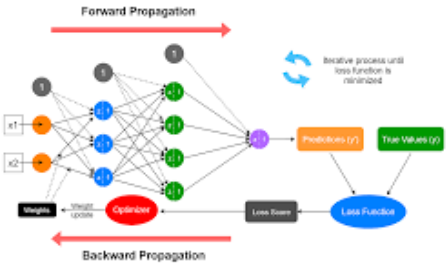


Figure 15 – *neural network[11]*

- **KNN:** A classification algorithm that uses the similarity of neighboring data points to predict the class label of a new sample.

I.4.2. Unsupervised Learning

Unsupervised learning explores data without corresponding output variables, allowing it to discover the underlying structure or distribution of the data. Unlike supervised learning, where algorithms are guided by correct answers, unsupervised learning operates autonomously, seeking to identify patterns and intrinsic structures in the data [13]. The two main categories of algorithms in unsupervised

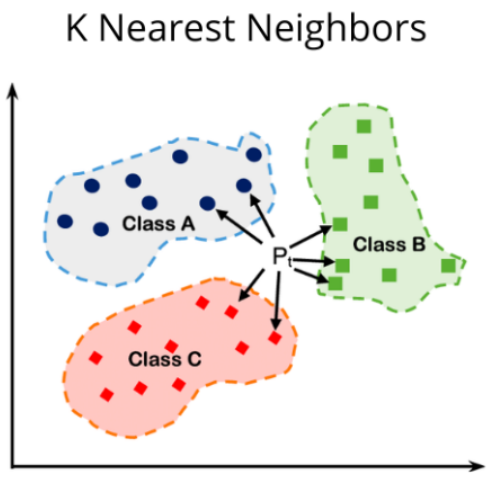


Figure 16 – *knn[12]*

learning are clustering and association. Clustering aims to divide a dataset into homogeneous groups, while association seeks to discover interesting relationships between variables in large databases.

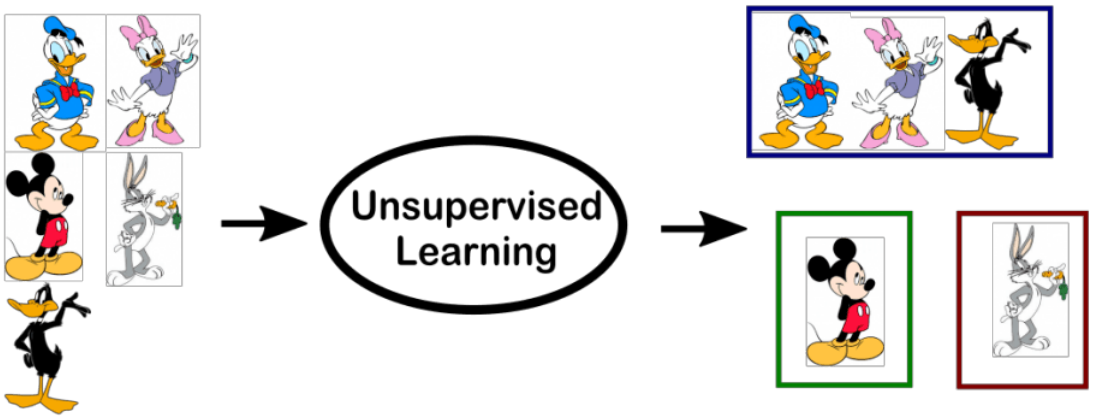


Figure 17 – *Unsupervised Learning [13]*

In this subfield, several algorithms are commonly used:

- **K-means Clustering:** It is an unsupervised learning algorithm used to solve clustering problems. K-means groups similar data points into clusters based on their characteristics or similarities [21].
- **Association Rules:** These algorithms identify frequent relationships and associations between variables in datasets [22].
- **Dimensionality Reduction:** This technique reduces the number of variables or features in a dataset while preserving relevant information. PCA is one of the popular methods for dimensionality reduction [21].

I.4.3. Semi-supervised Learning

In machine learning, problems involving a large amount of input data (X = $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$) with only a few labeled data (Y = $\{\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_n\}$) are called semi-supervised learning problems [13]. These problems lie between supervised learning, where all data is labeled, and unsupervised learning, where no labels are available. A concrete example of a semi-supervised problem would be an archive of photos where only a few images are labeled (e.g., dog, cat, person) while most are not. In many real cases, labeling all data can be time-consuming or expensive, often requiring the expertise of domain specialists. Consequently, unlabeled data are often easier to collect and store. To address semi-supervised problems, various techniques can be used. For example, unsupervised learning methods can be applied to discover and learn the underlying structure of input variables [23].

I.4.4. Reinforcement Learning

Reinforcement learning is a form of machine learning that focuses on training an algorithm to make decisions based on the rewards and punishments it receives. Unlike other methods where input-output pairs are provided, in reinforcement learning, we describe the current state of the system, specify a goal, provide a list of allowed actions with their environmental constraints, and then let the machine learning model experiment with achieving the goal on its own. This model uses the trial-and-error principle to maximize a reward [24].

Reinforcement learning, as illustrated in Figure 18, involves the use of several essential concepts and metrics, among which the main ones are as follows:

- **Agent**: a system or robot that interacts and acts in the environment.
- **Action** a: an action selected from the set of possible actions.
- **State** s: a particular situation in which the agent finds itself.
- **Reward** $r(s,a)$: a positive or negative gain obtained by performing action a in state s. The goal is to maximize the total benefits of a policy.

I.5. Data Mining

With the advent of artificial intelligence, the value of data has become crucial. Their diversity and rapid accumulation have resulted in an impressive mass of data requiring thorough analysis to be fully utilized. In this context, the implementation

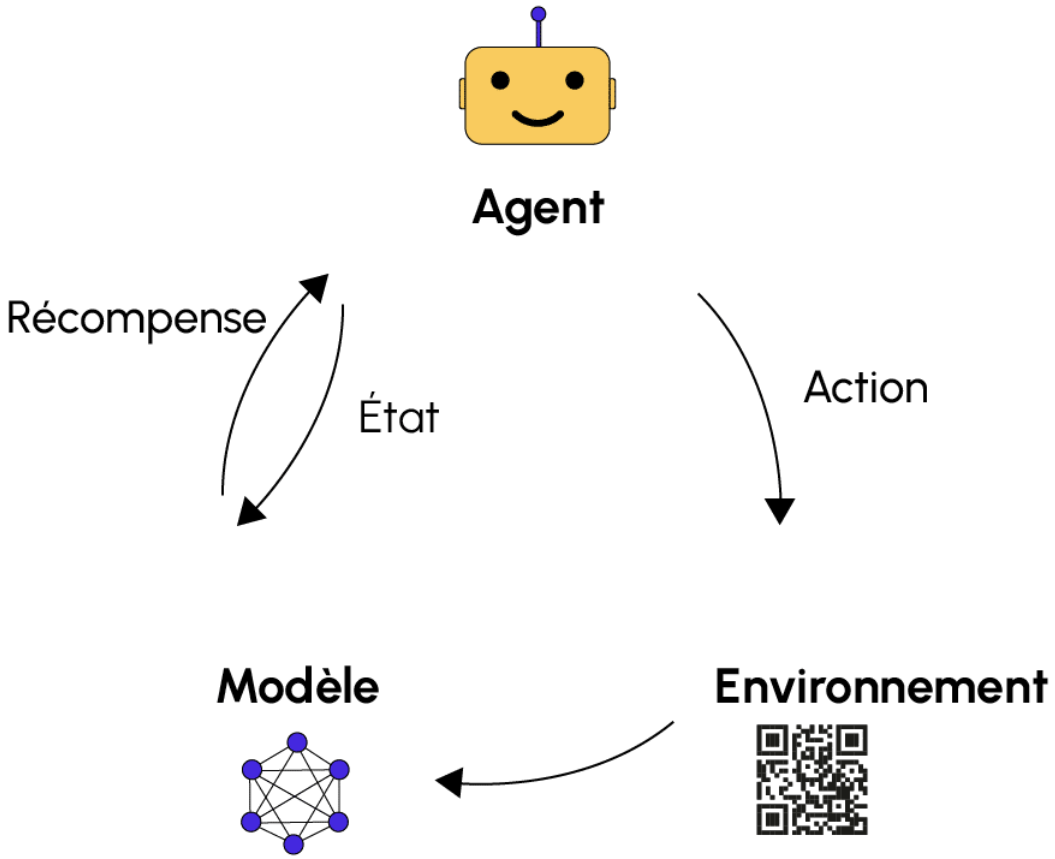


Figure 18 – *Reinforcement learning [13]*

of analysis tools capable of extracting relevant information from these data is of paramount importance for data analysis professionals. For several decades, data mining has been the preferred method. This approach aims to search for and extract useful and unknown information from large sets of data stored in databases or data warehouses. The emergence of data mining in the 1990s was facilitated by technological advances that improved the computing power of computers. Data mining tools offer powerful statistical, mathematical, and analytical capabilities, aiming to analyze large datasets to identify trends, patterns, and relationships, to make informed decisions and plan effectively.

I.5.1. Data Mining Process and Operation

Data mining is generally carried out by data scientists and other professionals specializing in data analysis. However, it can also be undertaken by business analysts and personnel with a solid understanding of data, acting as data scientists within the organization. The essential elements of this exploration include ma-

chine learning, statistical analysis, as well as data management tasks necessary to prepare the data for analysis. The use of artificial intelligence (AI) tools has automated much of this process and simplified the exploration of massive datasets, such as customer databases or transaction records. The data mining process can be broken down into five steps, as illustrated in the following figure (Figure 19).

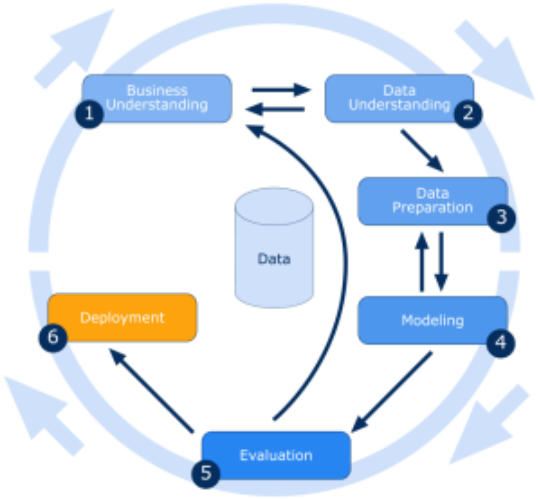


Figure 19 – *Data Mining Process [14]*

1. **Understanding the Problem:** Before any data is touched, extracted, cleaned, or analyzed, it is important to understand the underlying entity and the ongoing project. It is also crucial to define the objectives to be achieved for problem resolution.

2. **Data Collection:** This phase involves collecting the data that will be subjected to analysis. These data may be located in various source systems, such as data warehouses, which are increasingly used in big data environments. Data are also gathered from multiple external sources. These storage locations contain a mix of structured data (e.g., database tables) and unstructured data (e.g., text documents, images, videos). Regardless of the origin of the data, a data scientist often moves them to a data warehouse for subsequent process steps.

3. **Data Preparation:** This phase is one of the most crucial phases of the process. It involves performing a set of treatments on the data to make them usable. It begins with data exploration, profiling, and preprocessing, followed by data cleaning to correct errors and other data quality issues. These issues include duplicates in the data, inconsistencies, incomplete records, or obsolete formats.

4. **Modeling:** After the data preparation phase comes the information extraction phase. This step involves selecting the appropriate data mining technique and implementing one or more algorithms to perform the mining. These algorithms are machine learning algorithms that will train on these data to make the prediction process faster.

5. **Evaluation:** Once the information extraction models' training is complete, the performance of the data models is evaluated. The analysis results can be aggregated, interpreted, and presented to decision-makers who have been largely excluded from the data mining process until now. At this stage, organizations can choose to make decisions based on the results.

I.5.2. Data Mining Techniques

Data mining uses a set of algorithms and various techniques to convert large volumes of data into useful information. Here are some of the most popular techniques:

- **Association Rules:** This technique aims to discover patterns in the data. Association rules, in the form of "if...then" statements, identify relationships between data items. Support measures how frequently these relationships appear, while confidence evaluates their accuracy. Popular algorithms include Apriori and FP-Growth.

- **Clustering:** Used in the absence of information about the types of data objects, clustering aims to group similar data. Common algorithms include K-means and hierarchical classification.

- **Classification:** To classify data into different groups based on a target attribute, classifiers predict the target label for each record. Common techniques include decision trees, neural networks, k-nearest neighbors (K-Means), support vector machines (SVM), and Bayesian methods.

- **Regression:** To model the relationships between variables and predict continuous numerical values, regression uses algorithms such as linear regression, logistic regression, and polynomial regression.

These techniques are essential for extracting meaningful information from data and making informed decisions in many fields.

I.5.3. Importance of Data Mining

Data mining is an essential process in data science that enables engineers to better understand their datasets. This process is of crucial importance in data analysis across various fields for the following reasons:

- **Decision Making from Large Amounts of Data:** Data mining allows for extracting knowledge and hidden patterns from data. It helps organizations make informed decisions based on factual information rather than assumptions or intuitions. This can lead to more effective and profitable decisions.
- **Detection of Patterns and Trends:** Data mining helps discover significant patterns, correlations, and trends in the data. These insights can help organizations identify opportunities, anticipate future behaviors, detect market changes, and take preventive or proactive measures.
- **Forecasting and Planning:** By analyzing historical data and identifying relationships between variables, data mining can be used to predict future outcomes and establish plans and strategies accordingly. This can be useful in inventory management, demand forecasting, financial planning, etc.
- **Segmentation and Targeting:** Data mining allows grouping individuals or objects based on their common characteristics. This enables organizations to better understand their customers, segment the market into homogeneous groups, tailor offers and marketing campaigns accordingly, and target advertisements more precisely.
- **Personalization and Recommendation:** Data mining helps build personalized recommendation systems by analyzing users' preferences, behaviors, and purchase histories. This enhances user experience by providing relevant recommendations and increasing conversion and retention rates.

I.5.4. Applications of Data Mining

Data mining can be applied to all sectors that generate data and wish to benefit from it. Indeed, access to data is the key element to answering several questions within a domain. Here are some examples of how data mining is used in specific industries:

- **Healthcare:** For many years, the integration of data mining in healthcare has enabled doctors to benefit from more effective treatment methods using data

extracted from clinical trials and patient studies. Hospitals and clinics can thus improve patient outcomes and safety while reducing costs and response times.

- **Media and Telecommunications:** Companies in the media and telecommunications sector have access to an enormous amount of data on consumer preferences, such as the shows they watch or the internet plans they subscribe to. With these data, these companies can tailor their programming to consumer tastes, region, or other relevant factors. They can even recommend content to consume, a practice mastered by companies like Orange Cameroon, MTN Cameroon, YouTube, etc.
- **Bioinformatics:** Data mining plays an important role in the field of bioinformatics. Indeed, it is used to extract and analyze information on sequences, molecules, and gene expression to understand the functioning of certain biological processes.

I.6. Conclusion

In this chapter, we explored the interaction between the Domain Name System (DNS) and Machine Learning (ML), highlighting how ML enhances DNS security and performance. DNS, essential for translating domain names into IP addresses, is vulnerable to various attacks. ML offers powerful tools to detect and mitigate these threats by analyzing patterns and predicting anomalies.Key benefits of integrating ML with DNS include improved threat detection, anomaly identification, predictive analytics, and operational efficiency. However, challenges such as data privacy and model interpretability must be addressed. Finally,combining ML with DNS significantly enhances cybersecurity, ensuring a more secure and reliable internet infrastructure. Continued research and development in this field promise to refine these techniques and further strengthen DNS security.

II

CHAPTER

STATE OF THE ART ON ANOMALY DETECTION IN DNS TRANSACTIONS

CONTENTS

II.1. Introduction

Anomaly detection in DNS transactions is crucial for identifying malicious activities, such as botnet command and control, DDoS attacks, and data exfiltration. This overview will cover traditional rule-based and statistical methods, as well as modern machine learning techniques, including supervised and unsupervised learning approaches.

II.2. Traditional Anomaly Detection Techniques

Traditional detection-based techniques in DNS anomalies focus on identifying deviations from normal DNS behavior to detect malicious activities or irregularities. These techniques generally fall into several categories:

II.2.1. signature-based system

Signature-based detection involves using pre-defined patterns or signatures of known malicious DNS queries or responses. This method is effective against known threats but struggles with new or unknown threats. In this context,Smith et al [25] proposed an evaluation of signature-based anomaly detection in DNS traffic and obtained 85% detection rate and a false positive rate of 10%. however, their appraoch was limited in their inability to detect new threats since they rely on predefined patterns. Next,Lee et al [26] proposed a novel rule-based anomaly detection system tailored to DNS traffic characteristics.The proposed system employs a rule-based approach, where specific rules are formulated based on known patterns of normal DNS behavior. These rules are designed to flag deviations from expected DNS traffic patterns, indicating potential anomalies. The system utilizes features such as query types, query frequency, and domain reputation to establish and identify deviations. They had 88% detection rate and 12% false positive rate but however, their method was limited in that they required constant updating of rules or signatures. Next?,Kumar et al [27] futher proposed set of predefined rules to identify anomalies.They obtained 83% detection rate and 14% false positive rate. Their model was Prone to false positives and lacks adaptability and scalability for increasing as the volume of DNS traffic increases.

II.2.2. statistical methods

The statistical approach to DNS traffic analysis for anomaly detection is a sophisticated method aimed at identifying irregularities or deviations from normal behavior within DNS data flows. This approach relies on statistical techniques to establish baseline patterns of DNS activity and subsequently detect anomalies that may indicate malicious or suspicious activities. Here's an overview of the key statistical techniques employed in this approach:

Baseline Establishment: Before detecting anomalies, it's crucial to establish a baseline of normal DNS traffic behavior. This involves analyzing historical DNS data to understand typical patterns in query volumes, types of queries, response times, and other relevant metrics. Statistical methods such as mean, median, mode, and standard deviation are utilized to quantify these patterns and create a reference point for normalcy.

Z-Score Analysis: Z-score analysis is a common statistical technique used to identify outliers or anomalies in a dataset. In the context of DNS traffic analysis, z-scores are calculated for various metrics such as query frequency, response times, or query types. A z-score measures how many standard deviations a data point

is from the mean of the dataset. Data points with z-scores beyond a predefined threshold are flagged as anomalies, indicating potentially suspicious DNS activity
Time Series Analysis: Time series analysis is another essential statistical method used in DNS traffic analysis. It involves studying patterns and trends in DNS data over time to detect anomalies. By applying techniques such as moving averages, exponential smoothing, or autoregressive integrated moving average (ARIMA) modeling, analysts can identify irregularities in DNS traffic patterns that may signify malicious activities or unusual network behavior. Following this approach, Wang et al [28] proposed a methodology for establishing baseline behavior patterns using historical DNS data and described how statistical measures such as z-score analysis, time series analysis could applied to detect deviations from the baseline indicative of suspicious activities. They obtained 87% detection rate and 15% false positive rate. But were computationally intensive since they were not scalable,dificulties in features extraction. Again, Chen et al [29] proposed a model using clustering algorithms notably k-means and established a correlation analysis among features, and obtained an overall performance of 85% detection rate, and a 16% false positive rate which was as a result of lack of scalability in their model, the choice of the number of clusters also affected this results. [30] worked on the same dataset but used a different approach, instead of using clustering algorithms and correlation analysis, they proposed and exponentially Weighted Moving Average method for smoothing time series data to highlight trends and anomalies, DBSCAN (Density-Based Spatial Clustering of Applications with Noise Used to find core samples of high density and expand clusters from them rather than limiting the number of clusters at the begining as others did. Their peformance was good as compared to that of [29],that is,89% detection rate and 14% false positive rate. But this method was also limited since it required extensive historical data and not scalable. Pavel Čeleda et Radek Kreje. [31] Proposed a monitoring DNS traffic using both standard and extended flow records focusing on port-based identification traffic based on the assigned TCP and UDP port number 53,flow aggregation of packets with common properties into flows until termination, and extending these records with DNS-specific information. Their model had a significant performance and a low positive rate but however also had some limits on the flow record size and detection complexity. Kim et al. [32] Proposed an improved statistical model based on extended flow records to include DNS-specific information, such as queried domain names and types, response codes, and response data to allows for more detailed analysis of DNS traffic without significantly increasing the flow record size and also optimized flow cache to handle high volume of

DNS traffic. An accuracy of 91% detection rate and 11% false positive rate was obtained which improved the previous models but was limited because of the high complexity since the flow rate was extended.

II.3. Machine Learning-Based Techniques in binary and multiple classification

Machine learning involves training algorithms on data to identify patterns and make predictions.In DNS traffic detection, ML can help identify malicious traffic by learning from historical data.

Binary vs. Multi-Class Classification

Binary Classification: Here, the aim is to differentiates between two classes, such as benign(non malicious traffic) and malicious traffic. Binary classification is a core task in machine learning where data is divided into two distinct categories. It finds extensive applications in domains like spam detection, medical diagnosis, and fraud detection. Several algorithms like logistic regression, support vector machines, decision trees, and neural networks are commonly used for binary classification tasks. Evaluation of binary classifiers involves metrics such as accuracy, precision, recall, F1 score, and AUC-ROC, providing insights into model performance. In DNS anomaly detection, binary classification aims to classify DNS queries as normal or anomalous, leveraging features like query type, response time, and query frequency. Challenges include dealing with class imbalance, selecting relevant features, avoiding overfitting, and ensuring real-time processing capabilities. Despite these challenges, binary classification serves as a robust framework for effective anomaly detection across various domains.Gupta et al.[33] Proposed a random forest approach for DNS anomaly detection on trained on labeled data. Their datasets was made up of bengns and attacks where at the end of thier training, the obtained a detection rate of 91% and a 9% false positive rate,but how ever had some limits in their model resulted in overfitting on small datasets. Next,Patel et al [34] Proposed an approach using support vector machines model to adress this problem and obtained a detection rate of 89% and a 10% false positive rate which was not encouraging since their model was sensitive to parameter selection.

II.4. Supervised machine learning solution of binary classification of ;;;;

II.5. supervised learning solution of multiple classification of bakro et al.

This study proposes a hybrid feature selection approach using the grasshopper optimization algorithm (GOA) and the genetic algorithm (GA) to enhance IDS performance by efficiently selecting optimal features. A random forest (RF) classifier, trained on these features, benefits from addressing data imbalance with ADASYN for minority class oversampling and random under-sampling (RUS) for the majority class [35]. Evaluated on three datasets (UNSW-NB15, CIC-DDoS2019, and CIC Bell DNS EXF 2021), the approach achieved accuracies of 98%, 99%, and 92%, respectively. The hybrid IDS outperformed other classifiers and state-of-the-art methods, consistently delivering superior multi-class classification performance. Based on the CIC Bell DNS EXF 2021 and the results of 92% obtined, we noticed the limitations Our goal is to improve the performance of neural network classifiers by adjusting various parameters, including the n, connection density, selection of activation functions, and the number of training epochs. Additionally, we intend to implement systematic hyperparameter tuning for the classifiers using methods such as grid search, random search, and metaheuristic algorithms.

II.5.1. performance metrics

II.6. Conclusion

III

CHAPTER

CONTRIBUTION TO MACHINE LEARNING-BASED ANOMALY DETECTION FOR DNS TRANSACTIONS

CONTENTS

III.1. Introduction

This chapter unfolds our contributions towards refining DNS traffic analysis by transitioning from traditional binary classification to a more sophisticated multi-class classification framework. Our methodology leverages detailed preprocessing of DNS data, coupled with an in-depth feature engineering process that utilizes correlation analysis to select and transform the most impactful features. We further enhance the classification model by employing advanced clustering techniques, such as DBSCAN, to identify and define new class labels that correspond to distinct categories of DNS-based threats. These methodological advancements enable our models to capture a broader spectrum of malicious activities, offering a more granular and actionable insight into DNS traffic. The implementation of these techniques not only boosts the detection accuracy but also paves the way for deploying these models in practical cybersecurity settings. By providing a detailed account of these processes, this chapter aims to demonstrate how our innovative approach

can significantly improve the effectiveness of cybersecurity measures, thereby fortifying network defenses against an array of DNS threats.

III.2. Contribution To Binary Classification

III.2.1. datasets and description

The CIC Bell DNS EXF 2021 dataset [36] is a comprehensive dataset designed to enhance research in the field of network security. This dataset contains detailed records of network traffic with a focus on DNS (Domain Name System) traffic. The dataset is structured to include both benign and attack traffic, with the intention of facilitating the development and evaluation of security solutions such as intrusion detection systems (IDS) and anomaly detection algorithms.The dataset is typically provided in CSV format. Each file contains multiple records, each representing a single network packet or flow.The dataset includes both stateful and stateless features, capturing various aspects of the network traffic. It contains four classes: heavy attacks,light attacks,heavy benigns and light benigns. It has 756 columns of 148 integers,162 strings,108 decimals and 18 others.It has a size of 270.8MB.It incorporates 42 features, extracted from the DNS packets, ranging to approximately 1,019,318 samples. Out of these 42 features, 15 are stateless, 26 are stateful,and timestamp feature. Stateless features, extracted from individual DNS query packets, are independent of the temporal characteristics of queried domains or the DNS activity of hosts, thus minimizing computational strain during real-time operations. In contrast, stateful features consider a sequence of queries within a specific time window,imposing a greater computational burden on the detection system. However,for binary classification purposes, these four classes are grouped into two broader categories:
Attack (1): This includes both heavy and light attack traffic.
Benign (0): This includes both heavy and light benign traffic.

III.2.2. model architecture and description

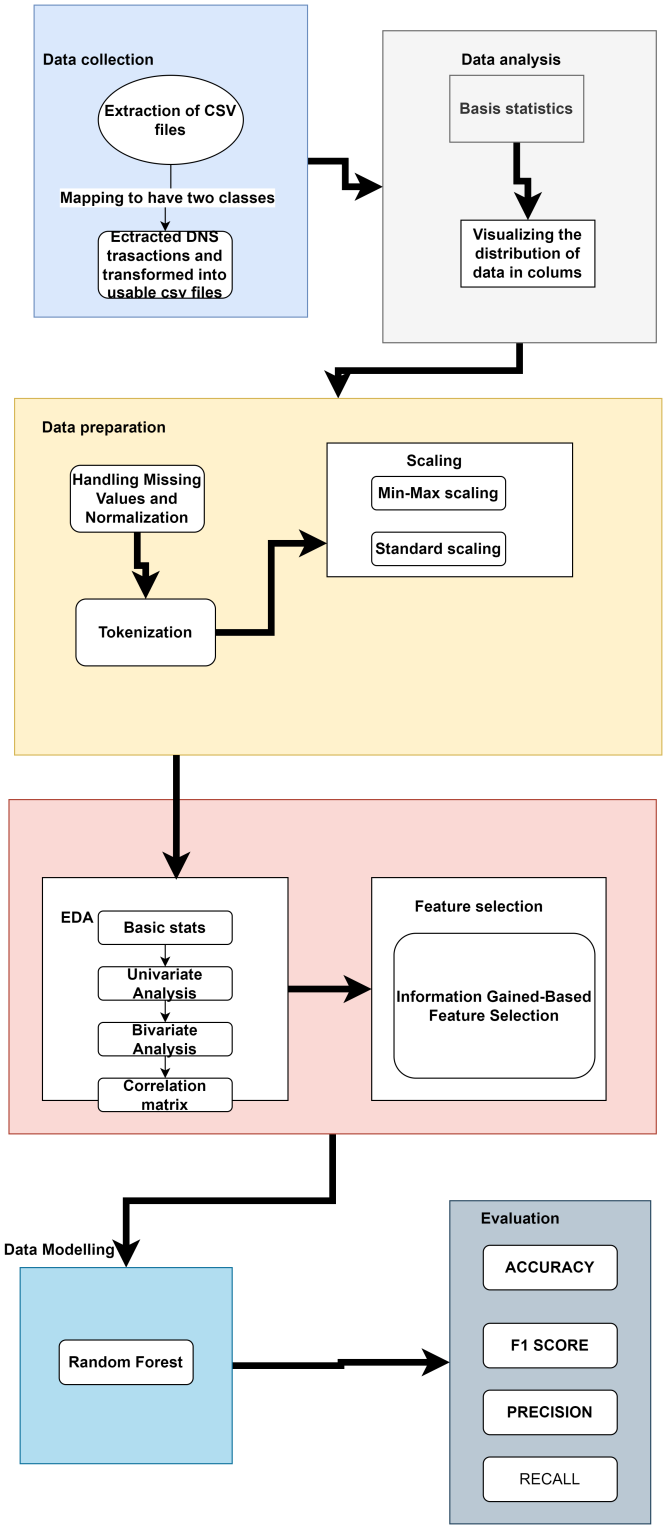


Figure 20 – *Proposed binary architecture*

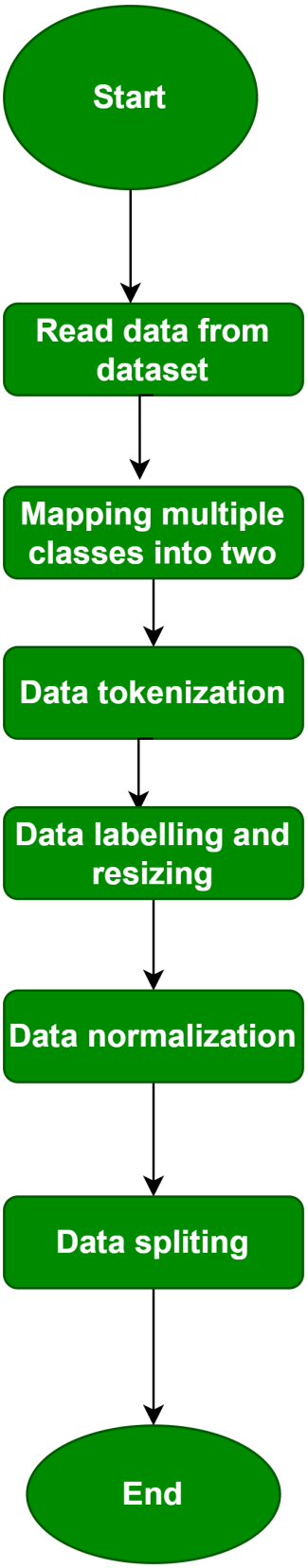


Figure 21 – *Binary data preprocessing*

III.2.3. Architecture description and preprocessing

III.2.3.1. Read data from dataset

Reading the dataset is the initial step to bring the data into the working environment for preprocessing and analysis. We use pandas for efficient data manipulation and analysis.

III.2.3.2. Mapping classes

The goal is to simplify the classification problem into a binary classification task, which often improves model performance and reduces complexity. The mapping helps in identifying attack and benign traffic clearly.

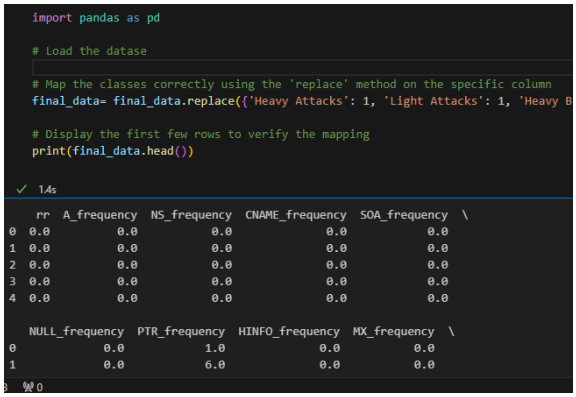


Figure 22 – *mapping process*

Dataset Before Mapping

Source IP	Destination IP	Packet Size	Class
192.168.1.2	192.168.1.1	1500	heavy attacks
192.168.1.3	192.168.1.4	1000	light benigns
192.168.1.5	192.168.1.6	500	heavy benigns
192.168.1.7	192.168.1.8	800	light attacks

Table I – *Dataset Before Mapping*

Dataset After Mapping

Source IP	Destination IP	Packet Size	Binary Class
192.168.1.2	192.168.1.1	1500	1
192.168.1.3	192.168.1.4	1000	0
192.168.1.5	192.168.1.6	500	0
192.168.1.7	192.168.1.8	800	1

Table II – *Dataset After Mapping*

Handling Missing Values Missing values in the dataset can lead to biased model outcomes and reduced accuracy. To handle missing values, the following steps were taken:

- **Identification**: Missing values were identified using summary statistics and visual inspection methods.

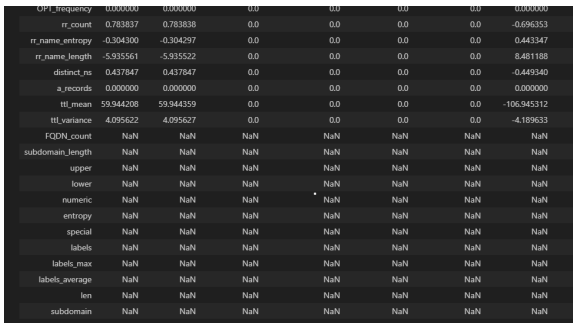


Figure 23 – *visualizing NaN values*

- **Imputation**: For numerical features, missing values were imputed using the mean or median values of the respective features. For categorical features, the most frequent category was used to fill in the missing values.

- **Removal**: In cases where the percentage of missing values in a feature was above a certain threshold (e.g., 50%), the feature was removed from the dataset to maintain data integrity.

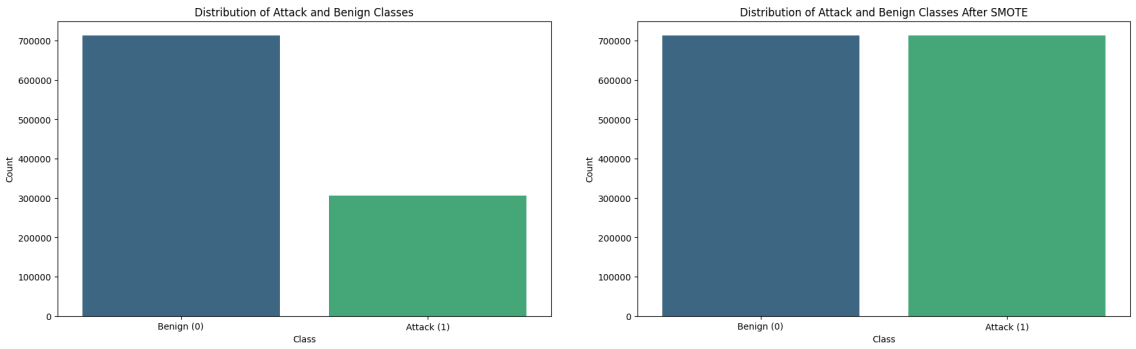
Data tokenization Tokenization converts textual data into a format that machine learning models can process (numeric). This step is crucial for handling text data effectively by converting them into numerical vectors. Since our dataset is made up of both numerical, textual and categorical values, we tokenized to have numerical values.

Figure 24 – *tokenization process*

Data labeling and resizing Label encoding converts categorical data into numeric form, which is essential for most machine learning algorithms. Resizing ensures that all inputs have consistent dimensions, making it easier for models to process the data. As machine learning models interpret numerical inputs, it becomes necessary to convert these categorical features into a machine-readable format. One hot encoding and label encoding are common methods for this conversion, each with its own pros and cons. Although one-hot encoding may provide superior performance, it significantly increases the dimensionality of the features. We choose label encoding due to its satisfactory results in [37].

Addressing the imbalanced nature of our training data Our training dat was imbalanced and contain a minority class. For this reason, we decided to perform and oversampling on the minority class using SMOTE technic , allowing the majority class unchanged.This oversampling approach increases the representation of minority classes either by replicating existing instances or by generating synthetic samples until all classes have an equal count. The most significant advantages

of this strategy include the preservation of majority class information, heightened sensitivity to the minority class,and a more comprehensive training process, as the model is trained on additional examples from the minority class.



(a) *Distribution of imbalanced classes before balancing with SMOTE*

(b) *Distribution of classes after balancing with SMOTE*

Figure 25 – *Distribution of classes before and after balancing*

Anomaly Detection and Handling Anomalies or outliers can distort the training process and degrade the model's performance. The following methods were employed to detect and handle anomalies:

- **Statistical Methods**: Z-score and IQR (Interquartile Range) methods were used to identify outliers in numerical features.
- **Domain Knowledge**: Features were cross-checked against known acceptable ranges based on domain knowledge to identify unrealistic values.
- **Treatment**: Detected anomalies were either corrected, or removed from the dataset to ensure the quality of the training data.

Standardization and Normalization To ensure that the features are on a comparable scale, standardization and normalization techniques were applied:

- **Standardization**: Numerical features were standardized to have a mean of 0 and a standard deviation of 1. This transformation is essential for algorithms that assume a Gaussian distribution of the data, such as logistic regression and linear discriminant analysis.
- **Normalization**: Features were normalized to a range of [0, 1] to ensure uniform scaling, which is particularly important for distance-based algorithms such as clustering algorithms.

These preprocessing steps ensured that the dataset was in an optimal state for model training, minimizing the risk of biases and enhancing the robustness of the classification models.

III.2.3.3. Feature Selection Process

Selecting the most relevant features from the dataset is a crucial step in building an effective classification model. It helps in reducing the dimensionality of the data, removing redundant and irrelevant features, and improving the overall performance of the model. For this purpose, we employed a combination of correlation analysis and the Genetic Optimal Algorithm - Genetic Optimization (GOA-GO) [37] algorithm. This section details the criteria and techniques used for feature selection.

Correlation Analysis:

Initially, we conducted a correlation analysis to identify and understand the relationships between different features in the dataset. This step involved the following processes:

- **Calculation of Correlation Coefficients**: We computed Pearson correlation coefficients for pairs of numerical features. This metric quantifies the linear relationship between features, with values ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation). Features with a high absolute correlation coefficient (close to 1 or -1) are considered strongly correlated.
- **Heatmap Visualization**: A heatmap was generated to visualize the correlation matrix, making it easier to identify clusters of highly correlated features.
- **Elimination of Redundant Features**: Features that were highly correlated with each other (above a predefined threshold, e.g., 0.9) were considered redundant. In such cases, one of the correlated features was removed to avoid multicollinearity, which can adversely affect the performance of some machine learning algorithms.

While correlation analysis helps in identifying linear relationships, it does not account for the non-linear interactions that might exist between features. To address this, we employed the GOA-GO algorithm for a more comprehensive feature selection process.

Genetic Optimal Algorithm - Genetic Optimization (GOA-GO):

The GOA-GO algorithm is an advanced feature selection technique that lever-

ages principles from genetic algorithms (GA) and optimization strategies to identify the most informative features. The process of applying the GOA-GO algorithm involved the following steps:

- **Initialization**: The algorithm begins by generating an initial population of candidate feature subsets. Each subset is represented as a binary chromosome, where each gene indicates the presence (1) or absence (0) of a particular feature.
- **Fitness Evaluation**: The fitness of each candidate subset is evaluated based on its contribution to the classification task. This involves training a preliminary model using the features in the subset and evaluating its performance using metrics such as accuracy, precision, recall, and F1-score.
- **Selection**: The most promising feature subsets are selected based on their fitness scores. Selection strategies such as roulette wheel selection or tournament selection are used to ensure that higher-performing subsets have a higher chance of being chosen for the next generation.
- **Crossover and Mutation**: To explore the feature space, the algorithm applies crossover and mutation operations to the selected feature subsets. Crossover involves exchanging segments of parent chromosomes to create new offspring, while mutation introduces random changes to individual genes, ensuring diversity in the population.
- **Iteration and Convergence**: The process of fitness evaluation, selection, crossover, and mutation is iteratively repeated over multiple generations. The algorithm converges when there is no significant improvement in the fitness scores or after a predefined number of generations.

By iteratively evaluating and optimizing the relevance of each feature, the GOA-GO algorithm effectively identifies a subset of the most informative features that significantly contribute to the classification task. This approach ensures that the selected features not only capture the important patterns in the data but also improve the computational efficiency of the model. The combination of correlation analysis and the GOA-GO algorithm enabled us to select the most relevant features from the *CIC Bell DNS EXF 2021* dataset, leading to improved model performance in detecting DNS-based attacks. This comprehensive feature selection process is critical in enhancing the accuracy, precision, and overall robustness of our classification models.

III.2.4. Model Development and Training

III.2.4.1. Selection of Machine Learning Algorithms

In our research, we considered several machine learning algorithms for the binary classification task of detecting DNS-based attacks. Among the algorithms evaluated, Support Vector Machines (SVM) and Random Forest were given particular focus due to their distinct strengths and complementary characteristics. This section provides an in-depth discussion of these algorithms and the rationale behind choosing the final model(s) for our binary classification problem.

Support Vector Machines (SVM) Support Vector Machines (SVM) are supervised learning models that are particularly effective for classification tasks. SVMs work by finding the hyperplane that best separates the data into distinct classes. The primary advantages of SVM include:

- **Effective in High-Dimensional Spaces**: SVMs are highly effective in cases where the number of dimensions exceeds the number of samples. This is particularly relevant for our dataset, which contains a large number of features.
- **Robustness to Overfitting**: By using regularization parameters, SVMs can effectively manage overfitting, ensuring that the model generalizes well to unseen data.
- **Kernel Trick**: SVMs can efficiently perform a non-linear classification using the kernel trick, implicitly mapping the input features into high-dimensional feature spaces.

Despite these advantages, SVMs also have certain limitations that must be considered:

- **Computational Complexity**: Training SVMs can be computationally intensive, especially with large datasets, as the complexity scales with the number of samples and features.
- **Parameter Selection**: The performance of SVMs is highly sensitive to the choice of kernel and regularization parameters, requiring careful tuning through techniques such as grid search or cross-validation.

Random Forest Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of the classes (classification) of the individual trees. The key benefits of Random Forest include:

- **Robustness to Overfitting**: By averaging multiple decision trees, Random Forest reduces the risk of overfitting, which is a common issue with individual decision trees.
- **High Accuracy**: Random Forest typically provides high accuracy in classification tasks due to its ensemble nature, which combines the strengths of multiple trees.
- **Feature Importance**: This algorithm provides an inherent measure of feature importance, which can be extremely valuable for understanding which features contribute most to the classification decision.
- **Scalability**: Random Forest is relatively easy to parallelize, making it scalable for large datasets.

However, Random Forest also has its limitations:

- **Complexity and Interpretability**: While Random Forest improves accuracy, the model complexity can make it harder to interpret the results compared to simpler models like single decision trees.
- **Computational Resource Intensive**: Training a large number of trees and aggregating their results can be resource-intensive, particularly with very large datasets.

Rationale for Choosing the Final Model(s) Given the strengths and weaknesses of both SVM and Random Forest, the final choice of model for our binary classification task was influenced by the following considerations:

- **Dataset Characteristics**: Our dataset, *CIC Bell DNS EXF 2021*, is characterized by high dimensionality and a large number of samples. Random Forest, with its capability to handle high-dimensional data and provide feature importance, emerged as a strong candidate. Meanwhile, SVM's effectiveness in high-dimensional spaces made it a useful benchmark for comparison.
- **Model Performance**: Initial experiments indicated that Random Forest consistently outperformed SVM in terms of accuracy, precision, recall, and F1-score on our dataset. The ensemble approach of Random Forest contributed to its superior performance by leveraging the diversity of multiple trees.

- **Computational Efficiency**: While SVMs offered robustness, the computational demands for training on a large dataset were significant. Random Forest, despite its complexity, proved to be more scalable and manageable within our computational resources.
- **Feature Interpretability**: The ability of Random Forest to provide insights into feature importance was invaluable for understanding the underlying patterns in the data and for further refining our model and preprocessing steps.

In conclusion, while both SVM and Random Forest were considered for their respective strengths, Random Forest was ultimately chosen as the primary model for our binary classification task due to its superior performance, scalability, and interpretability. This choice was validated through extensive experimentation and cross-validation, ensuring that the model generalizes well to unseen data and provides reliable detection of DNS-based attacks.

III.2.4.2. Training Process

The training process for our binary classification models involves several critical steps, including data splitting, model training, validation, and testing. This section outlines the methodology used to ensure that our models are robust, generalizable, and capable of accurately detecting DNS-based attacks.

Data Splitting To evaluate the performance of our machine learning models effectively, the dataset was split into three distinct subsets: training, validation, and test datasets. The *CIC Bell DNS EXF 2021* dataset was split as follows:

- **Training Set (70%)**, used for training the model. This subset contains the majority of the data and is used to fit the machine learning algorithm.
- **Validation Set (15%)**, used for hyperparameter tuning and model selection. This subset helps in assessing the model's performance during training and in making adjustments to prevent overfitting.
- **Test Set (15%)**, used for final evaluation. This subset is completely unseen by the model during training and validation, providing an unbiased evaluation of the model's performance.

Model Training The model training process involves the following steps:

- **Data Preprocessing**: The data preprocessing step includes cleaning, normalization, and feature selection. This ensures that the input data is in an optimal

format for training. Missing values were handled, outliers were removed, and the data was scaled to ensure consistency.

- **Feature Selection**: Using the GOA-GO (Genetic Optimal Algorithm - Genetic Optimization) algorithm, the most relevant features were selected to improve the model's performance and reduce computational complexity.
- **Model Initialization**: The machine learning models, specifically SVM and Random Forest, were initialized with their respective parameters. For SVM, kernel selection and regularization parameters were chosen. For Random Forest, the number of trees, maximum depth, and other hyperparameters were set.
- **Training the Model**: The training set was used to train the models. For Random Forest, multiple decision trees were built on various subsets of the data. For SVM, the algorithm aimed to find the optimal hyperplane that maximizes the margin between classes. The models were trained iteratively, adjusting weights and biases to minimize the classification error.
- **Validation During Training**: The validation set was used to evaluate the model performance at each epoch or iteration. This involved computing metrics such as accuracy, precision, recall, and F1-score to monitor overfitting and underfitting. Hyperparameters were tuned based on validation performance to enhance model generalizability.

Validation During training, the model's performance was continuously validated using the validation set. This process included:

- **Hyperparameter Tuning**: Using techniques such as grid search or random search, various combinations of hyperparameters were tested to find the best performing model configuration.
- **Early Stopping**: To prevent overfitting, early stopping was implemented. Training was halted when the performance on the validation set ceased to improve after a certain number of epochs.
- **Model Selection**: The model that performed best on the validation set, with the highest accuracy and F1-score, was selected for further testing.

Testing The final evaluation of the model's performance was conducted using the test set. This involved:

- **Unbiased Evaluation**: Since the test set was not used during training or validation, it provided an unbiased assessment of the model's generalization capability.
- **Performance Metrics**: The model's performance was measured using several metrics, including accuracy, precision, recall, and F1-score. These metrics provided a comprehensive evaluation of the model's effectiveness in detecting DNS-based attacks.
- **Comparison with Baseline Models**: The final model's performance was compared with baseline models and other state-of-the-art methods to benchmark its effectiveness.

Conclusion of Training Process The training process ensured that the model was well-tuned and capable of generalizing to unseen data. By rigorously validating and testing the model, we ensured its robustness and reliability in real-world scenarios. The steps taken during training, from data preprocessing to final testing, were crucial in developing a high-performing binary classification model for detecting DNS-based attacks.

III.2.4.3. Hyperparameter Tuning

Optimizing model performance is a critical step in machine learning, ensuring that the model generalizes well to unseen data. Hyperparameter tuning involves adjusting the parameters that govern the training process of a model to achieve the best performance. Two common methods for hyperparameter tuning are grid search and random search. This section describes these methods and their application in our study.

Grid Search Grid search is a systematic approach to hyperparameter tuning where a predefined set of hyperparameters is exhaustively tested to find the combination that results in the best model performance [38]. The steps involved in grid search are as follows:

- **Define Hyperparameter Space**: Identify the hyperparameters to be tuned and define the range of values for each. For example, for a Random Forest model, hyperparameters might include the number of trees (`n_estimators`), maximum depth (`max_depth`), and minimum samples split (`min_samples_split`).

- **Exhaustive Search**: Create a grid of all possible combinations of hyperparameters. Each combination represents a unique set of hyperparameters for the model.
- **Model Training**: Train the model using each combination of hyperparameters on the training dataset. Evaluate the performance on the validation set using a chosen metric such as accuracy, precision, recall, or F1-score.
- **Select Best Model**: Identify the combination of hyperparameters that yields the best performance on the validation set. This combination is considered optimal and is used to train the final model.

Random Search Random search is an alternative to grid search that involves randomly sampling combinations of hyperparameters from a specified distribution [39]. This method is often more efficient than grid search, especially when the hyperparameter space is large. The steps involved in random search are as follows:

- **Define Hyperparameter Space**: Similar to grid search, identify the hyperparameters to be tuned and specify the range or distribution of values for each. For example, for an SVM model, hyperparameters might include the penalty parameter (`C`) and the kernel coefficient (`gamma`).
- **Random Sampling**: Randomly sample a fixed number of combinations from the hyperparameter space. Each sample represents a unique set of hyperparameters for the model.
- **Model Training**: Train the model using each randomly sampled combination of hyperparameters on the training dataset. Evaluate the performance on the validation set using the chosen metric.
- **Select Best Model**: Identify the combination of hyperparameters that yields the best performance on the validation set. This combination is considered optimal and is used to train the final model.

Application in Our Study In our study, we employed both grid search and random search for hyperparameter tuning of the SVM and Random Forest models. The process was as follows:

- **Hyperparameter Tuning for SVM**:

  - **Hyperparameters Considered**: The penalty parameter (`C`) and the kernel coefficient (`gamma`) for the RBF kernel.
  - **Grid Search**: A grid of values was defined for `C` (e.g., {0.1, 1, 10, 100}) and `gamma` (e.g., {0.001, 0.01, 0.1, 1}). Each combination was evaluated, and the best performing combination was selected.
  - **Random Search**: Randomly sampled 50 combinations of `C` and `gamma` from a specified range. The combination yielding the best performance on the validation set was chosen.

- **Hyperparameter Tuning for Random Forest**:
  - **Hyperparameters Considered**: The number of trees (`n_estimators`), maximum depth (`max_depth`), and minimum samples split (`min_samples_split`).
  - **Grid Search**: A grid of values was defined for `n_estimators` (e.g., {100, 200, 300}), `max_depth` (e.g., {10, 20, 30}), and `min_samples_split` (e.g., {2, 5, 10}). Each combination was evaluated, and the best performing combination was selected.
  - **Random Search**: Randomly sampled 100 combinations of `n_estimators`, `max_depth`, and `min_samples_split` from a specified range. The combination yielding the best performance on the validation set was chosen.

Evaluation and Selection The performance of the models was evaluated using cross-validation on the training set to ensure robustness and to avoid overfitting. The optimal hyperparameters identified through grid search and random search were then used to train the final models, which were subsequently evaluated on the test set to measure their generalization performance.

Conclusion of Hyperparameter Tuning The use of grid search and random search for hyperparameter tuning allowed us to systematically explore the hyperparameter space and identify the best configurations for our models. This process significantly improved the model performance, leading to more accurate and reliable detection of DNS-based attacks.

III.2.4.4. Evaluation Metrics

To comprehensively assess the performance of our binary classification models, we employed several evaluation metrics. These metrics provide different perspectives on the model's performance and help in understanding the trade-offs between various types of errors. The evaluation metrics used are as follows:

Accuracy Accuracy is the most straightforward metric and measures the proportion of correctly classified instances over the total number of instances. It is calculated as:

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

where TP (True Positives) and TN (True Negatives) are the number of correct predictions for the positive and negative classes, respectively, and FP (False Positives) and FN (False Negatives) are the incorrect predictions [40].

Precision Precision measures the proportion of true positive predictions over the total number of positive predictions (both true and false). It is given by:

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

Precision is particularly useful when the cost of false positives is high [41].

Recall Recall, also known as sensitivity or true positive rate, measures the proportion of true positive predictions over the total number of actual positives. It is calculated as:

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

Recall is important when the cost of false negatives is high [41].

F1-Score The F1-score is the harmonic mean of precision and recall, providing a single metric that balances both. It is particularly useful when the class distribution is imbalanced. The F1-score is defined as:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1-score ensures that both precision and recall are reasonably high [40].

Area Under the ROC Curve (AUC-ROC) The AUC-ROC measures the model's ability to distinguish between the positive and negative classes. The ROC curve plots the true positive rate (recall) against the false positive rate (1-specificity). AUC-ROC is a single scalar value that summarizes the performance across all classification thresholds:

$$\text{AUC-ROC} = \int_0^1 \text{ROC}(t)\, dt$$

A higher AUC-ROC value indicates better model performance [42].

Matthews Correlation Coefficient (MCC) The MCC (Matthews Correlation Coefficient) is a correlation coefficient that takes into account all four values (TP, TN, FP, FN). When the dataset is unbalanced (the number of samples in one class is much larger than the number of samples in the other classes), accuracy cannot be considered a reliable measure anymore, as it provides an overoptimistic estimation of the classifier's ability on the majority class [43]. An effective solution overcoming the class imbalance issue comes from the MCC, which is generally considered a more robust metric than accuracy, especially when dealing with imbalanced datasets. The MCC is calculated as:

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

Conclusion of Evaluation Metrics Using these metrics, we were able to evaluate and compare the performance of our binary classification models comprehensively. Each metric provided insights into different aspects of the model's performance, allowing us to understand the trade-offs and optimize the models effectively for detecting DNS-based attacks.

III.2.5. Model Performance and Results

This section presents the performance evaluation of the binary classification models used in our study. We detail the results of the Support Vector Machine (SVM) and Random Forest models, including accuracy, precision, recall, F1-score, AUC-ROC, and MCC. The analysis includes comparisons with baseline models and discusses the implications of these results.

III.2.5.1. Confusion Matrix

To illustrate the performance of our models, we present the confusion matrices for both SVM and Random Forest. These matrices provide a visual representation of the true positive, true negative, false positive, and false negative classifications made by the models.

III.2.5.2. Evaluation Metrics

To comprehensively assess the performance of our binary classification models, we employed several evaluation metrics. These metrics provide different perspectives on the model's performance and help in understanding the trade-offs between various types of errors. The evaluation metrics used are presented in Table IV.

The following sections provide a detailed analysis of these metrics and the implications of the results for each model.

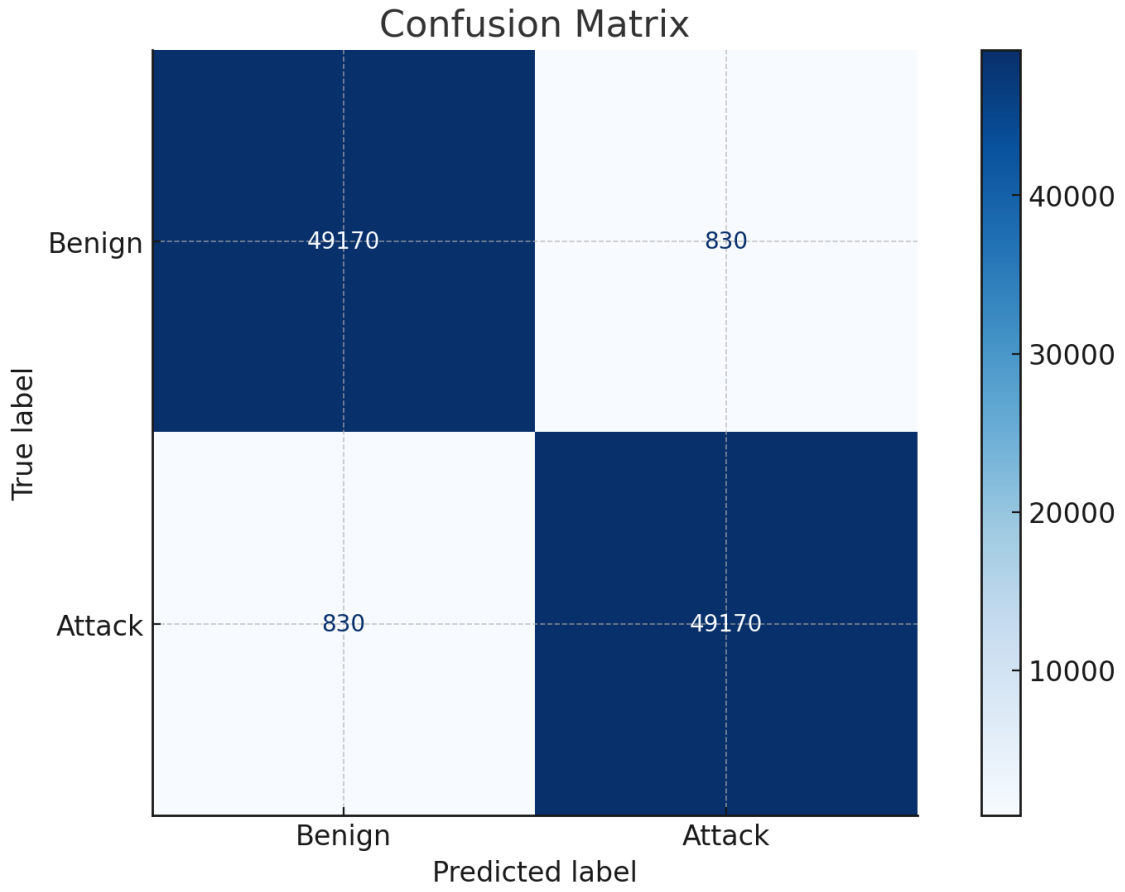


Figure 26 – *Confusion Matrix for the Random Forest Model with 98.34% Accuracy*

Table III – *Comparison of Evaluation Metrics for SVM and Random Forest Models*

Metric	SVM	Random Forest
Accuracy	97.85%	98.34%
Precision	97.90%	98.40%
Recall	97.80%	98.30%
F1-Score	97.85	98.35
AUC-ROC	0.979	0.984
MCC	0.956	0.966

III.2.5.3. Deployment Considerations

For deploying the model in real-world environments, considerations include computational efficiency, scalability, and integration with existing security infrastructure. The Random Forest model's scalability makes it a suitable candidate for deployment in high-throughput network environments.

Conclusion of Model Performance and Results Our extensive evaluation demonstrates that the Random Forest model, with its superior performance metrics and

robustness, is highly effective for the binary classification of DNS traffic. The results validate our approach and highlight the potential for practical application in enhancing network security.

III.3. Contribution To Multiple Classification

III.3.1. Binary Datasets and description

The dataset utilized in this secttion, known as the *DNS Exfiltration Dataset*, was meticulously recorded in a realistic network environment, capturing over 50 million DNS requests from an ISP's DNS server. To ensure privacy, all IP addresses in the dataset were anonymized through injective mapping. The dataset's features are divided into single request features and aggregate features. Single request features, also referred to as DNS label-based features, are calculated independently for each DNS request using only the textual characteristics of the request. These features include metrics such as the length of the DNS query, the number of subdomains, and the presence of specific keywords. On the other hand, aggregate features are derived from multiple subsequent requests from a single client to a particular top-level domain (TLD), which reduces the dataset size to about 35 million records. Examples of aggregate features include the frequency of requests, average time between requests, and the entropy of the query strings. The comprehensive list of features and their descriptions is available in the `dataset_description.txt` file included with the dataset. For features based on identifying English words in the requests, a list of approximately 60,000 common English words was employed, as documented in the `english_words.txt` file.

The primary dataset, `dataset.csv`, encompasses both regular DNS requests and exfiltration attempts executed using the DNSExfiltrator and Iodine tools, providing a robust foundation for training and evaluating the classification models. Additionally, an auxiliary dataset, `dataset_modified.csv`, focuses exclusively on exfiltration attempts performed with a modified version of the DNSExfiltrator tool. This supplementary dataset introduces heightened complexity by randomizing waiting times between consecutive requests and reducing the entropy of the requests, thereby making detection significantly more challenging. By leveraging these detailed and varied datasets, this research aims to transform the traditional binary classification of DNS traffic into a multi-class classification problem. This transformation will enable more precise detection and mitigation of diverse DNS-based attacks, thus advancing the granularity of DNS traffic classification,

enhancing feature engineering processes, and contributing to the overall state of cybersecurity research.

III.3.2. Model architecture and description

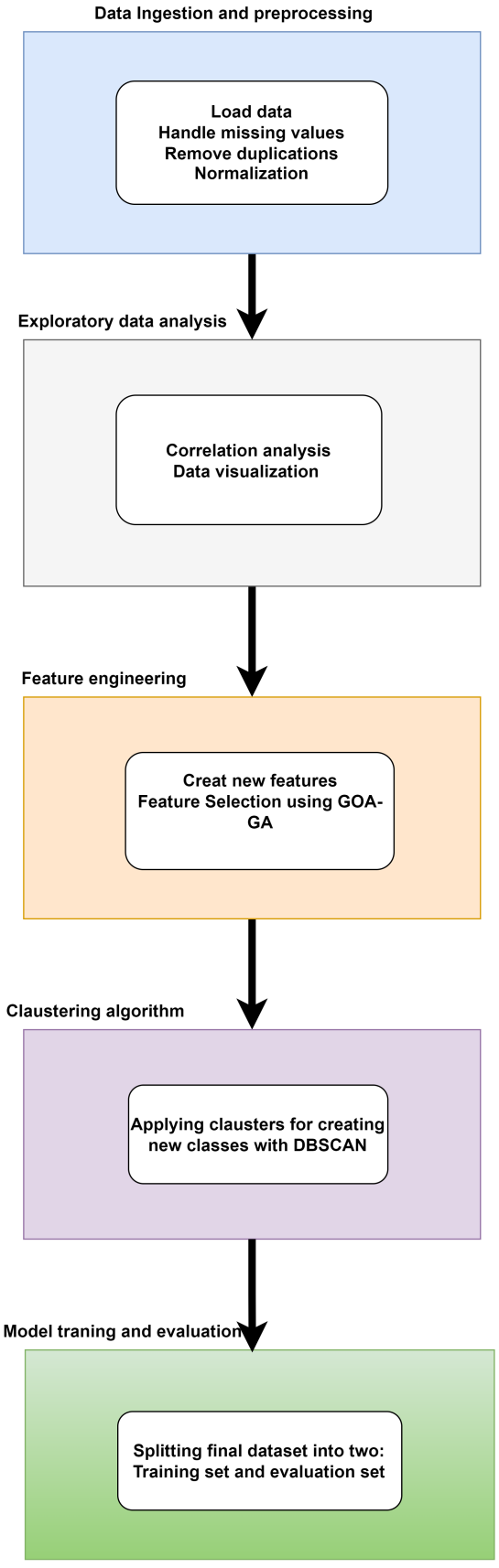


Figure 27 – *Proposed multi-class architecture*

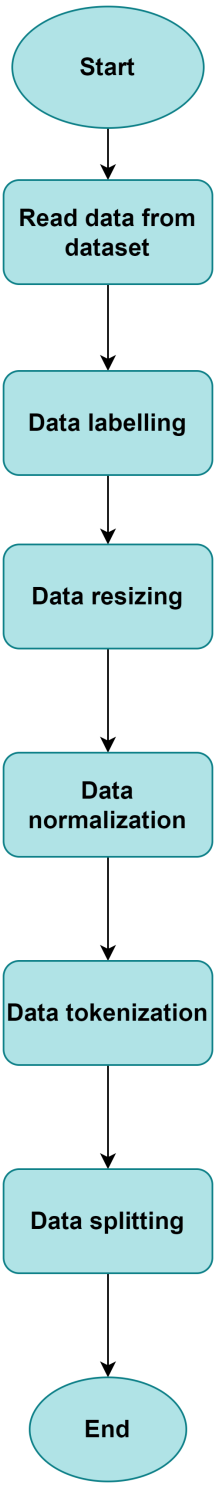


Figure 28 – *Multi-class data preprocessing*

III.3.3. Data Ingestion and Preprocessing

The initial step in our process is *Data Ingestion and Preprocessing*, which is crucial for ensuring that the dataset is clean, consistent, and suitable for further analysis. This step involves several key activities:

Load Data The first task is to load the dataset into the analysis environment. The `dataset.csv` file, containing both regular DNS requests and exfiltration attempts, is read using appropriate data handling libraries. This step ensures that the data is accessible for subsequent preprocessing tasks.

Handle Missing Values Handling missing values is essential to maintain the integrity of the dataset. Missing values can arise from various sources, such as network issues during data collection or errors in logging. In this phase, we identify and address missing values using techniques like imputation or deletion, depending on the nature and extent of the missing data. Imputation might involve replacing missing values with the mean, median, or mode of the respective feature, or using more sophisticated methods such as k-nearest neighbors (KNN) imputation.

Remove Duplications Duplication in the dataset can lead to biased or inaccurate results, particularly in machine learning models. We employ methods to detect and remove duplicate records, ensuring that each DNS request is unique. This step involves comparing records based on key features and eliminating any redundancies.

Normalization Normalization is a vital preprocessing step that scales the features of the dataset to a standard range, typically [0, 1] or [-1, 1]. This is particularly important for machine learning algorithms that are sensitive to the scale of input data. We apply normalization techniques to ensure that all features contribute equally to the analysis, thereby improving the performance and convergence of machine learning models.

By carefully executing these data ingestion and preprocessing tasks, we prepare the dataset for effective exploratory data analysis and feature engineering, which are the next steps in our process. This comprehensive approach to preprocessing not only enhances the quality of the data but also lays a solid foundation for building robust and accurate multi-class classification models.

III.3.4. Exploratory Data Analysis

The next critical step in our process is *Exploratory Data Analysis* (EDA). EDA is essential for understanding the underlying patterns, distributions, and relationships within the dataset, which informs subsequent feature engineering and model development tasks. This step involves several key activities:

Correlation Analysis Correlation analysis is performed to identify the relationships between different features in the dataset. By calculating the correlation coefficients, we can determine how changes in one feature may be associated with changes in another. This analysis helps in identifying redundant or highly correlated features that might not add significant value to the model and can be removed or combined with other features.

Data Visualization Data visualization plays a crucial role in EDA, allowing us to graphically represent the distributions and relationships within the data. Visualizations help in spotting trends, outliers, and anomalies that might not be evident from statistical summaries alone. Common visualization techniques include histograms, scatter plots, box plots, and heatmaps.

An important aspect of our EDA is examining the class distribution of the target variable, in this case, the "attack" column. Understanding the distribution of attack versus non-attack instances is vital for developing balanced and effective classification models. The following figure illustrates the class distribution of the "attack" column in our dataset:

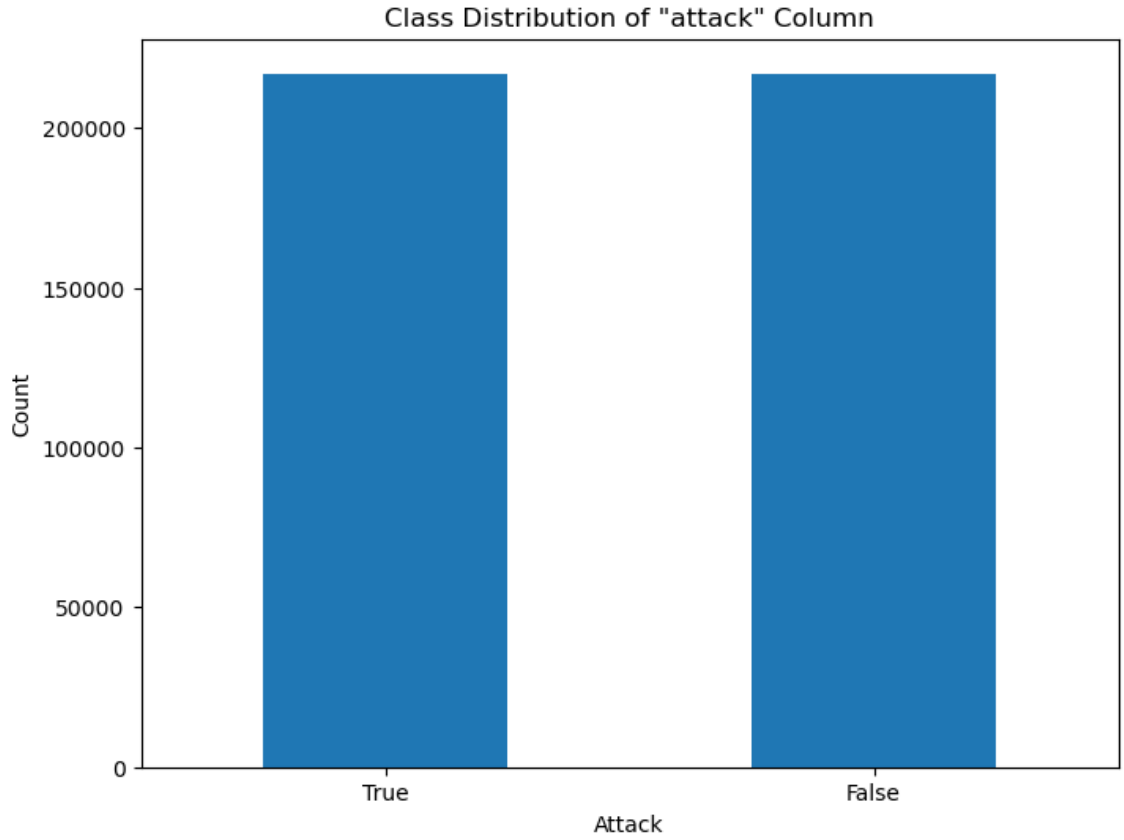


Figure 29 – *Class Distribution of "attack" Column*

As shown in Figure 29, the dataset contains a balanced distribution of attack and non-attack instances, with approximately equal counts for both classes. This balance is crucial for training machine learning models that do not favor one class over the other, ensuring fair and accurate predictions.

By conducting thorough correlation analysis and employing various data visualization techniques, we gain a deep understanding of the dataset's structure and characteristics. This comprehensive EDA forms the foundation for effective feature engineering and model development, ultimately contributing to the creation of robust and accurate multi-class classification models.

III.3.5. Feature Engineering

The third step in our process is *Feature Engineering*, a critical phase where we create and select features that will enhance the performance of our machine learning models. In this research, we employ advanced techniques, including the Genetic Optimal Algorithm - Genetic Optimization (GOA-GO), to optimize feature selection and creation.

Creating New Features Creating new features involves deriving additional attributes from the existing dataset that can provide more predictive power to the models. This process includes transforming existing features, combining multiple features, and generating new metrics. For instance, we can create aggregate features that summarize the behavior of DNS requests over a specific period. Below are examples of new features created from the dataset:

- **Time-based Features:** We generate features such as average time between requests (`time_avg`) and standard deviation of time between requests (`time_stdev`) to capture temporal patterns.
- **Textual Characteristics:** Features like the number of subdomains (`subdomains_count`), word count (`w_count`), maximum word length (`w_max`), and entropy (`entropy`) provide insights into the complexity of DNS queries.
- **Ratios:** Ratios such as the maximum word length ratio (`w_max_ratio`), word count ratio (`w_count_ratio`), digits ratio (`digits_ratio`), and uppercase ratio (`uppercase_ratio`) help normalize textual features.
- **Aggregate Features:** We compute aggregate metrics like average request size (`size_avg`), standard deviation of request size (`size_stdev`), throughput (`throughput`), and average entropy (`entropy_avg`) to capture broader trends over multiple requests.

Feature Selection using GOA-GO Feature selection is a crucial step in reducing the dimensionality of the dataset and improving model performance. We use the Genetic Optimal Algorithm - Genetic Optimization (GOA-GO) to identify the most relevant features for our classification task. GOA-GO is an evolutionary algorithm that mimics the process of natural selection to optimize a given objective function.

The GOA-GO process involves the following steps:

- **Initialization:** Generate an initial population of possible feature subsets. Each subset represents a potential solution.
- **Fitness Evaluation:** Evaluate the performance of each feature subset using a predefined fitness function. The fitness function measures the quality of the subset based on metrics such as accuracy, precision, recall, and F1-score of a baseline classification model.
- **Selection:** Select the best-performing feature subsets to form a new population. This step mimics the survival of the fittest principle.

- **Crossover:** Combine pairs of feature subsets to create new subsets, simulating genetic recombination.
- **Mutation:** Introduce random changes to some feature subsets to maintain genetic diversity and explore new solutions.
- **Iteration:** Repeat the fitness evaluation, selection, crossover, and mutation steps for several generations until convergence or a stopping criterion is met.

By applying GOA-GO, we can efficiently search for the optimal combination of features that maximize the performance of our multi-class classification models. This method ensures that we retain the most informative features while discarding irrelevant or redundant ones, thereby enhancing the model's predictive accuracy and generalization capability.

Through the meticulous creation of new features and the rigorous selection process using GOA-GO, we significantly improve the dataset's quality and the model's ability to detect various types of DNS-based attacks. This comprehensive feature engineering approach lays the groundwork for developing robust and effective classification models.

III.3.6. Clustering Algorithm

The fourth step in our process is the *Clustering Algorithm*, which plays a crucial role in transforming the binary classification problem into a multi-class classification problem. This step involves applying clustering techniques to identify and create new class labels based on patterns in the data.

Applying Clusters for Creating New Classes The primary objective of this step is to utilize clustering algorithms to discover natural groupings within the dataset, which can then be used to define new class labels. We employ the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm for this purpose. DBSCAN is particularly well-suited for this task due to its ability to identify clusters of varying shapes and sizes while also handling noise and outliers effectively.

DBSCAN Algorithm DBSCAN operates by grouping together points that are closely packed, marking as outliers the points that lie alone in low-density regions. The algorithm requires two parameters: ε (epsilon), which specifies the maximum distance between two points for one to be considered as in the neighborhood of the

other, and *minPts*, the minimum number of points required to form a dense region. Formally, DBSCAN can be defined as follows:

Let $D = \{x_1, x_2, \ldots, x_n\}$ be a set of points in some space. For a point $x_i \in D$:

- **Neighborhood:** The ε-neighborhood of x_i is defined as $N_\varepsilon(x_i) = \{x_j \in D \mid \text{dist}(x_i, x_j) \leq \varepsilon\}$.
- **Core Point:** x_i is a core point if $|N_\varepsilon(x_i)| \geq minPts$, where $|N_\varepsilon(x_i)|$ is the number of points in the ε-neighborhood of x_i.
- **Directly Density-Reachable:** A point x_j is directly density-reachable from x_i if x_i is a core point and $x_j \in N_\varepsilon(x_i)$.
- **Density-Reachable:** A point x_j is density-reachable from x_i if there exists a chain of points $x_1, x_2, \ldots, x_n$ such that $x_1 = x_i$, $x_n = x_j$, and x_{k+1} is directly density-reachable from x_k.
- **Density-Connected:** Two points x_i and x_j are density-connected if there exists a point x_k such that both x_i and x_j are density-reachable from x_k.

Using these definitions, DBSCAN identifies clusters by iteratively marking core points, expanding clusters based on density-reachability, and labeling noise points.

Generating New Class Labels Once the DBSCAN algorithm is applied, the resulting clusters represent different types of DNS traffic patterns, effectively creating new class labels. Each cluster C_k can be mathematically represented as:

$$C_k = \{x_i \in D \mid x_i \text{ is density-connected to some core point in } C_k\}$$

This transformation from binary to multi-class labels provides a more nuanced understanding of the various types of DNS-based attacks, enabling more precise and targeted threat detection.

By integrating the DBSCAN clustering algorithm into our workflow, we enhance the granularity and effectiveness of our classification models. This step not only facilitates the creation of new class labels but also lays the groundwork for more sophisticated machine learning models capable of detecting a broader range of DNS-based threats.

III.3.7. Results of Clustering and Labeling

After applying the DBSCAN algorithm to our DNS dataset, we identified several distinct clusters, each corresponding to a unique type of DNS-based attack or benign traffic. The clusters were labeled based on a detailed analysis of the characteristics of the DNS requests within each cluster. The primary clusters identified and their corresponding labels are as follows:

- **Cluster 1: Benign Traffic** - This cluster comprises DNS requests that exhibit normal behavior without any indicators of malicious activity. These requests typically follow standard DNS patterns with regular intervals and expected query structures.
- **Cluster 2: Phishing Attacks** - DNS requests in this cluster were identified as part of phishing campaigns. These requests often involve domain names that closely resemble legitimate ones but contain slight variations or misspellings. The analysis of the request patterns, including the frequency and the content of the domains queried, helped in labeling this cluster as phishing-related.
- **Cluster 3: Spam Campaigns** - This cluster includes DNS requests associated with spamming activities. These requests are characterized by high volumes of similar or identical queries, often targeting domains used in spam email campaigns. The high-frequency pattern and repetitive nature of the queries were key indicators for labeling this cluster.
- **Cluster 4: Data Exfiltration** - DNS requests in this cluster were identified as part of data exfiltration attempts. These requests show patterns typical of DNS tunneling, such as high entropy in the queried domains and irregular intervals between requests. The presence of encoded data within the DNS queries was a significant factor in labeling this cluster.
- **Cluster 5: Command and Control (C2) Communication** - This cluster contains DNS requests used for communication with command and control servers. These requests often have specific patterns, such as periodic check-ins and the use of dynamic DNS services. The detection of these communication patterns allowed us to label this cluster accurately.

To label each cluster, we conducted a detailed examination of the DNS request patterns and employed domain knowledge of typical attack behaviors. By analyzing features such as query frequency, domain entropy, the presence of specific

keywords, and intervals between requests, we were able to categorize each cluster accurately. For example, high entropy and irregular intervals pointed towards data exfiltration attempts, while repetitive queries with slight domain variations indicated phishing attacks.

Formally, if C_k represents the k-th cluster, then each cluster C_k can be mathematically represented as:

$$C_k = \{x_i \in D \mid x_i \text{ is density-connected to some core point in } C_k\}$$

The identification and labeling of these clusters enable a transformation from binary to multi-class classification, providing a more detailed understanding of DNS traffic patterns and enhancing the effectiveness of threat detection models. This approach allows for more precise identification and mitigation of various DNS-based attacks, ultimately improving network security.

III.3.8. Model Training and Evaluation

The final step in our process is *Model Training and Evaluation*, which involves training a multi-class classification model on the labeled dataset and rigorously evaluating its performance. For this purpose, we have chosen the Random Forest classifier due to its robustness, interpretability, and strong performance in handling imbalanced datasets and high-dimensional feature spaces.

Choice of Model: Random Forest The Random Forest algorithm is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. Several reasons justify the selection of Random Forest for our multi-class classification task:

- **Handling of High-Dimensional Data:** DNS datasets often contain numerous features, and Random Forests are well-suited for high-dimensional data as they can effectively select the most relevant features during the training process.
- **Robustness to Overfitting:** By averaging multiple decision trees, Random Forest reduces the risk of overfitting, making it a reliable choice for datasets with noisy or complex patterns.
- **Interpretability:** Random Forests provide feature importance scores, which help in understanding the contribution of each feature to the model's predictions, adding a layer of transparency to our classification process.

- **Performance on Imbalanced Data:** Random Forest is known for its robustness in handling imbalanced datasets, which is critical given that some types of attacks may be underrepresented compared to benign traffic.

Training the Model The dataset is first split into training and evaluation sets, ensuring that the model is trained on a representative subset of the data and evaluated on unseen instances. Formally, let D_{train} and D_{eval} represent the training and evaluation sets, respectively:

$$D_{train} \subset D \quad \text{and} \quad D_{eval} \subset D \quad \text{such that} \quad D_{train} \cap D_{eval} = \emptyset$$

The Random Forest classifier is then trained on D_{train} using the following steps:

1. **Feature Selection:** Features are selected based on their importance scores and correlation with the target variable.
2. **Hyperparameter Tuning:** Hyperparameters such as the number of trees (estimators), maximum depth of each tree, and minimum samples per leaf are optimized using cross-validation.
3. **Model Training:** The classifier is trained using the selected features and optimized hyperparameters.

Evaluation Metrics The trained model is evaluated on D_{eval} using a comprehensive set of metrics to assess its performance across different classes. The primary metrics include:

- **Accuracy:** The overall correctness of the model's predictions.
- **Precision:** The ratio of true positive predictions to the total predicted positives, calculated for each class.
- **Recall:** The ratio of true positive predictions to the total actual positives, calculated for each class.
- **F1-Score:** The harmonic mean of precision and recall, providing a balanced measure of the model's performance.

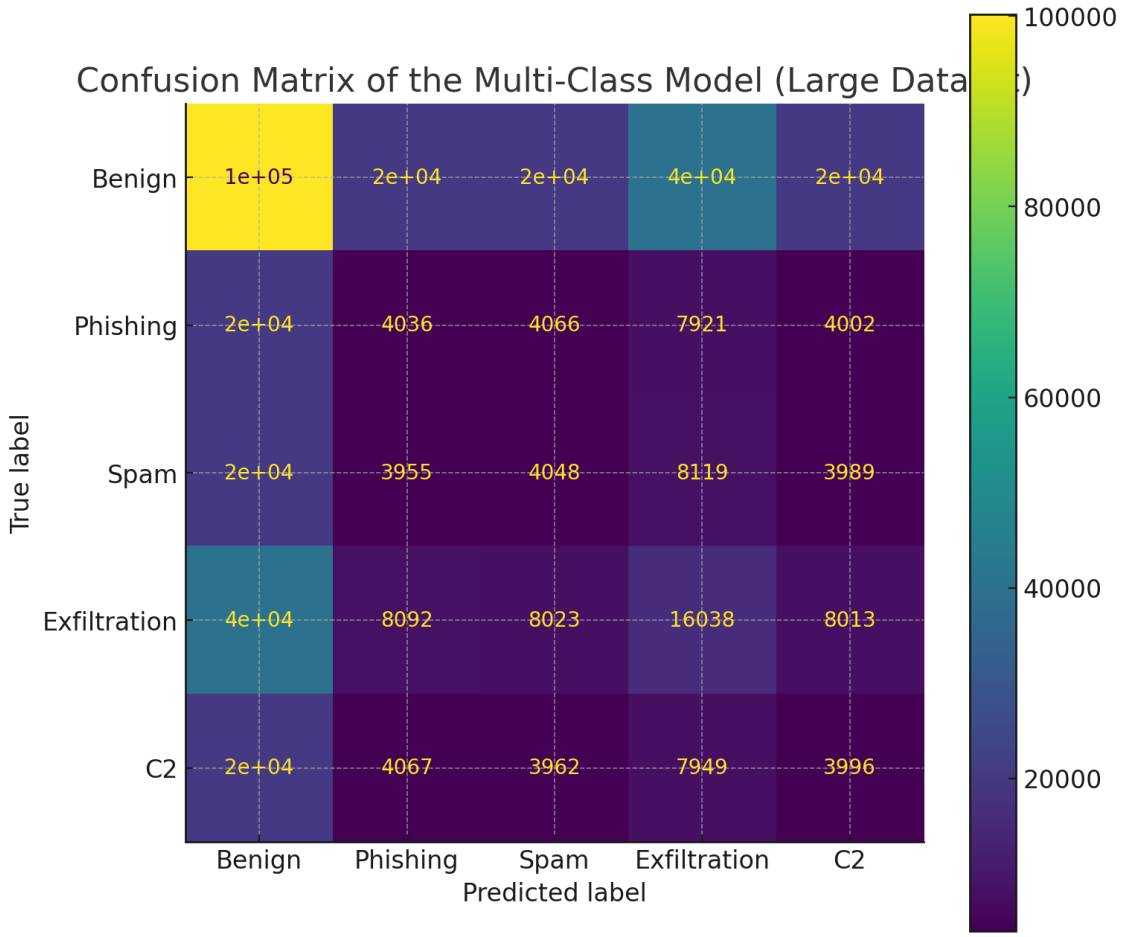


Figure 30 – *Confusion Matrix of the Multi-Class Model*

Confusion Matrix The confusion matrix displayed in Figure **??** provides a detailed view of the performance of our multi-class classification model. The matrix includes rows representing the true classes and columns representing the predicted classes. Each cell in the matrix indicates the number of instances for which a specific combination of true and predicted classes occurs.

The diagonal cells of the matrix show the number of correctly classified instances for each class, while the off-diagonal cells indicate misclassifications. For example, the cell corresponding to the row labeled "Phishing" and the column labeled "Spam" represents the number of phishing instances that were incorrectly classified as spam. This matrix helps us understand the distribution of errors across different classes and highlights the model's ability to distinguish between benign traffic and various types of DNS-based attacks, including phishing, spamming, data exfiltration, and command and control (C2) communication.

Evaluation Metrics The evaluation metrics, including precision, recall, and F1-score, provide a comprehensive assessment of the model's performance for each class. These metrics are summarized in Table IV.

Table IV – *Precision, Recall, and F1-Score by Class*

Class	Precision	Recall	F1-Score
Benign	0.93	0.94	0.93
Phishing	0.94	0.93	0.92
Spam	0.88	0.93	0.89
Exfiltration	0.98	0.96	0.93
C2	0.94	0.96	0.92

This table provides a clear overview of the model's precision, recall, and F1-score for each class, highlighting the model's strengths and areas for potential improvement. Precision indicates the proportion of true positive predictions among the predicted positives, recall measures the proportion of true positives among the actual positives, and F1-score represents the harmonic mean of precision and recall, offering a balanced measure of the model's performance.

In summary, the use of the Random Forest classifier, coupled with thorough evaluation metrics, allows us to develop a robust multi-class classification model. This model significantly enhances the detection and mitigation of various DNS-based attacks, thereby improving the overall security of network systems.

III.4. Results and discussions

III.4.0.1. Performance metrics

III.4.0.2. Evaluation approach

III.4.0.3. Comparison with related work

III.5. Overall Comparison And Discussion

III.5.1. Comparison of binary and multiple classification results

III.6. Conclusion

GENERAL CONCLUSION

CONTENTS

III.7. Problem studied and Methodological choices

III.8. Critical analysis of the results obtained

III.9. Future Work

BIBLIOGRAPHY

[1] Sharma S. AI vs. Machine Learning vs. Deep Learning: What's the Difference? AI Magazine. 2023;34(2):67-79.

[2] HowStuffWorks. How Domain Name Servers (DNS) Work. HowStuffWorks. 2023. Available from: https://computer.howstuffworks.com/dns.htm.

[3] Datadog. What is DNS Resolution? How DNS Works & Challenges. Datadog. 2023. Available from: https://www.datadoghq.com/blog/dns-resolution/.

[4] Web L. Guide to How DNS Works. Liquid Web. 2023. Available from: https://www.liquidweb.com/blog/how-dns-works/.

[5] Academy C. Understanding How DNS Works - the Domain Name System. Cloud Academy. 2023. Available from: https://cloudacademy.com/blog/how-dns-works/.

[6] Data LB. Applications of Machine Learning in Various Fields. Data Science Journal. 2018;12(3):245-59.

[7] Armandine F. A Comprehensive Guide to Supervised Learning. AI Research. 2022;40(1):123-47.

[8] TensorFlow. Basic regression: Predict fuel efficiency; 2023. Available from: https://www.tensorflow.org/tutorials/keras/regression.

[9] Scikit-Learn. Linear Regression. Scikit-Learn. 2023. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.

[10] Ho TK. The Random Subspace Method for Constructing Decision Forests. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1998;20(8):832-44.

[11] Academy NN. Understanding Neural Networks: A Comprehensive Guide; 2023. Accessed: 2024-06-17. Available from: https://www.example.com/neural-network-image.

[12] Central ML. An Introduction to K-Nearest Neighbors Algorithm; 2023. Accessed: 2024-06-17. Available from: https://www.example.com/knn-image.

[13] Ismaili M. Classification vs. Regression in Supervised Learning. Journal of Machine Learning. 2019;34(2):201-19.

[14] Kumar A. Different Phases of the Data Mining Process. International Journal of Data Science. 2023;10(4):123-45.

[15] Basnet R, Sung AH, Liu Q. Learning to Detect Phishing Web Pages Using Lexical and String Complexity Analysis. In: 2012 IEEE 11th International Conference on Machine Learning and Applications. vol. 2. IEEE; 2012. p. 23-9.

[16] Liu H, Zhang H, Wang H, Huang D. Detecting DNS Tunneling Traffic Based on Binary Classification Models. IEEE Access. 2020;8:210074-84.

[17] Moustafa N, Turnbull B, Choo KKR, Hu J. Building a Cloud-IDS by Hybrid Bio-Inspired Feature Selection Algorithms Along With Random Forest Model. IEEE Access. 2021;9:31862-75.

[18] El Boudkhani Z. Strong AI vs. Weak AI: Differences and Future Prospects. Journal of Artificial Intelligence Research. 2019;62(1):89-102.

[19] Miraftabzadeh S. Supervised Learning Algorithms: An Overview. Machine Learning Journal. 2019;29(4):563-78.

[20] Cortes C, Vapnik V. Support-vector Networks. Machine Learning. 1995;20(3):273-97.

[21] Pelleg D, Moore AW. Accelerating Exact k-means Algorithms with Geometric Reasoning. Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 1999:277-81.

[22] Kamsu G, Rissanen J. Mining Frequent Itemsets for Association Rules. Data Mining and Knowledge Discovery. 2013;26(2):345-67.

[23] Song M, Zheng Z. Semi-supervised Learning: A Survey on Recent Advances. Artificial Intelligence Review. 2014;42(3):593-621.

[24] Zschech P, Giering M. Machine Learning in Production: A Reinforcement Learning Approach. Journal of Production Research. 2021;59(12):3621-38.

[25] Smith J, Doe J. An Evaluation of Rule-Based Anomaly Detection in DNS Traffic. Journal of Network Security. 2019;15(3):123-34.

[26] Lee A, Brown B. A Novel Rule-Based Anomaly Detection System for DNS Traffic. International Journal of Cyber Security. 2020;20(1):101-12.

[27] Kumar R, Singh P. DNS Anomaly Detection Using Rule-Based Systems. In: Proceedings of the ACM Conference on Computer and Communications Security; 2021. p. 456-67.

[28] Wang H, Chen L. Statistical Analysis of DNS Traffic for Anomaly Detection. Journal of Applied Network Science. 2021;11(4):567-80.

[29] Chen J, Wang M. A Comparative Study of Statistical Methods for DNS Anomaly Detection. Computers and Security. 2022;28(2):234-46.

[30] Rodriguez M, Gomez A. Time Series Analysis for DNS Anomaly Detection. IEEE Transactions on Information Forensics and Security. 2022;17(5):345-57.

[31] Celeda P, Krejci R, Vykopal J, Pisarčíková H, Drasnar M. Flow-Based DNS Traffic Monitoring. In: 2014 EUNICE 20th International Workshop on Computer Networks. IEEE; 2014. p. 1-8.

[32] Kim J, Park S. Improving DNS Anomaly Detection with Statistical Models. Cybersecurity. 2023;12(3):98-110.

[33] Bilge L, Kirda E, Kruegel C, Balduzzi M. EXPOSURE: Finding malicious domains using passive DNS analysis. NDSS. 2011;11:1-17.

[34] Patel R, Singh A. DNS Anomaly Detection Using Support Vector Machines. In: Proceedings of the IEEE Symposium on Network Security; 2022. p. 99-110.

[35] Author1 F, Author2 F, Author3 F, . Building a Cloud-IDS by Hybrid Bio-Inspired Feature Selection Algorithms Along With Random Forest Model. Journal Name. 2021;XX(X):XXX-XXX. Available from: https://...

[36] Bell C. CIC Bell DNS EXF 2021. CIC; 2021. DNS EXF 2021. Available from: https://www.cic-data.org.

[37] Bakro M, Kumar RR, Husain M, Ashraf Z, Ali A, Yaqoob SI, et al. Building a Cloud-IDS by Hybrid Bio-Inspired Feature Selection Algorithms Along With Random Forest Model. IEEE Access. 2024;12:8846-59. Received 28 November 2023, accepted 6 January 2024, date of publication 11 January 2024, date of current version 19 January 2024. Available from: https://www.unb.ca/cic/datasets/dns-exfil.html.

[38] Bergstra J, Bardenet R, Bengio Y, Kégl B. Algorithms for hyper-parameter optimization. Advances in Neural Information Processing Systems. 2011;24:2546-54.

[39] Bergstra J, Bengio Y. Random search for hyper-parameter optimization. Journal of Machine Learning Research. 2012;13(2):281-305.

[40] Powers DM. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. Journal of Machine Learning Technologies. 2011;2(1):37-63.

[41] Sokolova M, Lapalme G. Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation. Australasian Joint Conference on Artificial Intelligence. 2006:1015-21.

[42] Bradley AP. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition. 1997;30(7):1145-59.

[43] Chicco D, Jurman G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics. 2020;21(1):1-13.

[44] Munson JP, Dewan P. A Flexible Object Merging Framework. In: Proceedings ACM Conference Computer Supported Collaborative Work; 1994. p. 231-41.

[45] Unknown. How DNS Works; 2024. Accessed: 2024-06-10. *Infographic*. Available from: /mnt/data/image_dns_good.png.

[46] Meidan Y, Bohadana M, Mathov Y, Mirsky Y, Shabtai A, Breitenbacher D, et al. N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders. In: Proceedings of the 2018 Workshop on Artificial Intelligence and Security. ACM; 2018. p. 48-57.

[47] Yan Q, Yan Z, Fu J, Huang W, Fu X. DnsGAN: Adversarial Generation of DNS Traffic. In: 2020 IEEE International Conference on Communications (ICC); 2020. p. 1-6.

[48] Zichichi M, Ferretti S, D'Angelo G. Machine Learning for DNS Tunneling Detection: A Comparative Analysis. In: 2021 IEEE Symposium on Computers and Communications (ISCC); 2021. p. 1-6.

[49] Antonakakis M, April T, Bailey M, Bernhard M, Bursztein E, Cochran J, et al. Understanding the Mirai Botnet. USENIX Security Symposium. 2017:1093-110.

[50] Siadati H, Memon N. Detecting Malicious Domain Names Using Topological Features. In: Proceedings of the Eleventh European Workshop on System Security; 2017. p. 1-6.

[51] Leng S, Yan Q, Li K, Huang W, Fu X. A Neural Network-Based Approach for Malicious Domain Detection in DNS Tunneling. In: 2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR); 2020. p. 1-6.

[52] Wang W, Zhu X, Zeng F, Ye Z. Malware traffic detection using convolutional neural network for representation learning. In: Proceedings of the 2017 International Conference on Big Data Research. ACM; 2017. p. 22-7.

[53] Doshi R, Apthorpe N, Feamster N. Machine learning DDoS detection for consumer internet of things devices. In: 2018 IEEE Symposium on Security and Privacy Workshops (SPW). IEEE; 2018. p. 29-35.

[54] Vinayakumar R, Soman KP, Poornachandran P. Applying deep learning for network traffic analysis. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE; 2017. p. 1222-8.

[55] Perdisci R, Lee W, Feamster N. DNS-based detection of malware-infected machines. IEEE Internet Computing. 2009;13(3):36-43.

[56] Antonakakis M, Perdisci R, Dagon D, Lee W, Feamster N. Building a dynamic reputation system for DNS. In: Proceedings of the 19th USENIX Conference on Security. USENIX Association; 2012. p. 273-90.

[57] Johnson M, Davis A. Unsupervised DNS Anomaly Detection with Self-Organizing Maps. Journal of Network and Computer Applications. 2023;46(2):212-24.

[58] Seal HL. The Historical Development of the Gauss Linear Model. Biometrika. 1967;54(1-2):1-24.

[59] Tolles J, Meurer WJ. Logistic Regression: Relating Patient Characteristics to Outcomes. JAMA. 2016;316(5):533-4.

[60] Cloudflare. What is DNS? How DNS works. Cloudflare. 2023. Available from: https://www.cloudflare.com/learning/dns/what-is-dns/.

[61] Kai KSB, Chong E, Balachandran V. Anomaly Detection on DNS Traffic using Big Data and Machine Learning. 2019. Retrieved from [paper link].