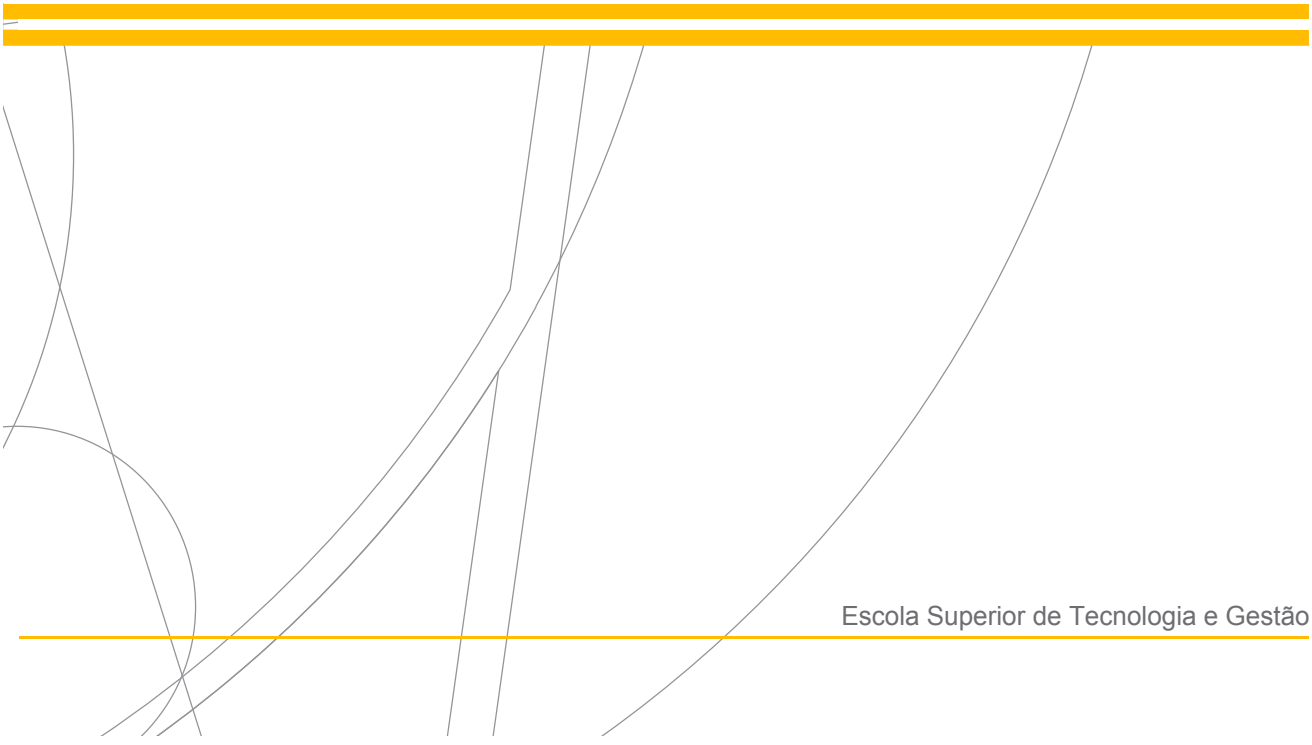
Instituto Politécnico
de Viana do Castelo

DNS FIREWALL BASED ON MACHINE LEARNING

Cláudio Marques



Escola Superior de Tecnologia e Gestão

Instituto Politécnico
de Viana do Castelo

Cláudio Roberto Cunha Marques

DNS FIREWALL BASED ON MACHINE LEARNING

Nome do curso de Mestrado

Mestrado em Cibersegurança

Trabalho efetuado sob a supervisão de

Professor João Paulo Ferreira de Magalhães

Professor Silvestre Lomba Malta

Dezembro de 2021

Mestrado em Cibersegurança
Master in Cybersecurity

DNS Firewall based on Machine Learning

a master's thesis authored by

Cláudio Roberto Cunha Marques

and supervised by

João Paulo Ferreira de Magalhães

IPP

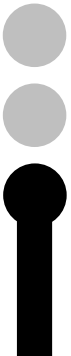
Silvestre Lomba Malta

IPVC

This thesis was submitted in partial fulfilment of the requirements for the Master's degree in Cybersecurity at the Instituto Politécnico de Viana do Castelo

ipvc

13 of December, 2021



Abstract

Nowadays there are many Domain Name Service (DNS) firewall solutions to prevent users to access malicious domains. These can provide real-time protection and block illegitimate communications. Most of these solutions are based on known malicious domain lists that are being constantly updated. However, in this way, it is only possible to block malicious communications for known malicious domains, leaving out many others that are malicious but have not yet been updated in the blocklists.

This work intends to provide a DNS firewall solution based on Machine Learning (ML) to improve the detection of malicious DNS requests on the fly. For this purpose, a dataset with thirty-four features and 90000 records was created based on real DNS logs. The data will be enriched using Open Source Intelligence (OSINT) sources. The exploratory analysis and data preparations steps were carried and the final dataset submitted to different Supervised ML algorithms to accurately and timely classify if a domain request is malicious or not.

The results show that the ML algorithms were able to classify the benign and malicious domains with accuracy rates between 89% and 96% and the time to test between 0.01 and 3.37 seconds which provides a valuable register to the scientific community which can be applied in firewall systems in order to increase the security analysis and performance.
Keywords: Cybersecurity. DNS. Firewall. Machine Learning.

Resumo

Hoje em dia existem muitas soluções de firewall DNS (Sistema de Nomes de Domínio) para prevenir os utilizadores de acederem a domínios maliciosos. Estas podem fornecer proteção em tempo real e bloquear comunicações ilegítimas. A maioria destas soluções são baseadas em listas de domínios maliciosos já conhecidos que estão em constante atualizacão. No entanto, desta forma, só é possível bloquear comunicações maliciosas para domínios maliciosos já conhecidos, deixando de fora muitos outros que são maliciosos mas ainda não foram atualizados nas listas de bloqueio.

Este trabalho pretende fornecer uma solução de firewall DNS baseada em Machine Learning (ML) para melhorar a detecção de pedidos maliciosos ao DNS em tempo real. Para isso, um conjunto de dados com trinta e quatro características e 90000 registos foi criado com base em logs de DNS reais. Os dados foram enriquecidos usando fontes abertas (OSINT). As fases de análise exploratória e preparação de dados foram realizadas e o conjunto de dados final foi submetido a diferentes algoritmos de ML supervisionados para classificar de forma precisa e oportuna se um domínio pedido ao serviço de DNS é malicioso ou não.

Os resultados mostram que os algoritmos de ML foram capazes de classificar os domínios benignos e maliciosos com taxas de precisão entre 89% e 96% e o tempo de teste entre 0,01 e 3,37 segundos, o que fornece um registo valioso para a comunidade científica que pode ser aplicado em sistemas de firewall para melhorar o desempenho e a análise de segurança.

Palavras-chave: Cibersegurança, DNS, Firewall, Machine Learning.

Aknowledgements

I would like to thank my supervisor at Polytechnic of Porto, Professor Dr. João Magalhães, my co-supervisor Professor Silvestre Malta from Polytechnic Institute of Viana do Castelo for all guidance provided throughout the thesis. I am thankful for all their support and ideas. I am very grateful for their constant availability for discussing the thesis direction and progress. I appreciate all the teachers of the Master's course in Cybersecurity at the Polytechnic Institute of Viana do Castelo for their excellent curriculum content, initiatives over these two years and a great transfer of knowledge.

Finally, my heartfelt thanks to my parents, Domingos and Manuela, my brother and sister, Carlos and Helena, as well as my girlfriend, Liliana, for supporting me this whole time and for all their encouragement.

Contents

List of Figures

List of Tables

List of Abbreviations

ASN Autonomous System Number

AutoML Automated Machine Learning

CART Decision Tree

CSV Comma-separated Values

CWMP CPE WAN Management Protocol

DDOS Distributed Denial of Service

DGA Domain generation algorithm

DKIM Domain Keys Identified Email

DMARC Domain-Based Message Authentication Message Conformance

DNS Domain Name Service

DNSSEC DNS Security Extensions

DoS Denial of Service

FTP File Transfer Protocol

GDPR General Data Protection Regulation

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IANA Internet Assigned Numbers Authority

IMAP Internet Message Access Protocol

IMAPS Internet Message Access Protocol over Secure Sockets Layer

IP Internet Protocol

IT Information Technology

KNN K-Nearest Neighbors

LDA Linear Discriminant Analysis

LR Logistic Regression

ML Machine Learning

MSA Mail Message Submission Agent

MX Mail Exchanger

NB Naive Bayes

OS Operative System

OSINT Open Source Intelligence

POP3 Post Office Protocol version 3

POP3S Post Office Protocol version 3 over Secure Sockets Layer

RBF Radial basis function

RE Reverse Engineering

RFC Request For Comments

RFE Recursive Feature Elimination

RPZ Response Policy Zone

RSA Rivest-Shamir-Adleman

SANS SANS Internet Storm Center

SMB Server Message Block

SMTP Simple Mail Transfer Protocol

SMTPS Simple Mail Transfer Protocol over Secure Sockets Layer

SNMP Simple Network Management Protocol

SPF Sender Policy Framework

SSH Secure Shell

SVM Support-Vector Machine

TCP Transmission Control Protocol

TLD Top Level Domain

TTL Time To Live

TXT Text

UDP User Datagram Protocol

VPN Virtual Private Network

Chapter 1

Introduction

1.1 Context

The Domain Name Service (DNS) is a fundamental service to the functioning of the Internet. The name servers keep the mapping between a logical name and the IP address of the servers so that all the communications start by contacting the DNS to obtain the IP address before accessing the desired service.

As the Internet grows, so does the number of DNS domains. According to the report provided by Verisign [76], the second quarter of 2020 closed with 370.1 million new domain name registrations across all Top Level Domain (TLD). An increase of 0.9 percent when compared to the first quarter of 2020 and a growth of 4.3 percent, year over year. Unfortunately, this growth has a less positive side. In [58] it is said that 70% of newly 200 000 registered domains every day are malicious or suspicious. According to the same study, these new domains just stay active for a very brief period, just hours, while others are quickly spotted behaving as command and control servers or distributing malware, phishing attacks, or used for typo squatting.

According to Netsurion [5], a provider of cloud-managed Information Technology (IT) security services, there are six different types of DNS attacks:

- Malware installation: A common approach to get users to install malware is based on DNS queries hijacking which allows inserting malicious Internet Protocol (IP) addresses or domains in the DNS response. This type of attack continues to grow and the 2021 Check Point mid-year security report [4] points to an increase of 93%

of carried ransomware attacks when compared with the first half of 2021 and the same period of the past year.

- Credential theft: For credential theft, an adversary can create a malicious domain based on a legitimate domain name and perform phishing attacks. According to the FBI internet crime report of 2020 [28], phishing victims doubled from 114 702 in 2019 to 241 324 in 2020 and tend to grow in the current year making this attack the most common type.

- Command & Control communication: The command & control communication occurs after an initial compromise where the DNS traffic is manipulated in order to establish a communication with a command and control server. The botnet malware is a common application of this attack. The adversary can remotely control multiple machines which can lead to credential and data theft or distributed attacks such as Distributed Denial of Service (DDOS). Only in the second quarter of this year, the Spamhaus Malware Labs [68] observed more than 1000 botnet command and control servers with the ".com" being the top used TLD.

- Network footprinting: DNS queries can be made to build a network map of the victim. There are two types of footprinting: passive and active. On passive, the data collection has origin only in open available sources. On active, there is a direct interaction with the target to obtain information. The abusive use of PTR, SOA, and AXFER queries are a good indicator of compromise for this type of attack.

- Data theft: For the data theft, there is an abuse of DNS traffic to transfer data. Commonly the adversary makes use of tunneling from others protocols such as FTP or SSH through DNS queries and responses. Tunneling allows the attacker to execute remote malicious commands (command and control) and malware installation on the victims machine.

- Denial of Service (DoS): In this type of attack, the malicious actors flood a remote server or website making it temporarily unavailable. The use of DNS amplification techniques allow attackers to turn small queries into larger payloads capable of bring down the target server.

The last Global DNS Threat Report [35] found that, globally, 87% of the surveyed organizations suffered from DNS attacks with an average cost of 779 008€. In the report there are two major recommendations to protect against these attacks: securing network endpoints and DNS traffic analysis. According to the suggestions of the report, the DNS must be the first line of defense both for the protection of an organization and to stop the spread of attacks.

1.2 Problem Statement and Motivation

Most of the DNS firewall solutions are based on blocklists of malicious domains. When a request is sent to these systems it is filtered and a query is produced to check the veracity of the domain. The blocklists are constantly updated, but if a new malicious domain is not yet in the list the request continues the flow without any filter leaving the user/organization exposed to potential cyberattacks. Another approach to malicious domains detection is the DNS passive analysis. Although DNS traffic is in real-time, the passive analysis makes use of stored DNS registries. This method consists of the analysis and exploration of DNS logs to find patterns and extract valuable information in order to detect malicious domain names.

In [77], the author collected and analyzed real-world DNS data to detect malicious behavior and was able to detect different patterns such as unauthorized name servers and botnet controllers. Also, based on passive DNS analysis, the study [48] focuses on the passive analysis of recursive DNS traffic traces collected from multiple large networks. It was analyzed more than 2500 million DNS queries and the proposed approach was able to accurately detect malicious flux service networks which can benefit spam filtering applications.

Considering the number of malicious domains and their short lifetime, the filtering based on blocklist and passive DNS analysis is not enough to mitigate the problems. To cope with the challenge of detecting attempts to communicate with malicious domains in a timely manner, we propose the study and implementation of a DNS firewall solution based on Machine Learning (ML). There are previous studies that also detect malicious domains on the fly such as [19], [47] and [59]. However, as described in the next section, we

will explore a different approach in order to detect malicious domains with high accuracy rates and in a timely manner.

1.3 Objectives

The main objective of our work is to implement a firewall solution capable of classifying DNS requests and filtering them regarding the risk of being malicious or not. This objective presupposes the implementation of a set of steps, all of them important. Namely:

- Due to the nonexistence of a DNS dataset, a list of malicious and non-malicious domains needs to be selected;
- From the list of domains, other features that can add value to the analysis should be derived. This process consists on the data enrichment and involves the use of Open Source Intelligence (OSINT) sources and with extra fields like the entropy verified in the domain name and WHOIS information;
- The resulting dataset must be analyzed and prepared before it is submitted to ML algorithms. This step includes an exploratory data analysis to ascertain the existence of missing data and extreme values. It also involves the processing of data in order to obtain a dataset that can be analyzed using ML algorithms;
- The ML algorithms should be selected according the type of analysis to be done (e.g. supervised versus unsupervised learning);
- The features importance should be accessed to maximize the accuracy while reducing the time to get the results;
- Conduct different experimental analysis should be done to determine which algorithm(s) is most suitable for further use;
- Conclude on the effectiveness of the solution and on its possible use in in-band or out-band mode to detect and mitigate problems related to communication with malicious domains.

The creation of a DNS Firewall capable of detecting and blocking malicious communications in an accurate and timely manner is very important. Since the DNS is the basis for

establishing communications, detect and interrupting malicious communications at their source, independently from attack vector, will prevent the escalation of cyber-attacks, thus protecting organizations from the economic and reputation damages associated with this kind of attacks.

1.4 Organization

This document is organized as follows. Chapter 2 describes the work carried out by other researchers. Chapter 3 presents the proposed methodology to implement the DNS firewall solution. Chapter 4 focus on the DNS dataset creation and analysis. Chapter 5 presents the data analysis through data preparation, ML classification models used on this work, and the results. Chapter 6 concludes the thesis.

Chapter 2

State of the art

The DNS is the central element of the internet and a widely know entry point for cyber-attacks. According to International Data Corporation's 2020 threat report [34] almost 80% of cybersecurity issues involve a malicious DNS query.

The DNS is responsible to map domain names to an IP address and make use of a caching system to store a domain name with a defined Time To Live (TTL). Nowadays millions of devices are using DNS servers to be able to navigate on the internet. The DNS servers communication are made in unique and unsigned packets[25] making these a target to attackers which are able to exfiltrate data from a victim using data encapsulation techniques over DNS requests. According to [15] there are numerous vulnerabilities related to the DNS such as man in the middle attacks, packet sniffing, transaction ID guessing, caching problems, cache poisoning, and DDOS attacks. Some threats such as phishing attacks and malware installation might depend on a DNS request [13] either to establish a communication between a victim and the command and control server or to simply create a malicious domain to promote malware dissemination and data theft. Therefore, there is a need to improve the DNS security to ensure secure DNS communications.

Motivated by the many vulnerabilities and a historical record of attacks on DNS systems, two major security improvements were made: the use of DNS Security Extensions (DNSSEC) and the DNS firewall solutions.

2.1 DNSSEC

The DNSSEC is defined by several Request For Comments (RFC) such as the RFC 4033 [54] where is established a standard to add data origin authentication and data integrity to the DNS using cryptographic digital signatures based on public-key cryptography. Although the DNSSEC is present for a long time and 90% of the TLD has the DNSSEC implemented and enabled [33], according to [12] "only 1% of the .com, .org, and .net domains attempt to deploy DNSSEC".

According to [7], the DNSSEC provides protection against spoofing of DNS data and also man in the middle attacks by adding a layer of authentication to the traditional DNS. There is some complexity in the implementation of a DNSSEC solution as mentioned in [14]. This complexity can lead to a bad configuration which can result in vulnerabilities in the DNSSEC server. In [75] the authors gathered 282766 domain names and 4% of the domains with DNSSEC implemented showed a form of misconfiguration. Furthermore, for the misconfigured domain names 73.86% were unable to be reached by a DNSSEC resolver. In a more recent study about vulnerable domain names over DNSSEC [12] the authors were able to find multiple domain names with issues on the key generation mechanism such as the use of shared keys to sign more than one domain and the use of short (weak) Rivest-Shamir-Adleman (RSA) keys. From a dataset of 1.9M registries, the authors conclude that 35% are using RSA keys shared with other domains and 66% uses too short keys (1024 bit or less).

According to [15] there are some different vulnerabilities to the DNSSEC such as the chain of trust (root public key injection), security considerations for key management, key rollovers, zone private key storage, and DNSSEC timing issues.

There are recent studies to improve DNSSEC security [22] [39] which aims to add features such as queries confidentiality using encryption between DNSSEC servers.

Since there is not a silver bullet for the DNS security, all proposals must be taken into account and the implementation of DNSSEC solutions brings more advantages than disadvantages.

2.2 DNS firewall solutions

The DNS based attacks such as the use of botnets are one of the biggest security threats allowing the combined power of multiple remotely controlled machines to perform an attack increasing significantly the success and damage probability. To avoid the takedown of botnets by the law enforcement authorities, malicious actors make use of Domain generation algorithm (DGA). This technique allows the generation of random domain names that change over time. According to [49] there are four methods in which the DGA are based: arithmetic, hash, wordlist and permutation. These methods take a seed as an input, outputs a string, and appends a TLD. Due to the randomness of these domains it is difficult to determine if a given domain name is malicious or not. Reverse Engineering (RE) is used to verify how these domains are created and then sinkhole the communications, although RE is time consuming leaving the domains operational for large periods. In this scenario the application of ML to improve the classification of malicious and non-malicious domains is essential.

In [53], a ML framework is proposed to identify DGA domain names based on deep neural networks. The results are encouraging reaching 98.70% on average for the F1 score and a precision of 83%. With a ML model trained to detect malicious domains, created using DGA methods or not, it is a valuable complement to actual firewall systems.

In [59] the authors present a prototype of a DNS firewall with flexible response control. The prototype analyses the DNS queries done by the DNS clients to the university DNS server and manipulates the responses of queries from attackers based on the analysis. A more recent study presented in [42] combines blocklists / allowlists with a ML approach on DNS traffic. A deep neural architecture model was trained using passive DNS database. This study was able to detect if a domain is benign, malign, or a sinkhole with 95% of accuracy on malicious and a false positive rate of 1:1000. The study also uses different algorithms and presents a comparison table for each one.

In [30] authors used nine features of botnet domain querying and the Random Forest classifier algorithm to pick the malicious domains out of DNS traffic and were able to reach the 99.38% of accuracy after four optimizations.

In [19] the authors proposed the EXPOSURE, a system that employs passive DNS

analysis techniques to detect malicious domains. They build a classifier model based on Decision Tree algorithm (J48), working on real-time over 15 features and reaching 98% of accuracy on a 10 fold cross-validation. Another DNS based approach was proposed in [47] and combines the domain features, host-based features, and web-based features to identify malicious domains. Using a small set of domains (10000), the authors could reach 60% of accuracy, which can be certainly improved when applied to larger datasets.

The use of DNS Response Policy Zone (RPZ) to block malicious domains is still a useful approach as demonstrated in [81]. Using ML algorithms, such as the logistic regression classification algorithm, to actively identify possible threats at DNS level as presented in [47] combined with a DNS RPZ approach can be valuable and improve the detection of malicious domains.

Although there are numerous studies on DNS firewall systems, the need to analyze and predict newly malicious domains over DNS communications in a timely manner is a valuable cybersecurity countermeasure to mitigate, prevent and alert both organizations and end-users against malicious DNS traffic.

The solution proposed and the methodology adopted in this study is presented in the next chapter.

Chapter 3

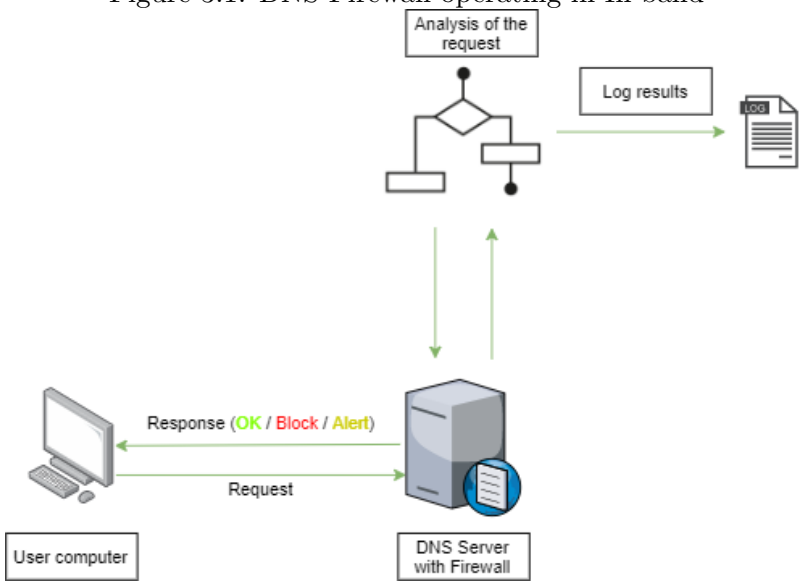
Proposal and Methodology

A DNSfirewall system as we propose can be used to complement the DNS firewall systems based on block/allow lists with a ML model capable of verifying if a domain is malicious or not. In this way, it is possible to block access to domains that are already recognized as malicious and to others that are not yet cataloged as such, but which have a high probability of being so.

The proposed firewall solution will be built in two distinct operation modes: in-band and out-band. The in-band mode makes use of active DNS analysis and the out-band mode follows the passive DNS analysis approach as described below:

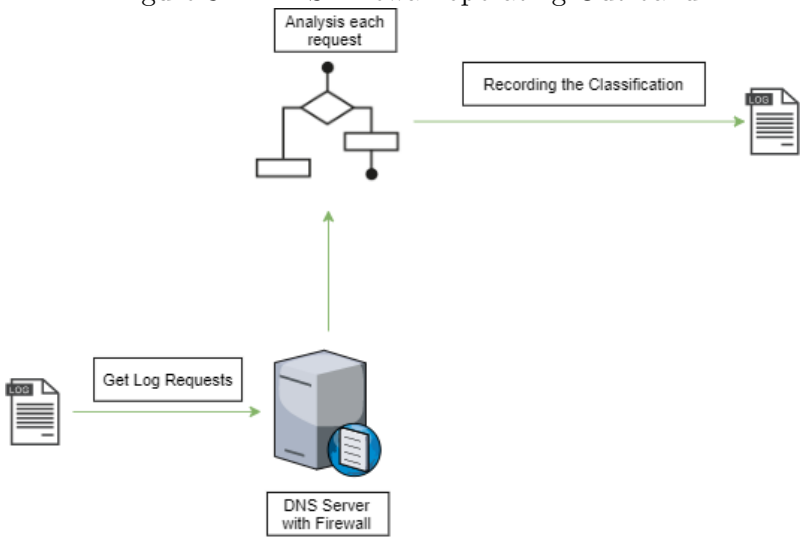
- In-band: As illustrated in Fig. 3.1, in the in-band way the DNS resolution requests are analyzed in real time. This allows to determines whether the requested domain is malicious or not and blocks or alerts as soon as possible to the fact. The overhead introduced by the DNS firewall is important and will be addressed in the project considering different DNS request workloads.

Figure 3.1: DNS Firewall operating in In-band



- Out-band: As illustrated in Fig. 3.2, in the out-band mode the DNS firewall will just analyze periodically the DNS log. This way it will only be able to provide a posterior analysis. Thus, it is useful to identify devices within the network that are involved in malicious communications. This operation mode does not overload the normal processing of DNS requests.

Figure 3.2: DNS Firewall operating Out-band

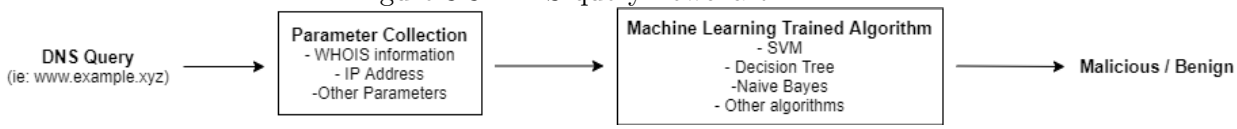


The analysis of the two modes of operation is very important as it allows to evaluate the effectiveness of the solution and to measure its online behavior.

The system will analyze each DNS query to gather information about the domain

name and use that information as input for posterior processing. In this step, parameters such as the WHOIS information, IP address, and subdomains will be collected. After the parameters collection, the data will be sent to a ML classification algorithm already trained. The classification response will be used to determine if the domain is malicious or not (considering the algorithm accuracy). This process is presented in Fig. 3.3.

Figure 3.3: DNS query flowchart



The entire process of the DNS dataset creation to be used to train ML algorithms and choosing the best ML algorithm from a historical set of DNS requests will be presented in the following chapters.

Chapter 4

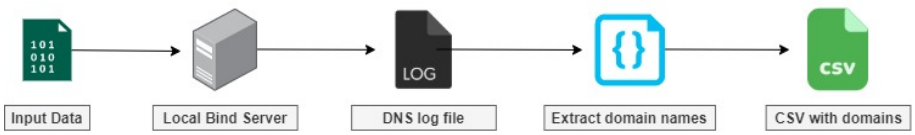
DNS Dataset

In this chapter the DNS dataset creation process is presented. The data collection for non-malicious domains was based on DNS logs and uses four types of records (A, AAAA, CNAME and MX). The domains were collected from Rapid7 Labs [51] that provides datasets of DNS requests from their Project Sonar. This data is open and provides a structured schema that allows a simpler extraction process. A total of 45000 domains were randomly selected from these lists.

The malicious domains were collected from the SANS Internet Storm Center (SANS) [55] public list. Are well-known suspicious domain list validated using common virus detectors publicly available. A total of 45000 domains were randomly selected from the list.

In a real scenario, the data will be obtained from the DNS server and not from structured public available files. To mimics the real scenario we replicated the DNS queries for each domain name collected (Fig. 4.1). To do so, we have configured a local DNS Bind server [20] allowing us to collect the DNS queries and responses. These responses were outputted to a file that was parsed giving rise to a Comma-separated Values (CSV) file containing the domains names. The CSV with the domains was then used to extract the dataset features.

Figure 4.1: DNS requests and extraction process.



A total of 90000 domain names constitute the entry point to the information gathering process using OSINT sources, geographic information, and analytic description. The information gathering was achieved using a Python application capable of extract domain names from log records, fill the information for each domain and save the dataset in CSV format. This file uses a comma as a delimiter and a header to label each column. The last column named "Class" is the dependent variable classifying each domain as malicious or non-malicious based on the collected data sources. Providing a classification for each row, this dataset is a valuable data source for supervised ML applications.

4.1 Creating the dataset

To create the dataset we defined thirty-four different features per DNS domain name. The features are presented in Table 4.1, containing the description, the data type and the default value. The decimal values are rounded to the first decimal place. The X denotes the type of data and N/A refers to the non-applicable field. The DNSRecordType feature was left in the dataset for filtering purposes, allowing the analyst to select data according to the DNS record type (A, AAAA, CNAME and MX). The enumerated data types are described in Table 4.2.

Table 4.1: Dataset features with description, data types and default value

Feature	Description	Data Type					Default Value
		Text	Boolean	Integer	Decimal	Enumerate	
Domain	Baseline DNS used to enrich data, e.g. derive features	X					N/A
DNSRecordType	DNS record type queried	X					N/A
MXDnsResponse	The response from a DNS request for the record type MX		X				False
TXTDnsResponse	The response from a DNS request for the record type TXT		X				False
HasSPFInfo	If the DNS response has Sender Policy Framework attribute		X				False
HasDkimInfo	If the DNS response has Domain Keys Identified Email attribute		X				False
HasDmarcInfo	If the DNS response has Domain-Based Message Authentication		X				False
IP	The IP for the domain	X					null
DomainInAlexaDB	If the domain is registered in the Alexa DB		X				False
CommonPorts	If the domain is available on common ports (80, 443, 21, 22, 23, 25, 53, 110, 143, 161, 445, 465, 587, 993, 995, 3306, 3389, 7547, 8080, 8888)		X				False
CountryCode	The country code associated with the IP of the domain	X					null
RegisteredCountryCode	The country code defined in the domain registration process (WHOIS)	X					null
CreationDate	The creation date of the domain (WHOIS)					X	0
LastUpdateDate	The last update date of the domain (WHOIS)					X	0
ASN	The Autonomous System Number for the domain			X			-1
HttpResponseCode	The HTTP/HTTPS response status code for the domain					X	0
RegisteredOrg	The organization name associated with the domain (WHOIS)	X					null
SubdomainNumber	The number of subdomains for the domain			X			0
Entropy	The Shannon Entropy of the domain name			X			0
EntropyOfSubDomains	The mean value of the entropy for the subdomains			X			0
StrangeCharacters	The number of characters different from [a-zA-Z] and considering the existence maximum of two numeric integer values			X			0
TLD	The Top Level Domain for the domain	X					null
IpReputation	The result of the blocklisted search for the IP		X				False
DomainReputation	The result of the blocklisted search for the domain		X				False
ConsoantRatio	The ratio of consonant characters in the domain				X		0
NumericRatio	The ratio of numeric characters in the domain ($numeric\_chars/len(domain)$)				X		0
SpecialCharRatio	The ratio of special characters in the domain				X		0
VowelRatio	The ratio of vowel characters in the domain				X		0
ConsoantSequence	The maximum number of consecutive consonants in the domain			X			0
VowelSequence	The maximum number of consecutive vowels in the domain			X			0
NumericSequence	The maximum number of consecutive numbers in the domain			X			0
SpecialCharSequence	The maximum number of consecutive special characters in the domain			X			0
DomainLength	The length of the domain			X			N/A
Class	The class of the domain (malicious = 0 and non-malicious = 1)			X			N/A

For the feature selection, it was taken into account the study of related works resulting in around 30% of the selected features. The features Autonomous System Number (ASN), IP address, all the sequence and ratio (numeric, vowel, consonant and special characters) and geographic information were based on DNS features from related articles such as [47], [83] and [61]. The remaining features are based on meetings with the thesis supervisors where unique features of the domains (malicious and non-malicious) were discussed and a more detailed description of each feature is presented in the next paragraphs.

The MXDnsResponse feature was considered interesting because a registered non-malicious domain for professional purposes, normally, contains a registry to point to the Simple Mail Transfer Protocol (SMTP) provider. In this context, the nonexistence of the

Mail Exchanger (MX) registry on the DNS for the associated domain raises an alert flag for a not structured organization which can be considered suspicious behavior and valuable input to the exploratory analysis step. For the TXTDnsResponse feature was applied the same logic.

The HasSPFInfo feature is based on the Sender Policy Framework (SPF) attribute, the HasDkimInfo is related to Domain Keys Identified Email (DKIM) attribute and HasDmarcInfo corresponds to the Domain-Based Message Authentication Message Conformance (DMARC) attribute. These parameters can be extracted from the response of a TXT query made to a DNS server. According to Google [50], the SPF "lets you specify the servers and domains that are allowed to send an email for your organization. When receiving mail servers get a message from your organization, they compare the sending server to your list of allowed servers. This lets receiving servers verify the message actually came from you. The DKIM adds an encrypted digital signature to every message sent from your organization. Receiving mail servers use a public key to read the signature, and verify the message actually came from you. DKIM also prevents message content from being changed when the message is sent between servers. The DMARC tells receiving servers what to do with messages from your organization when they don't pass either SPF or DKIM. DMARC also sends reports that tell you which messages pass or fail SPF and DKIM. These reports help you identify possible email attacks and other vulnerabilities". If a domain does not contain SPF, DKIM and DMARC registries can be considered vulnerable to spam, phishing or spoofing.

The IP address is a unique identifier for a device on the internet or a local network. It is a set of four octets (e.g, 192.152.78.150) and each octet ranges between 0 and 255. For the dataset it was taken into account the public IP address for the associated domain. To retrieve the IP address from a domain it was used a native Python library (socket). A detailed explanation of how the IP address was retrieved from a domain name can be found further in this chapter.

The DomainInAlexaDB feature is based on the Alexa Top 1 million sites [11]. The data source for the Alexa top sites is the Alexa Traffic Rank [10] which provides the popularity trend, calculated using the page views and daily visitors over a three month period. For this dataset, if a domain has a positive match in the Alexa top sites list it is set to true

otherwise is set to false.

To choose common ports for the CommonPorts feature, a search for well-known network ports was performed. The ports numbers are used to distinguish between distinct services that are integrated with transport protocols such as the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

The Internet Assigned Numbers Authority (IANA) [37] provides "the global coordination of the DNS Root, IP addressing, and other Internet protocol resources". On their Service Name and Transport Protocol Port Number Registry [60] there are all the reserved and non reserved in use port numbers. According to the Red Hat Security Guide [24] and using the IANA registries as a data source, is possible to check numerous well-known ports. The network ports used in this feature and their common use are described below:

- 80 / 443 - Used for the Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS) protocols respectively;
- 21 - Commonly used for File Transfer Protocol (FTP) port;
- 22 - Used on the Secure Shell (SSH) service;
- 23 - The Telnet service;
- 25 - Used on the SMTP;
- 53 - Normally used for the DNS service;
- 110 - Used on Post Office Protocol version 3 (POP3);
- 143 - Used on Internet Message Access Protocol (IMAP);
- 161 - Used on Simple Network Management Protocol (SNMP);
- 445 - Used on Server Message Block (SMB) over TCP/IP;
- 465 - Used on Simple Mail Transfer Protocol over Secure Sockets Layer (SMTPS);
- 587 - Used on Mail Message Submission Agent (MSA);
- 993 - Used on Internet Message Access Protocol over Secure Sockets Layer (IMAPS);

- 995 - Used on Post Office Protocol version 3 over Secure Sockets Layer (POP3S);
- 3306 - Commonly used for the MySQL database service;
- 3389 - Commonly used for Windows Remote Desktop connections;
- 7547 - Used on CPE WAN Management Protocol (CWMP);
- 8080 - Commonly used for world wide web caching;
- 8888 - Normally used as an alternative to HTTP port.

The CountryCode, as the name suggests, is the country code (e.g. US, NL, PT) for a given IP address. To collect the geographic information from the IP address was used the geoip2 [45] framework. This feature allows the dataset to have a geographic distribution based on the country code.

The RegisteredCountryCode may differ from the CountryCode. It is based on the WHOIS information which provides valuable information from a domain name. Similar to CountryCode, this feature allows the dataset to have another geographic distribution and also to correlate differences between these geographic features.

The CreationDate and LastUpdateDate are also values gathered from the WHOIS database. Both features use enumeration to simplify all possible date ranges and are described in Table 4.2. A malicious domain name typically has a recently created or updated date registry. On the other side, a non-malicious domain tends to have an older creation date registry and less frequent updates. This information and the correlation between malicious and non-malicious are further described in the exploratory analysis chapter.

According to Cloudflare [79], the ASNs "are the big networks that make up the Internet. More specifically, an autonomous system (AS) is a large network or group of networks that have a unified routing policy. Every computer or device that connects to the Internet is connected to an AS.". The collection of the ASN for each domain can be used to investigate the concentration of malicious and non-malicious domains over an autonomous system. For instance, if the malicious domain names are related to a specific ASN, this feature will have a higher importance on the feature selection phase.

The HttpResponseCode makes use of enumeration to fit in the range of all possible HTTP response codes. The enumeration used is described in Table 4.2. Depending on the DNS record type, the HTTP/HTTPS response to a domain may indicate suspicious behavior. For example, a domain where a DNS server request was made with a record type of "A" will likely have an associated HTTP/HTTPS service.

The RegisteredOrg corresponds to the associated organization for a given domain name from the WHOIS database. The top most common registered organizations can be related to malicious or non-malicious domains which can provide valuable insights and it will be explored further in the exploratory analysis.

For the SubdomainNumber feature, a third-party framework named Sublist3r [70] was used. According to the documentation, the "Sublist3r is a python tool designed to enumerate subdomains of websites using OSINT. Sublist3r enumerates subdomains using many search engines such as Google, Yahoo, Bing, Baidu and Ask. Sublist3r also enumerates subdomains using Netcraft, Virustotal, ThreatCrowd, DNSdumpster and ReverseDNS". For some search engines, there is a limit of requests per day, hour, and in some cases per minute. To avoid these limits, only the following search engines were used: PassiveDNS and Bing. Usually, a professional organization has multiple subdomains used in a distinct variety of services such as email (e.g. email.domain) and Virtual Private Network (VPN) services (e.g. vpn.domain). The correlation between the subdomain number for malicious and non-malicious domains is described further in the exploratory analysis.

The Entropy feature allows computing the randomness of a domain name. It was used the Shannon Entropy method [74] which allows a metric for the randomness of characters in a domain name. For instance, simulating a well-known domain and a DGA domain it was calculated the entropy for the following domain names: "google.pt" and "asdhbajhdsbj.com". The domain "google.pt" has an entropy value of 2.73 and the domain "asdhbajhdsbj.com" has an entropy value of 3.25 as expected. The entropy values for a domain can be a valuable input to a future ML application. Domains based on the DGA are typically random, and so the entropy constitutes an interesting feature. Similarly to the Entropy feature, the EntropyOfSubDomains represents the mean value of the entropy of the subdomains.

Generally, the malicious domains and principally the ones generated by DGA con-

tain much more unusual characters. For the StrangeCharacters feature was calculated the number of characters different from [a-zA-Z] and considering a maximum of two numeric integer values. For example, the domain "my123domain.com" contains two strange characters. Firstly, we removed the [a-zA-Z] characters leaving the domain as "123.". Secondly, we remove two numeric characters and the result is "3.".

The TLD feature provides the TLD from a domain name. According to Cloudflare [78], "In the DNS hierarchy, a top-level domain (TLD) represents the first stop after the root zone. In simpler terms, a TLD is everything that follows the final dot of a domain name. TLDs play an important role in the DNS lookup process. For all uncached requests, when a user enters a domain name like 'google.com' into their browser, the DNS resolvers start the search by communicating with the TLD server. In this case, the TLD is '.com', so the resolver will contact the TLD DNS server, which will then provide the resolver with the IP address of Google's origin server". There are also five distinct types of TLD:

- Generic: The most common TLD used such as ".com" and ".net";
- Country code: Based on the geographic information such as ".uk" and ".pt";
- Sponsored: TLD with an associated sponsor to represent a community (e.g. ".edu" used by educational institutions recognized by the US Department of Education);
- Infrastructural: So far this category only contains a registry ".arpa", the first TLD created. This is used by the US Defense Advanced Research Projects Agency;
- Reserved: The common example for a reserved TLD is the ".localhost" always reserved for the local machine.

To extract the TLD from a domain name it was used the TLD python package [17].

The IPReputation and DomainReputation features are based on the match response to a given IP address or domain provided by the pydnsbl framework [38]. According to the documentation, this framework allow to check if a IP address or domain is listed in anti-spam DNS blacklists provided by the Spamhaus project [72].

The ratio and sequence-based features focus on the analytical domain characteristics. The ratio values are calculated by the mean of all targeted characters on the domain

length. For example, the ConsoantRatio represents the mean value of consonants characters in the domain (e.g.: "mydomain.com" as a consonants ratio of 0.58). The sequence values are calculated by the bigger sequence of the targeted characters. For instance, the VowelSequence for the domain "mydomain.com" is two (ai).

Further in this chapter, there is a critical analysis of all the features based on exploratory analysis.

The enumerations data types identified in Table 4.1 are presented in Table 4.2. The enumerations have been created to prepare the data to be better supported by the ML algorithms. The values adopted resulted from the analysis of studies, like [40], [18] and [82] that focus on domains names gathering and DNS features to improve the malicious detection based on ML applications.

Table 4.2: Values description for enumeration features where X denotes all possible values

Feature	**Values description**
CreationDate LastUpdateDate	Without data = 0 Until one month = 1 Until six months = 2 Until one year = 3 After one year = 4
HttpResponseCode	Without data = 0 1XX response = 1 2XX response = 2 3XX response = 3 4XX response = 4 5XX response = 5

In order to get all the features, we created several programs using Python, developed modules for better organization and reuse and used available frameworks. For instance, to get the IP address for each domain it was created a function to deal with it. The Code Snippet 4.1 is part of the information gathering (data enrichment) process and it makes use of the socket native python library to retrieve the IP from a domain name. Some domains are not configured with redirect rules to take care of a request without the

"www" subdomain prefix. If the DNS query fails on the first try, we concatenated with "www" to maximize the results.

Code Snippet 4.1: Function to get the IP for a given domain

```
import logging
import socket

def getIp(domain):
    try:
        socket.setdefaulttimeout(0.05)
        return socket.gethostbyname(domain)
    except:
        try:
            socket.setdefaulttimeout(0.05)
            return socket.gethostbyname("www." + domain)
        except Exception as e:
            logging.error("Exception in getIp: " + str(e))
            pass
        pass
    return 'null'
```

The full Python code used to create the dataset is available in [44]. The structure of the code is based on the main Python file that calls functions in different modules and makes use of utilities and data stored in different folders. The structure is the following:

- main_create_datasets.py - The main file to create the dataset and order all the steps starting in the data collection to information gathering;
- data/ - Inside the folder there are two sub folders (input and output). The logs collected must be inside the input folder categorized by the DNS record type and by the class. The output sub folder is the path where the final dataset will be saved;
- lib/ - Contains the functions and modules to support the information gathering process;

- utils/ - Contains utility functions in Python Scripts, such as the constants to be used in run-time. It contains a database sub folder with the AlexaDB [11] and GeoIp [45] databases inside. The data tools sub folder contains the functions to the collection of the domains from the logs files and for the information gathering.

The third-party libraries/data used to collect information from a domain are:

- AlexaDB [11] - Alexa top 1 million sites. Used to evaluate if a domain is or not in the list which allowed us to fill the "DomainInAlexaDB" feature;
- geoip2 [45] - GeoIP Database. This framework allows us to gather the geographic information (e.g. country code) for a given IP address;
- pydnsbl [38] - Anti-spam blacklists domain or IP checker. For a given domain (DomainReputation) or IP address (IpReputation) this framework returns a positive result if a match is found in the blacklists providing the reputation feedback;
- sublist3r [70] - Python package to enumerate subdomains of a given domain using OSINT (SubdomainNumber). The engines used were PassiveDNS and Bing;
- tld [17] - Python package to extract the TLD for a given domain;
- IPWhois [31] - Python package to retrieve information from WHOIS. Probably one of the most used frameworks on OSINT tools that return valuable information for a given domain such as the creation date, last update date, the registered organization and the registered country.

The result of the information gathering process is stored in a CSV file. A small excerpt of the result is illustrated in Fig. 4.2. The values of the IP and domain columns, which are considered personal data under the General Data Protection Regulation (GDPR) rules, were anonymized using the Label Encoder from SkLearn framework [57] with a value between 0 and $n\_classes - 1$.

Figure 4.2: Dataset result snippet.

```
44200,MX,True,False,False,False,False,11465,False,False,US,null,4,4,62729,0,ASO-239164-20,1,2,0,0,com,False,False,0.4,0.2,0.0,0.2,2,1,1,0,9,1
44201,MX,True,False,False,False,False,12851,False,False,DK,DK,0,4,51468,2,ONECOM-INFRA,0,2,0,0,info,False,False,0.4,0.2,0.0,0.3,2,1,1,0,10,1
44202,MX,True,False,False,False,False,6869,False,True,US,US,0,4,58182,2,wix_com_inc,0,2,0,0,tokyo,False,False,0.5,0.2,0.0,0.3,2,1,1,0,11,1
44203,MX,True,False,False,False,False,14755,False,False,US,null,4,4,31815,0,MEDIATEMPLE-105,0,3,0,0,co.za,False,False,0.4,0.1,0.0,0.4,2,1,1,0,17,1
48179,A,True,True,True,False,False,10108,False,False,US,null,4,4,62729,2,SOL-207-210-200-0-22,0,3,0,0,com,False,False,0.6,0.0,0.0,0.3,4,1,0,0,13,0
84056,A,False,False,False,False,False,11278,False,False,US,null,4,4,396190,2,NETBLK-NOBIS-TECHNOLOGY-GROUP-18,0,3,0,0,com,False,False,0.6,0.0,0.0,0.4,5,3,0,0,14,0
63572,A,False,False,False,False,False,16984,False,False,null,null,0,0,-1,0,null,0,3,0,1,dyndns-ip.com,False,False,0.6,0.1,0.0,0.2,6,1,2,1,24,0
76819,A,False,False,False,False,False,16984,False,False,null,null,0,0,-1,0,null,0,2,0,0,com,False,False,0.6,0.0,0.0,0.3,2,1,0,0,11,0
66146,A,False,False,False,False,False,16984,False,False,null,null,0,0,-1,0,null,0,3,0,1,com,False,False,0.5,0.0,0.1,0.4,2,2,0,1,15,0
```

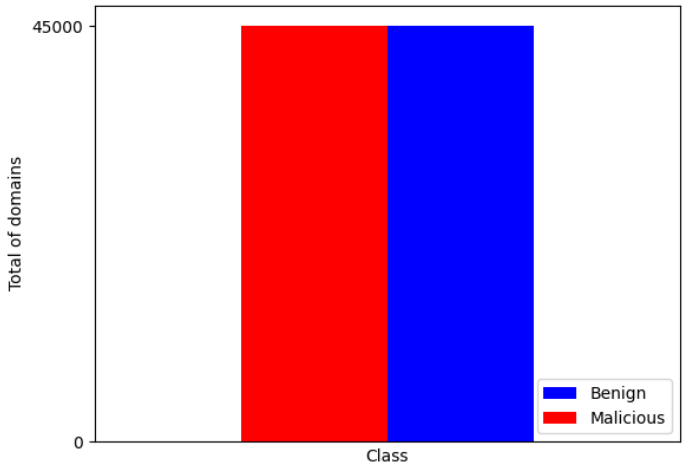
The full dataset is publicly available at Mendeley Data Repository [44]. It can be used by other researchers to conduct experiences.

4.2 Exploratory analysis

In this section is presented an exploratory analysis for each feature in the dataset and their correlation to the class (malicious or benign).

The dataset is targeted for Supervised ML Classification. It is a binary classification predictive modeling since according to the features presented in Table 4.1, the class is zero (0) for malicious domains and one (1) for non-malicious domains. The dataset is also balanced and distributed as illustrated in Fig. 4.3. The ML classification should consider the default dataset characteristics or adapt the dataset for other types of data analysis.

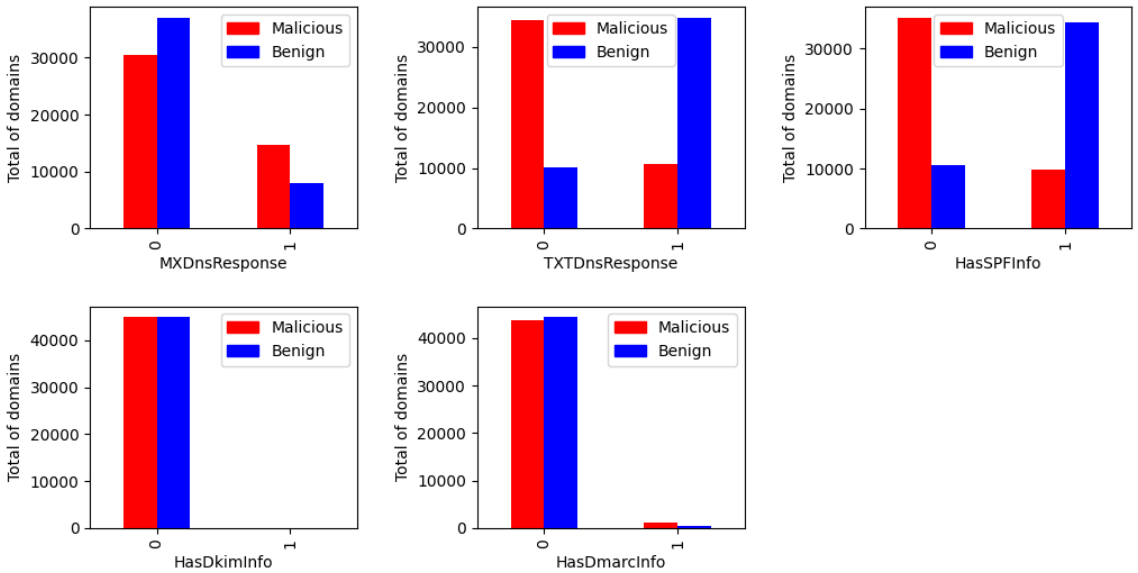
Figure 4.3: Class label distribution graph



The total number of complete entries without “null” values in the dataset is 11547 (12,83%). There are 78453 (87,17%) rows where at least one of the features is “null”. The “null” values should be considered in the data preparation phase allowing the data analyst to choose the best approach to handle the “null” values.

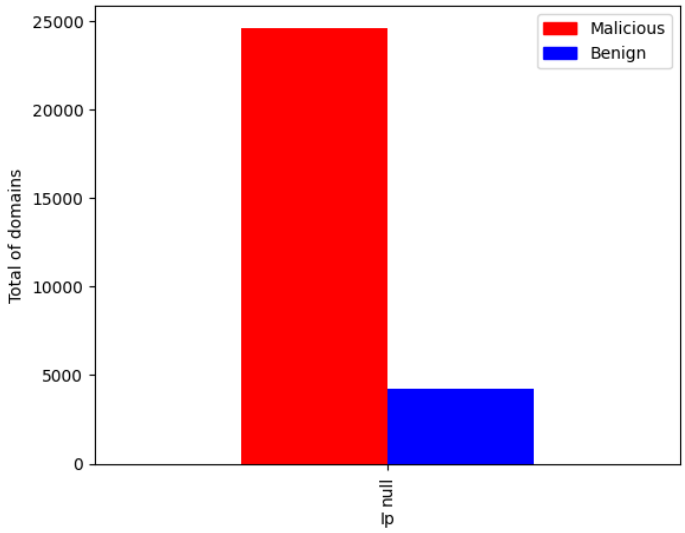
The features related to the DNS response are illustrated in the Fig. 4.4. The figure shows the type of DNS record type requested per class.

Figure 4.4: DNS response by class



As expected the feature IP has a large number of different values. In Fig. 4.5 is presented a correlation between "null" IP address values per class label. The figure shows that there is a high number of malicious domains that do not have an associated IP. This may indicate that the IP has not yet been assigned to the domain or that the domain has already been used for cyberattacks and its DNS mapping has been removed. This analysis is useful for posterior feature selection.

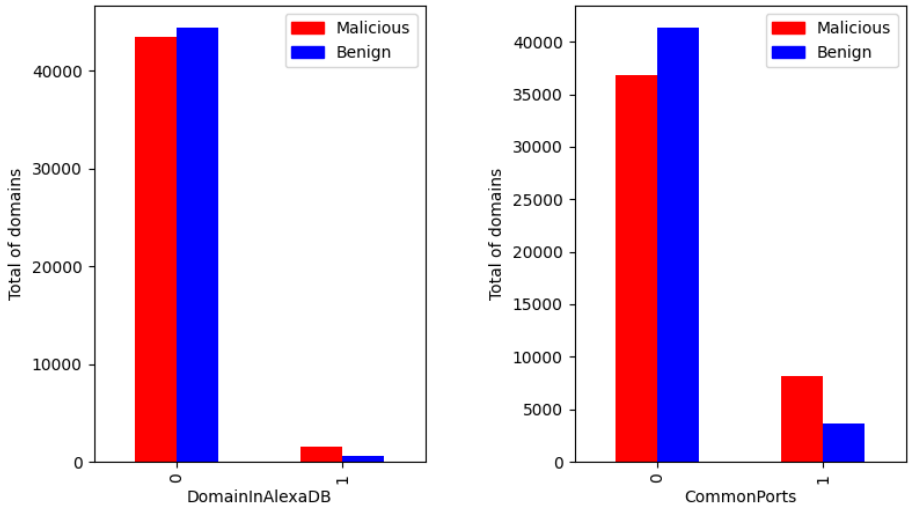
Figure 4.5: IP null values per class



A simple plot between the DomainInAlexaDB and CommonPorts, illustrated in Fig. 4.6, reveals that the distribution of features according to the class is uniform and similar.

The first plot shows that there are many domains in the dataset that are not included in the Alexa DB [11], regardless of class. The second shows that for a significant part of the domains there are no common active ports (no common services are being provided).

Figure 4.6: DomainInAlexaDB and CommonPorts by class



The CountryCode and RegisteredCountry features reveal geographic information about the domain or the IP associated with the domain. These features are illustrated in Fig. 4.7. With regard to the CountryCode, it appears that most IP are active in the USA and that they are mostly malicious. This has to do with the data source implying that malicious domains are largely associated with servers hosted in the USA (they may be sinkhole DNS servers). The RegisteredCountry focuses on the domain information (collected using the WHOIS database). It shows a greater geographical dispersion and distributed among the classes under analysis. This analysis does not consider the null values and we used only the top 20 country codes for presentation purposes.

Figure 4.7: Geographic information and null count by class

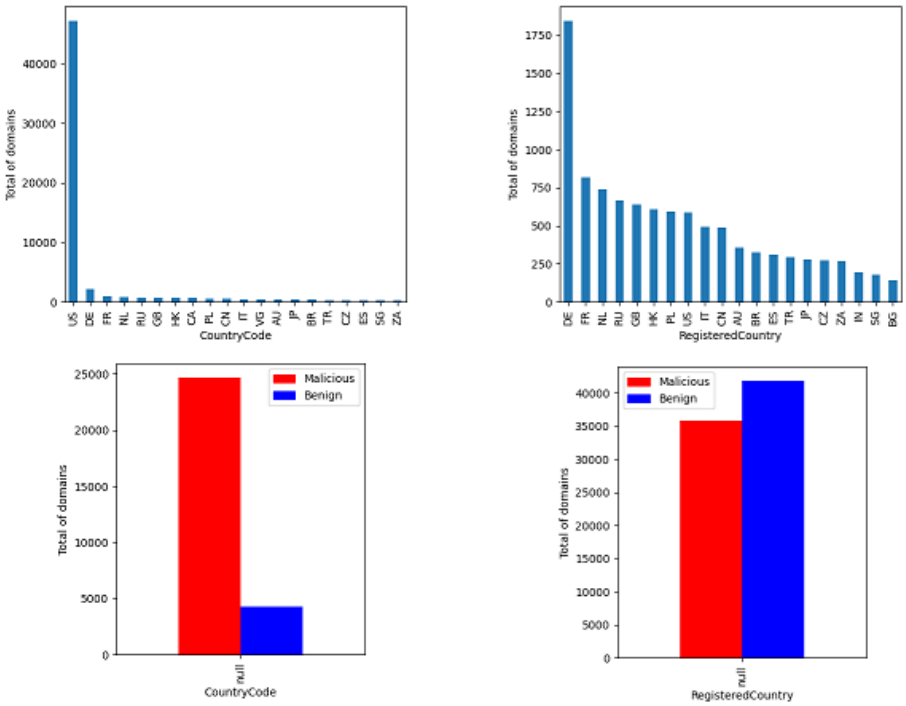


Fig. 4.8 and 4.9 presented a different view for the CountryCode and RegisteredCountry labels respectively.

Figure 4.8: Country Code geographic distribution in a world map

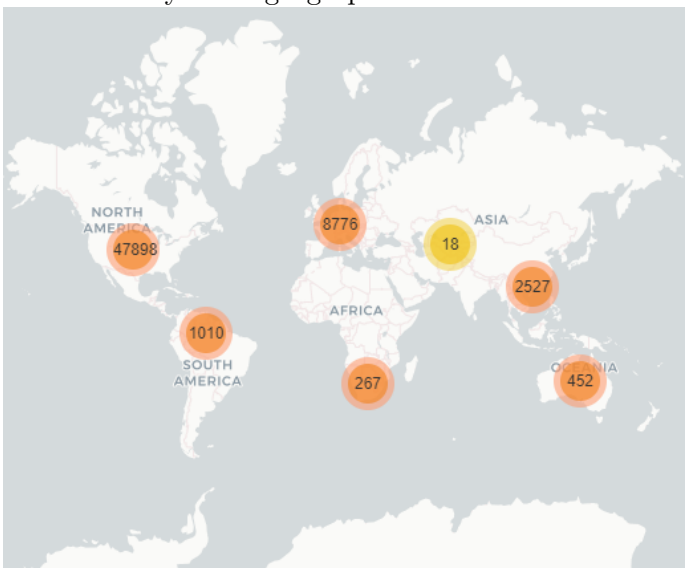
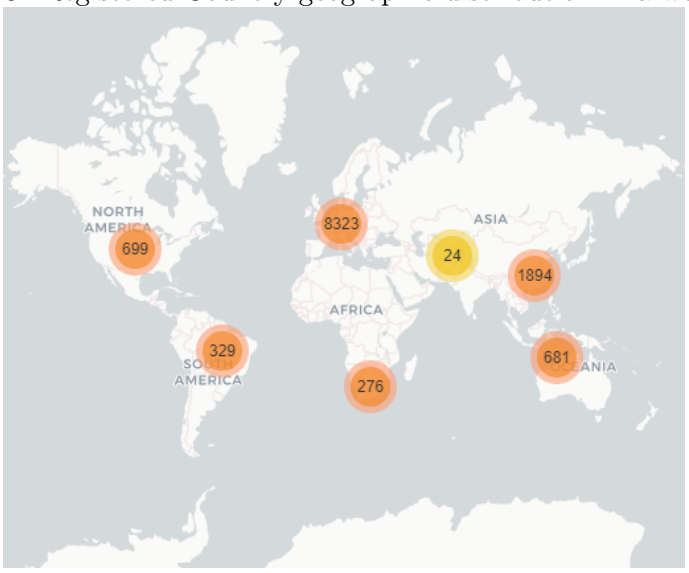
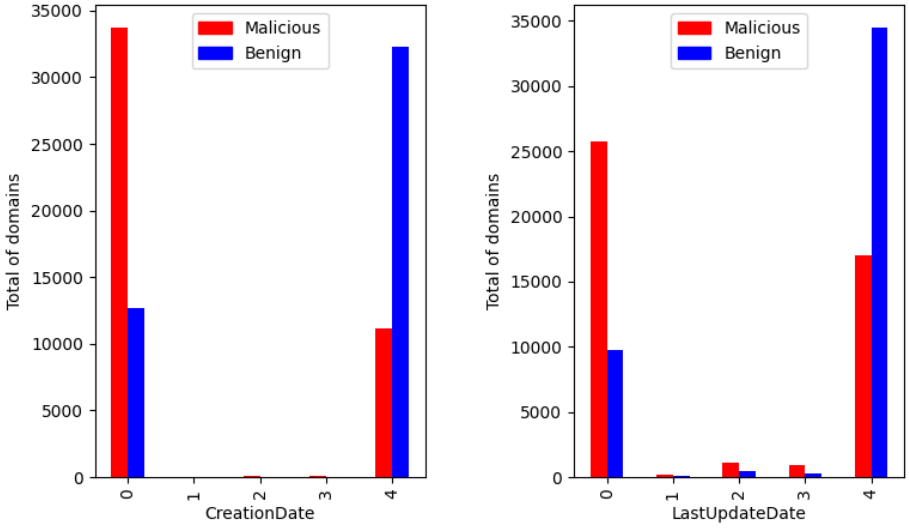


Figure 4.9: Registered Country geographic distribution in a world map



The domain registration/creation date and last update date uses an enumeration to reduce the range of dates. In Fig. 4.10 these features are illustrated grouped by the class label. This analysis is important to understand if malicious domains are created and changed more frequently than non-malicious domains (older and more stable, particularly for well-known domains).

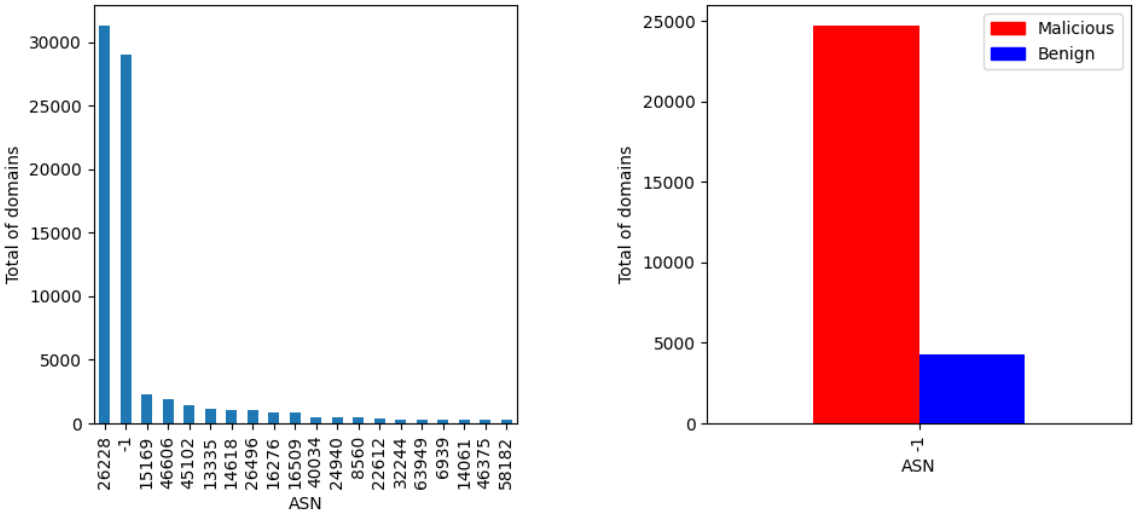
Figure 4.10: Domain creation and last update date per class



The ASN distribution is illustrated in Fig. 4.11. The minus one value means that there is no ASN information for the domain. From the figures it is also possible to observe the non found (negative) values regarding the class label. It is important to refer that

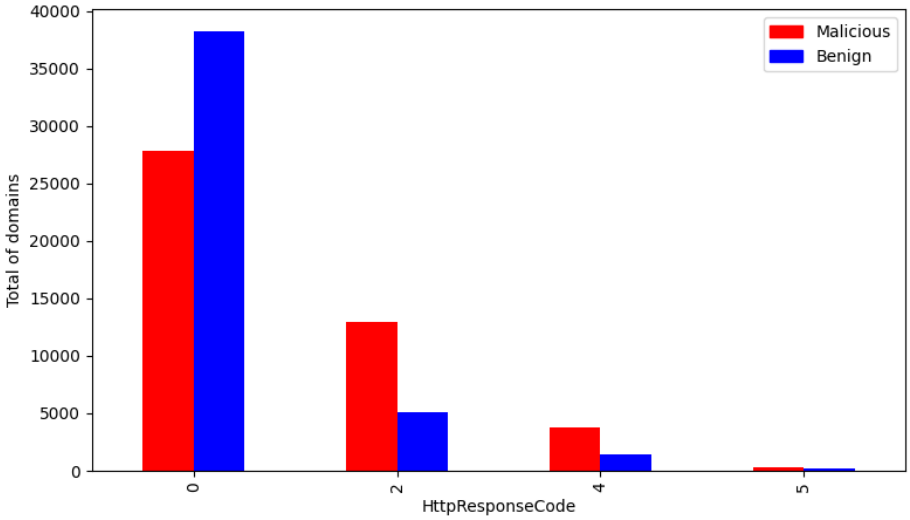
the ASN information is related with the existence of an IP address associated with the domain, so it is not strange at all that the output of this analysis is very similar to IP analysis previously presented. Once more it is possible to verify the influence of ASN 26228. This is not due to outliers but it seems to result from the data origin.

Figure 4.11: ASN data distribution per class



The HTTP / HTTPS response code enumeration is described in Table 4.2. In Fig. 4.12 it is possible to see the distribution of the response code by the class label.

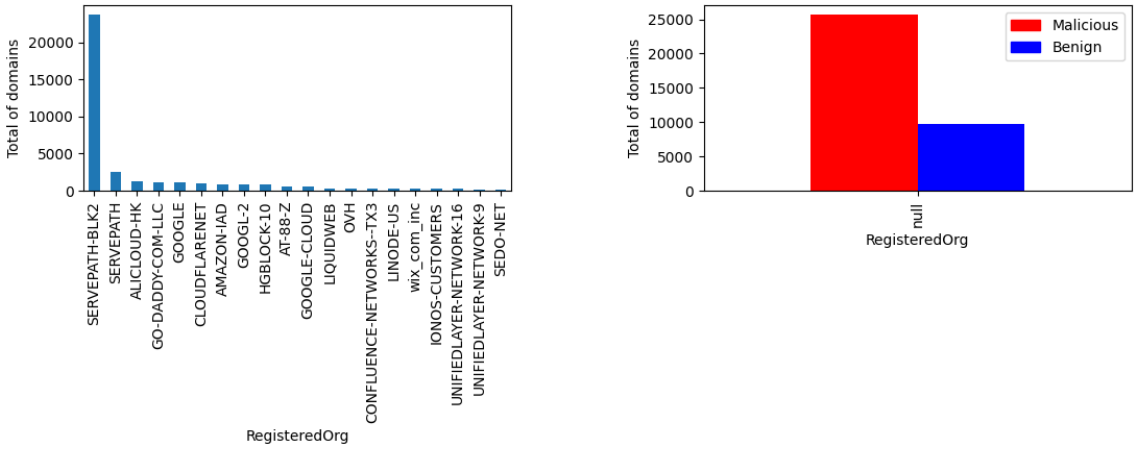
Figure 4.12: HTTP / HTTPS response by class



The registered organization feature results from querying the WHOIS service. In Fig. 4.13 are illustrated the top 20 organizations in the dataset. Typically, a non-malicious organization does not hide this information when registering the domain. On the other

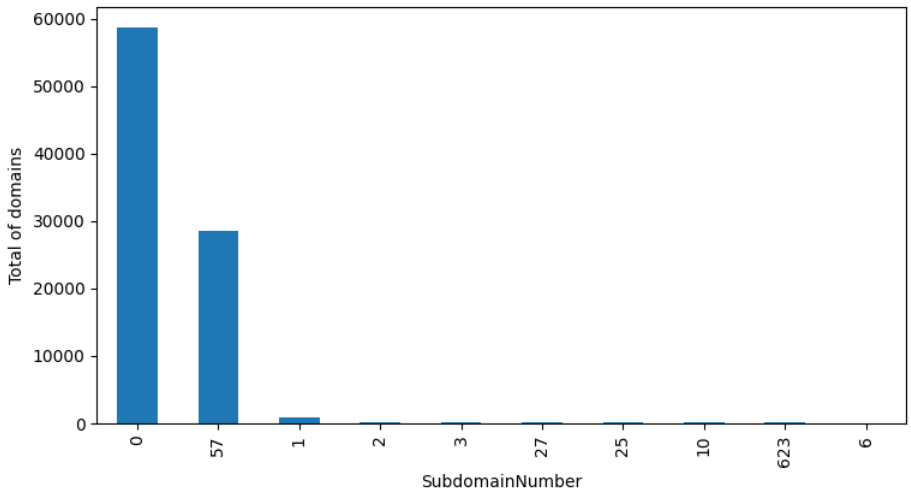
hand, malicious actors tend not to reveal this information or to tamper with it. In the figure is presented the relationship between the "null" values (e.g. when the information is not available) per class.

Figure 4.13: Registered Organization label distribution



The subdomain feature allows checking if a given domain has subdomains registered. The rationale behind this parameter is to follow. It is normal for a real organization to have multiple subdomains associated with the domain. On the contrary, a malicious domain will not normally have many associated subdomains. The figure shows the top 10 most frequent subdomain count and the number of domains that fit in each count. From the illustration we observed that more than 55 thousand domains do not have subdomains associated. From the dataset it is also possible to improve the analysis. For example, it allows to do a cross-table between the number of subdomains and their class.

Figure 4.14: Subdomains label distribution



Other features presented in the dataset are the domain entropy and the mean entropy value obtained from the entropy of each subdomain. The result was rounded to integer values and the distribution by the class label illustrated in the Fig. 4.15.

Figure 4.15: Entropy of domain and mean entropy of subdomains by class

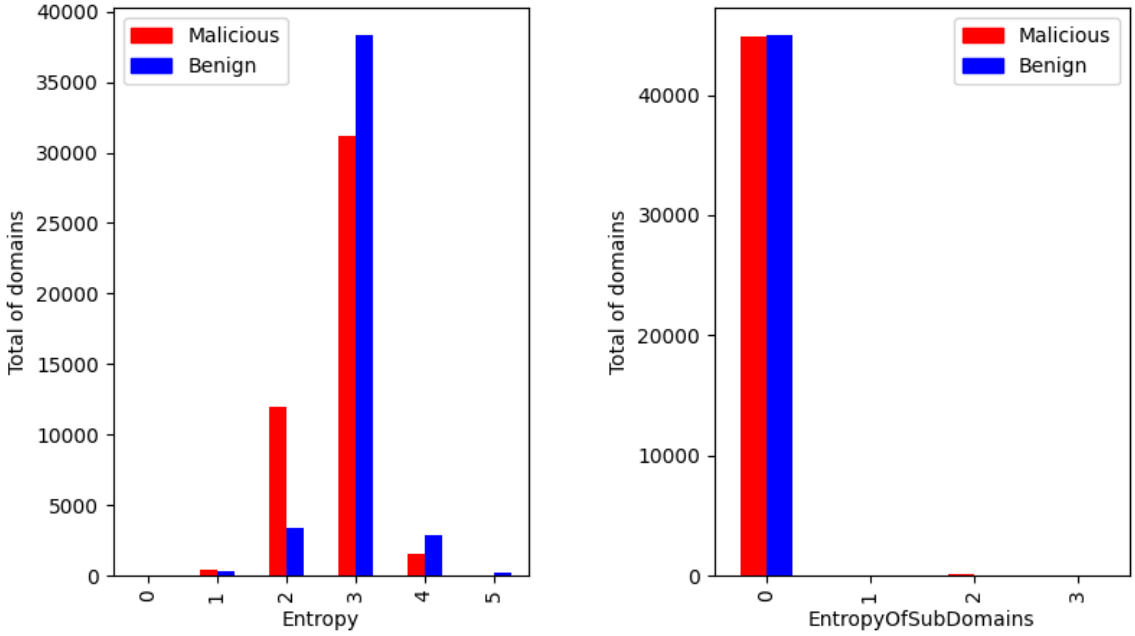
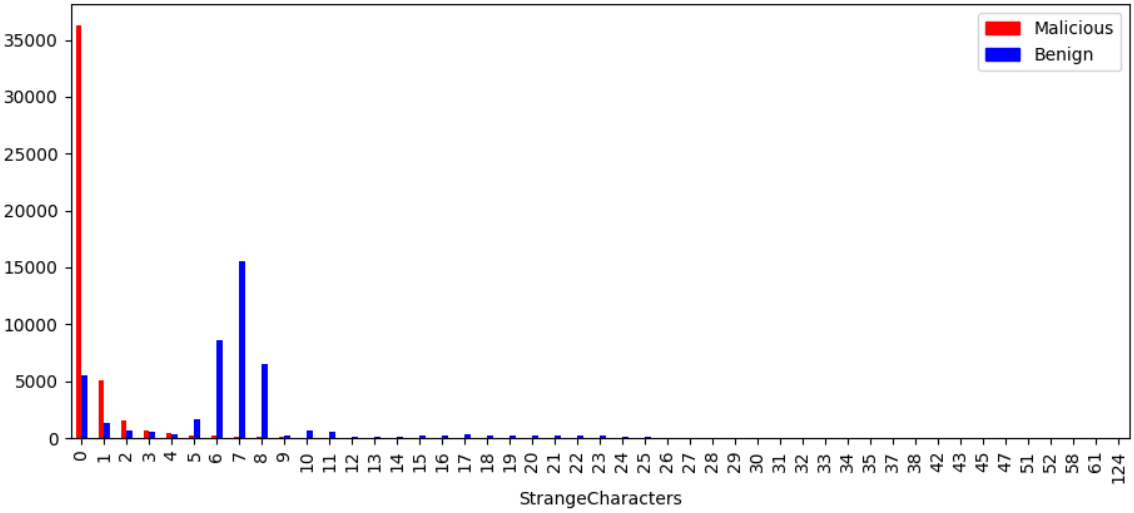


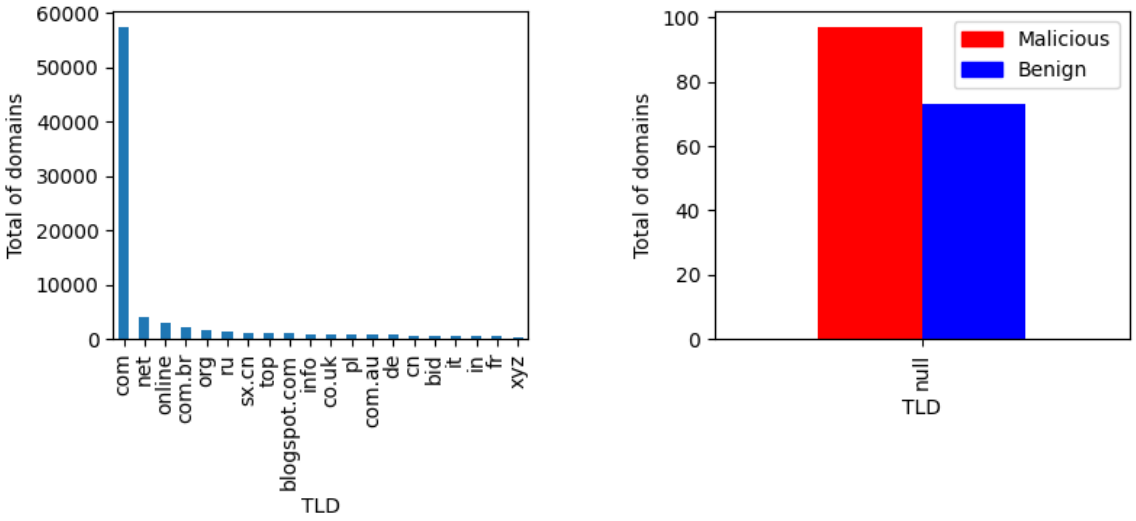
Fig. 4.16 illustrates the strange characters feature per class. The domain name is part of an organization identity, so it is expected that the name will be chosen in order to be easily used and memorized. The existence of strange characters contradicts these logic and can serve as an indicator of the existence of domains for malicious purposes.

Figure 4.16: Strange Characters label distribution by class



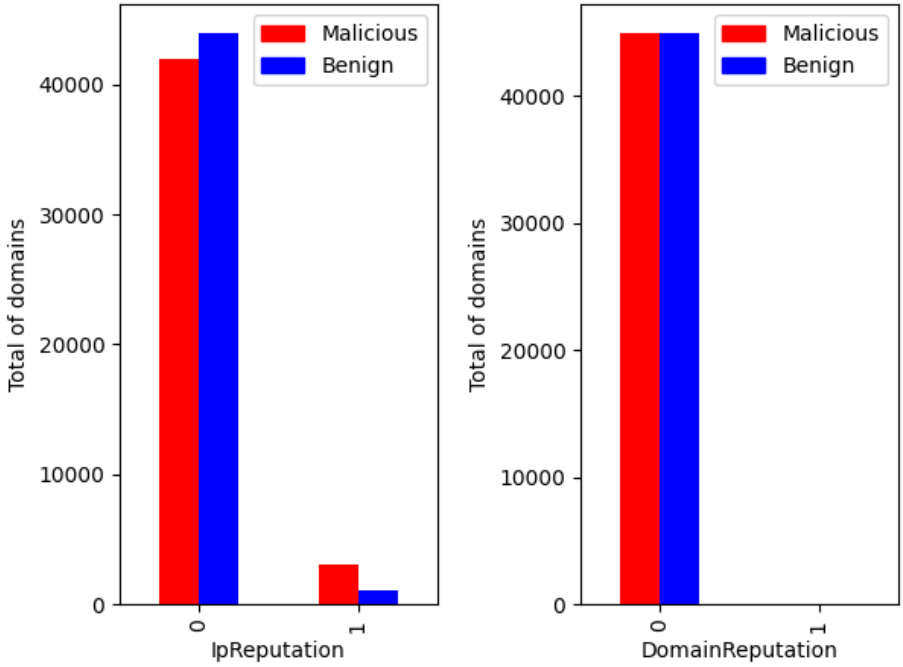
The TLD feature per class is illustrated in Fig.4.17. For presentation purposes only the top 20 TLD are presented. The null values per class are illustrated in the right-side of the figure.

Figure 4.17: TLD label distribution and null values per class



The IP and domain reputation feature per class is illustrated in the Fig. 4.18. Both parameters result from the classification made by third parties and are widely used in the area of cybersecurity to identify possible IP and malicious domains. Their existence in the dataset allow to ascertain the weight they will have for the classification process.

Figure 4.18: IP and domain reputation labels distribution per class



The distribution of the ratio of vowels, consonants, numeric and special characters in the domain is illustrated in Fig. 4.19. The same representation was made for the sequences illustrated in Fig. 4.20. The combined analysis between these parameters is interesting, as it is expected that a non-malicious domain name will be created in order to be easily memorized and used.

Figure 4.19: Ratios distribution by class

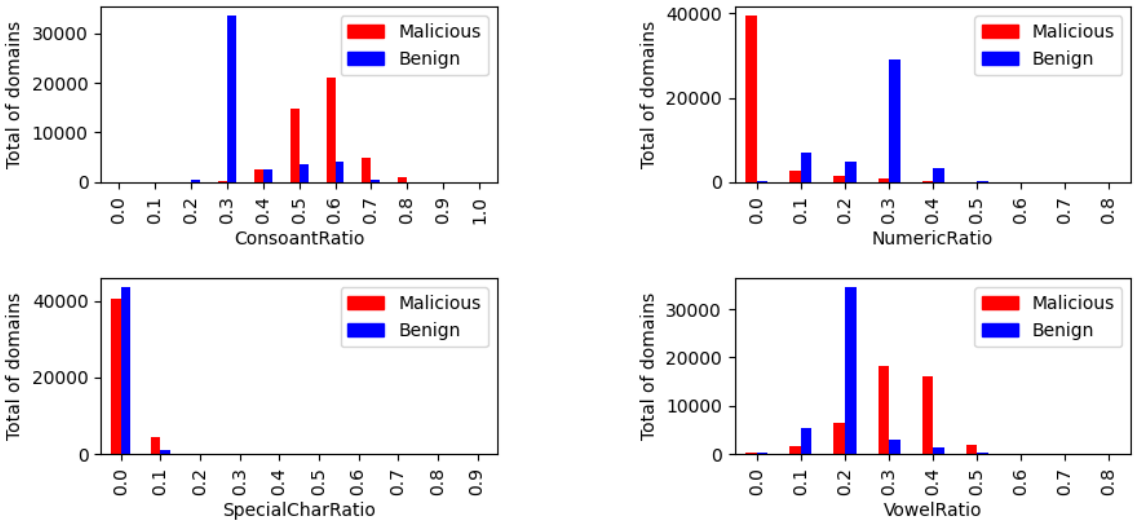
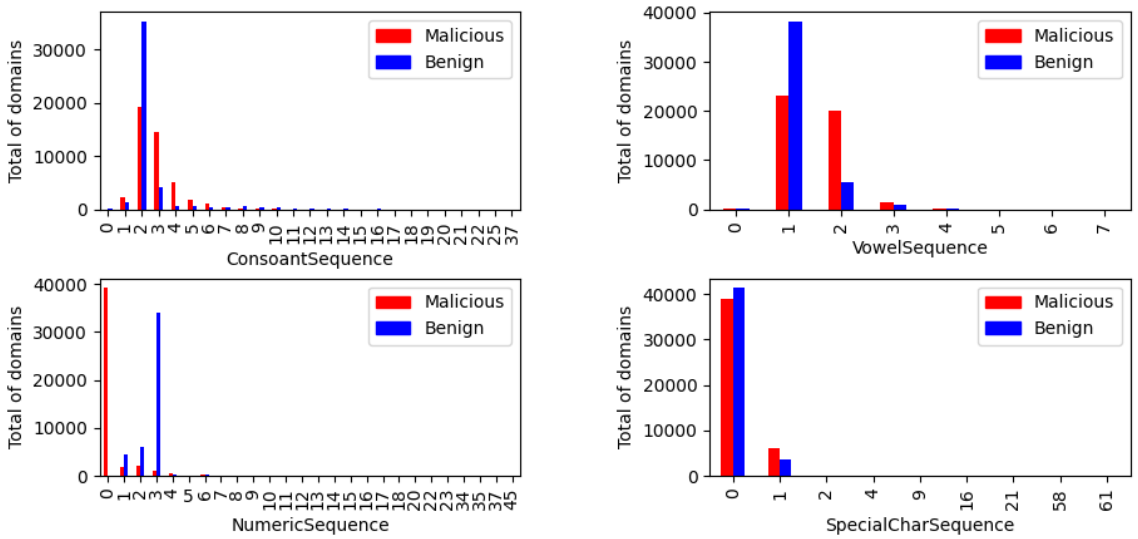
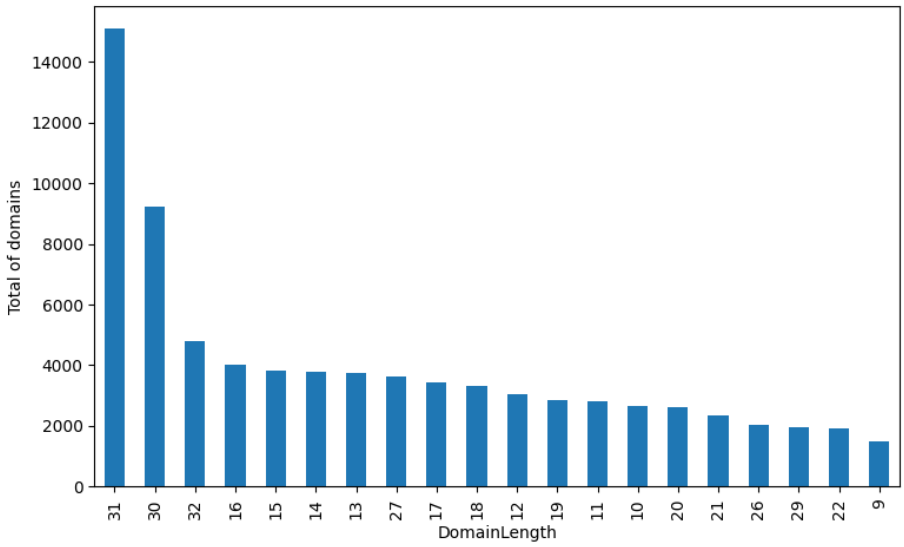


Figure 4.20: Sequence distribution by class



The domain length per class is illustrated in Fig. 4.21. Once more, for presentation purposes, a frequency chart with the 20 most common domain sizes is presented.

Figure 4.21: Domain length label distribution



The results of the exploratory analysis lead to considering several features that stood out. The HasSPFInfo feature that looks for SPF information in a domain is usually associated with benign domains, which was confirmed in this analysis. The same happened with the IP feature's null values, taking into account that malicious domains tend to have a short lifespan. Consequently, the CountryCode feature had a much higher number of null values for malicious domains. The creation and last update dates obtained by the WHOIS database show that malicious domains tend not to have this information associated, unlike non-malicious domains. Similarly, the RegisteredOrg feature has a higher number of null values for malicious domains. Ratio and sequence-related features such as ConsoantRatio, NumericRatio, and VowelSequence show a large discrepancy of values between classes (malicious and benign), indicating that they will be good features for later selection to apply ML algorithms.

In the next chapter, an analysis of the data will be presented. The dataset presented in this chapter will be used as data source for the analysis. Before the analysis it will be prepared and normalized with methods such as label encoding. Afterward, different ML algorithms will be applied and the results and discussion will be presented.

Chapter 5

Data analysis

In this chapter we present the data preparation and data analysis phases. The results achieved by different ML algorithms is also presented.

5.1 Data preparation

In this section, the preparation of data will be presented. According to [21] there are several tasks for preparing the data such as data cleaning and data transformation.

For the data cleaning step all null values for text type columns were set as "null" (string data type). For the remaining column types the null values were replaced by the default value described in the Table 4.1.

In the data transformation step, we ignored the Domain and DNSRecordType columns, as mentioned earlier, the Domain column was anonymized according to the GDPR rules, and the DNSRecordType was left in the dataset for filtering purposes only. Therefore, these features have no value for later use in ML algorithms. For the text and boolean type columns we use the Label Encoder from SkLearn framework [57] with a value between 0 and $n\_classes - 1$ as shown in Code Snippet 5.1. The boolean type was transformed to integers (0 and 1).

Code Snippet 5.1: Function to encode text and boolean columns

```
from sklearn.preprocessing import LabelEncoder

def label_encoding(df):
```

```
    label_encoder = LabelEncoder()
    columns = df.columns.tolist()
    for col in columns:
        col_data_type = df.dtypes.__getitem__(col)
        if col_data_type.char == "O" or col_data_type.name == "bool":
            integer_encoded = label_encoder.fit_transform(df[col])
            df[col] = integer_encoded
    return df
```

For all integers (excluding the Class label) it was used the min-max normalization, a well-known approach used in past scientific experiments like [69] and [56], to export all the values to a float data type and a range between 0.0 and 1.0 as shown in Code Snippet 5.2. According to [9] the min-max method performs a linear transformation on the original data fitting numeric values into a desired scale (in this case to a range between 0.0 and 1.0).

Code Snippet 5.2: Function to normalize integers columns

```
def get_min_max(df):
    for col in df.columns:
        if col != 'Class' and df[col].dtype.name in ["int64", "int32"]:
            df[col] = (df[col] - df[col].min()) / \
                      (df[col].max() - df[col].min())
    return df
```

According to Google [46] there are four normalization techniques:

- Scaling to a range: converting numeric values from their natural range (e.g. 0 to 500) to a standard range (e.g. 0.0 to 1.0) and min-max normalization operates in the same way. It should be used on a feature with more or less homogeneous distribution over a fixed range;
- Clipping: Basically this technique allows to establish a fixed value for values above (or below) a certain value. For instance, in a scale from 0 to 100 it can be set a rule

that ensures the values above 50 will remain 50. It should be used when the feature contains some extreme outliers;

- Log scaling: Used on a widely range of values to fit into a smaller scale. For example, in a range between 0 - 100.000 it can be performed this technique to reduce the scale to 0 - 1000. As cited in [46], "log scaling changes the distribution, helping to improve linear model performance". It should be used when the feature conforms to the power law;

- Z-score: This technique allows to variate a scaling to ensure a feature distribution that has a mean value of zero and a standard deviation definition of one. It should be used when the feature distribution does not contain extreme outliers.

Although there are several techniques for data normalization, the one that fits best in the scope of this dataset is the "Scaling to a range" using the min-max method since we have no extreme outliers values.

The resulting CSV file does not include the Domain and DNSRecordType features. All the columns with text, boolean or integers as the datatype were normalized. An example result row is illustrated in Fig. 5.1.

Figure 5.1: Normalized CSV file snippet

```
0,0,0,0,0,1,0,0,1,0.9469026548672567,0,0,0,0,0.9931459904043866,0,0.8,0,0.1532258064516129,0.7195767195767195,\\
0,0,0.6,0.1,0,0.2,0.13513513513513514,0.5714285714285714,0.044444444444444446,0,1,1
```

In this section, we described how the normalization of the data was done through well-known methods. This phase is very important in a ML context as a weak approach in data preparation can lead to unwanted results that are not coherent with the dataset content. In the next section, it will be presented the feature importance methods to extract the most important features from the dataset (normalized) and also it will be presented the application of ML classification algorithms over the selected features.

5.2 Machine Learning: Classification

In this section it is presented the method for feature selection / importance and also the application of supervised ML classification algorithms.

The ML process was divided in two steps: feature selection / importance and ML algorithms application.

In the first step, after the normalization of the dataset content as explained in the previous section, we used three different methods to select the features that have more impact and importance for the analysis. The feature selection process allows a better understanding of the data, better understanding of the model, and reduces the number of input features. There are numerous reasons to apply feature selection / importance methods such as:

- A faster ML training process;
- Decrease of the complexity of a model making it easier to understand;
- Possibility to improve the accuracy if the right subset is chosen;
- Reduces the overfitting problem (the use of irrelevant features).

Although there are numerous methods of feature selection / importance for different scopes, the following methods were selected based on the study of the state of art:

- Feature importance: This method uses the Extra Trees Classifier algorithm to fit and return the importance of the features;
- Univariate selection: Using a mathematical approach based on statistics this method works by picking up the best features on univariate statistical tests;
- Recursive Feature Elimination (RFE): According to [52], this method fits a model and deletes the feature (or features) with the lowest importance until the specified number of features is reached.

As demonstrated in [26], the behavior and performance of a feature selection algorithm depends strongly on the classification rule, sample size, and feature/label distribution.

The feature importance method makes use of the Extra Trees Classifier algorithm to fit and returns the importance of the features as shown in Code Snippet 5.3. According to the SkLearn official documentation for the Extra Trees Classifier [62], the feature_importances_ method calculates the impurity-based feature importance. Therefore, the most important

features represent the most higher values. After the model fit using all the features, a variable feat_importances saves all the values of the feature_importances_ method. Finally, it is returned a predefined number (feature_number variable) of the most important features.

Code Snippet 5.3: Feature Importance function

```
import pandas as pd
from sklearn.ensemble import ExtraTreesClassifier

def feature_importance():
    model = ExtraTreesClassifier()
    model.fit(X, y)
    feat_importances = pd.Series(model.feature_importances_, index=X.columns)
    return feat_importances.nlargest(feature_number).axes[0].values
```

The univariate selection method selects features according to the K highest scores and with chi-squared stats of non-negative features as input function. The chi-square method can be used to find the optimal subset of all features and to remove the ones with the lowest rank and was applied in past similar investigations such as [71] which purpose an intrusion detection model based on the fusion of this method and the Support-Vector Machine (SVM) ML algorithm. As shown in the Code Snippet, it was used the SelectKBest from SkLearn [64], the score_func parameter was set to chi2 and the k value represents the number of features to return (predefined feature_number variable). After the model fit, were created the dfscores and dfcolumns variables to save both scores related to the feature importance and the columns associated. This information was concatenated which allows to return the predefined number of features (feature_number) based on the higher score. The underlying code is presented in the Code Snippet 5.4.

Code Snippet 5.4: Univarate Selection function

```
import pandas as pd
from sklearn.feature_selection import SelectKBest

def univariate_selection():
```

```
bestfeatures = SelectKBest(score_func=chi2, k=feature_number)
fit = bestfeatures.fit(X, y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
featureScores = pd.concat([dfcolumns, dfscores], axis=1)
featureScores.columns = ['Features', 'Score']
return featureScores.nlargest(feature_number, 'Score').Features.values
```

The RFE (with cross-validation) method consists in the recursive processing of smaller and smaller sets of features to gather the features with the highest importance. It needs an estimator (in this case was used the Decision Tree Regressor) that will iterate and seek for the lowest feature importance and exclude it. The recursive method stops when the number of features left is equal to the desired number of features selected. Firstly, it was defined the rfecv variable from the RFECV method which contains the following parameters:

- Estimator: According to the SkLearn documentation [63], the estimator represents "a supervised learning estimator with a fit method that provides information about feature importance either through a coef_ attribute or through a feature_importances_ attribute." The selected Decision Tree Regressor algorithm fits the requirements allowing to get the feature importances using the feature_importances_ method;
- Step: If used an integer number (e.g. 1) this parameter represents the number of features to remove on each iteration. If the value is in a range between 0.0 and 1.0 then the step will represent the percentage of features to remove at each iteration;
- CV: Represents the cross-validation splitting strategy. In this case it was used the StratifiedKFold function and according to the official documentation [66] "this cross-validation object is a variation of KFold that returns stratified folds. The folds are made by preserving the percentage of samples for each class". The value 10 is based on the study of the state of art. If no value is defined, the default behavior for this parameter is the use of 5-fold cross validation;

- Verbose: If set to 1, the output will be more verbose;
- min_features_to_select: As the name implies, this parameter allows to set the minimum features to select. For this it was used the predefined variable (feature_number);
- n_jobs: According to the official documentation [63] this parameter represents the "number of cores to run in parallel while fitting across folds". It was used the number 4 because the processor used to run this task (described further in this chapter) only contains 4 cores.

After the definition of the parameters the model was fit and transformed using the method transform that allows reducing X to the selected features. Similarly to the previous method, it was created a data-frame to save each attribute and the associated importance. It was sorted by the importance in descending order and finally it is returned the top feature_number (predefined number of features desired). The code is presented in the Code Snippet 5.5.

Code Snippet 5.5: RFE function

```
import pandas as pd
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import StratifiedKFold
from sklearn.feature_selection import RFECV

def rfe_cross_validation():
    rfecv = RFECV(estimator=DecisionTreeRegressor(), step=1,
                  cv=StratifiedKFold(10), verbose=1,
                  min_features_to_select=feature_number,
                  n_jobs=4)
    rfecv.fit(X, y)
    rfecv.transform(X)
    dset = pd.DataFrame()
    dset['attr'] = X.columns
    dset['importance'] = rfecv.estimator_.feature_importances_
```

```
    dset = dset.sort_values(by='importance', ascending=False)
    return dset.attr.head(feature_number).values
```

The selection of the best features is a very important process especially when the number of features is large. It is necessary to remove the features with less importance to feed the ML algorithm with only the most important features to avoid overfitting, to decrease the training time, and to have a higher probability of improvement of the final results.

As presented, the first step is to select the most significant features and the second one is the application of ML algorithms and analysis of the metrics to determine the accuracy, precision, recall, F1 score, and the score time (seconds) achieved per algorithm.

The algorithms used to the evaluation were:

- SVM: According to [6], the SVM "accomplishes the classification task by constructing, in a higher dimensional space, the hyperplane that optimally separates the data into two categories". The SVM algorithm can solve both linear and non-linear problems by using a parameter named kernel which allows the researcher to set the more relevant mathematical function regarding the problem. According to the official SkLearn documentation [2], there are four distinct kernel functions: linear, polynomial, Radial basis function (RBF) and sigmoid. As the input dataset has a binary classification (0 or 1) and according to the study [36] where an identical dataset was used and comparisons were made between the four kernel types, the precision achieved was higher when linear kernel functions were used. Therefore on this evaluation was also used the linear kernel function;

- Logistic Regression (LR): The LR algorithm is a statistical regression method based on supervised learning and used for predictive analysis. Makes use of linear relationships between the independent features and the target label. According to [8], this algorithm has advantages such as simple implementation and excellent performance over linear datasets. Considering that one of the goals of this research is to have a solution capable of predicting malicious domains as close to real-time as possible, performance is a huge impact factor;

- Linear Discriminant Analysis (LDA): According to [16] the LDA is a commonly used method to classify data based on dimensionality reduction. The main goal of dimensionality reduction is to delete the redundant features in a dataset while retaining the majority of the data. Although it is used for predictive analysis in classification problems, there are vast use cases for this algorithm such as face recognition [43] and medical researches [29] and [27];

- K-Nearest Neighbors (KNN): As mentioned by DeepAI [41] "the k-nearest neighbors algorithm, or kNN, is one of the simplest machine learning algorithms. Usually, k is a small, odd number - sometimes only 1. The larger k is, the more accurate the classification will be, but the longer it takes to perform the classification". According to [32], the optimal value of k-nearest neighbors is different for distinct data samples. For this research, the number of the k-nearest neighbors parameter was the default (5) from the SkLearn framework [67];

- Decision Tree (CART): The CART term refers to Decision Tree algorithms that can be used on predictive problems to do classification and regression. For this research it was used the Decision Tree Classifier which according to the official SkLearn documentation [1] the main goal is "to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features". One of the main advantages of this algorithm is the simplicity of understanding and interpretation and also the excellent performance demonstrated in previous analyzes already mentioned in the state of art chapter;

- Naive Bayes (NB): The NB algorithms are used on supervised ML algorithms by applying Bayes Theorem. According to SkLearn documentation [3] there are three major models based on NB: Gaussian, Multinomial, and Bernoulli. The Gaussian is commonly used for classification problems. The Multinomial is more used on text classification problems and implements the NB algorithm for multinomial data. The Bernoulli method implements the NB algorithm according to multivariate data distribution mainly used for text classification. Therefore was selected the Gaussian NB model in order to classify and predict in a timely manner the input dataset.

These ML algorithms are often used in supervised machine learning classification prob-

lems, as already mentioned in the state of art, and the code to build each model is described in the Code Snippet 5.6.

Code Snippet 5.6: Model build

```
from sklearn import svm
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier


ML_ALGORITHMS = {"SVM": {'model': svm.SVC(kernel='linear', cache_size=500)},
                 "LR": {'model': LogisticRegression(max_iter=500)},
                 "LDA": {'model': LinearDiscriminantAnalysis()},
                 "KNN": {'model': KNeighborsClassifier()},
                 "CART": {'model': DecisionTreeClassifier()},
                 "NB": {'model': GaussianNB()}}
```

After the ML models building, the dataset was split between training and testing groups using 10-fold cross-validation. For the number of folds (10), the analysis of the state of the art was taken into account. The number of features (9) was based on repetitive evaluations, starting with only one feature and increasing the number for each iteration which allows us to detect the lowest improvement on the results after the ninth feature. The time metric evaluates the mean of the prediction time, for each iteration of the cross-validation. The computational performance must be taken into account. The experiments were performed on a computer with the following characteristics:

- Processor: Intel Core i7-8565U @ 1.80Ghz;
- Memory: SDRAM DDR4-2400 16 GB (16 GB x 1);
- Graphics Processor: Intel UHD Graphics 620;
- Storage: 512 GB PCIe Gen 3x4;

- Read up to 3,100 MB/s;
- Write up to 2,800 MB/s.

To run each ML algorithm we created a function that receives as input the dataset containing the best nine features using each one of the three feature selection / importance methods mentioned earlier in this chapter. To run the cross-validation process with the features and the class label was used the cross_validate function from SkLearn [65] which allows to "evaluate metric(s) by cross-validation and also record fit/score times". Finally, the result was appended to an array to produce the final table with all the results. The Code Snippet 5.7 illustrates the code used to evaluate each algorithm.

Code Snippet 5.7: Cross Validation function

```
from sklearn import model_selection
from sklearn.metrics import make_scorer, accuracy_score, recall_score, \
    f1_score
from sklearn.model_selection import cross_validate

label = dataset['Class']
scoring = {'accuracy': make_scorer(accuracy_score),
           'precision': 'precision',
           'recall': make_scorer(recall_score),
           'f1': make_scorer(f1_score)}
kf = model_selection.KFold(n_splits=k, random_state=None)
result = cross_validate(model, features, label, cv=kf, scoring=scoring)
table_data_k_cross.append([str(key),
                           str(k) + "-fold-cross-validation",
                           len(features.columns),
                           result['test_accuracy'].mean(),
                           result['test_precision'].mean(),
                           result['test_recall'].mean(),
                           result['test_f1'].mean(),
                           result['score_time'].mean()])
```

To evaluate the effectiveness of the ML models we choose common metrics as follows:

- Accuracy: The accuracy is a ratio between the correctly predicted observations versus the total number of observations;
- Precision: The precision is the ratio between the correctly predicted positive observations and the total predicted positive observations;
- Recall: The recall is the ratio between the correctly predicted positive observations and all observations in the actual class (0 or 1);
- F1 score: The F1 score is the average of precision and recall and takes into account the false positives and false negatives;
- Score time: Corresponds to the time for the test of the estimator on each cross-validation iteration.

The return of each ML function is an array with the results of each iteration and the result for the metrics is the average value.

The source code used to evaluate the feature importance and the ML algorithms is publicly available in [23].

5.3 Results and discussion

The results for each feature importance method are described in the Table 5.1, Table 5.2 and Table 5.3. As mentioned in the previous section, the test mode selected was 10-fold-cross-validation and the number of features selected was nine. The features vary for each feature selection / importance methods (feature importance, univariate selection and RFE).

In Table 5.1 results for the features that have higher importance using the feature importance method were: 'NumericRatio', 'ConsoantRatio', 'NumericSequence', 'HasSPFInfo', 'TXTDnsResponse', 'DomainLength', 'VowelRatio', 'CreationDate' and 'StrangeCharacters'.

Table 5.1: Results using the Feature Importance feature selection method

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Time (sec)**
SVM	0.912456	0.949	0.872111	0.898517	3.08712
LR	0.916622	0.949854	0.8788	0.905137	0.0142102
LDA	0.908822	0.950375	0.8616	0.894534	0.0122217
KNN	0.951833	0.965961	0.936311	0.946446	1.793
CART	0.947911	0.967106	0.926867	0.939979	0.0117228
NB	0.903156	0.947481	0.851622	0.887892	0.012998

Using the feature importance method, the ML algorithm with the highest accuracy (95%), recall (93%), and F1 score (94%) was the KNN. Although this algorithm has good results, the time to predict is above 1 second (1,79) losing to the CART algorithm with a score time of 0.01 seconds. Also, the precision was slightly higher for the CART algorithm.

In Table 5.2 results for the features that have higher importance using the univariate selection method were: 'HasSPFInfo', 'TXTDnsResponse', 'CreationDate', 'NumericRatio', 'LastUpdateDate', 'HttpResponseCode', 'ConsoantRatio', 'MXDnsResponse' and 'StrangeCharacters'.

Table 5.2: Results using the Univariate Selection feature selection method

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Time (sec)**
SVM	0.919911	0.940908	0.895222	0.909192	3.37574
LR	0.916711	0.949785	0.878156	0.904936	0.0143973
LDA	0.906233	0.957141	0.848222	0.889276	0.0131586
KNN	0.947333	0.966278	0.926156	0.940556	3.23797
CART	0.956011	0.969233	0.941756	0.952492	0.0133054
NB	0.908289	0.955469	0.854178	0.891494	0.013912

Using the univariate selection method, the algorithm with the best results was the CART. This algorithm shows a validation accuracy of 95% with a precision of 96%. The recall reached 94% and the F1 score 95%. Although the score time (0.0133) is considered

an acceptable value the LDA algorithm was slightly faster to predict with a score time of 0.0131 seconds.

In Table 5.3 results for the features that has higher importance using the RFE method were: 'NumericRatio', 'ASN', 'DomainLength', 'NumericSequence', 'Ip', 'ConsoantRatio', 'TLD', 'StrangeCharacters' and 'CountryCode'.

Table 5.3: Results using the RFE feature selection method

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Time (sec)**
SVM	0.910622	0.937951	0.878444	0.899988	3.2505
LR	0.913867	0.939699	0.883	0.903959	0.013961
LDA	0.907033	0.938683	0.869489	0.895892	0.0142611
KNN	0.957311	0.970312	0.943422	0.95398	1.36982
CART	0.961533	0.976461	0.945867	0.95891	0.0136005
NB	0.8975	0.936195	0.851222	0.884616	0.0127075

The results presented in Table 5.3 are the most interesting ones. They are based-on the CART algorithm and we were able to reach 96% of accuracy, 97% of precision with a recall value of 94% and F1 score of 95%. The time score (0.013) was also a good result losing only to NB algorithm (0.012) but the difference was almost insignificant.

The results obtained so far are encouraging. These results show that the algorithms are able to distinguish between benign and malicious domains with accuracy rates between 89% and 96%. Considering that the time to detect malicious domains is very important, the analysis presented also considers the time, in seconds, that each algorithm took to make the decision and calculate the mean value.

5.4 AutoML - Test and Comparison

The entire process done so far was done manually. From data preparation, selection of the most important features to the application of ML algorithms, none of these processes were carried out using Automated Machine Learning (AutoML) processes. According

to Microsoft [80] the AutoML "is the process of automating the time-consuming, iterative tasks of machine learning model development. It allows data scientists, analysts, and developers to build ML models with high scale, efficiency, and productivity all while sustaining model quality". In order to make a comparison between the results achieved trough the traditional and manual analysis with an AutoML method we decided to use the TPOT [73] tool that optimizes machine learning pipelines using genetic programming. This tool takes care of the following processes: data cleaning, feature-related tasks (selection, preprocessing, and construction), ML model selection, parameter optimization, and model validation providing at the end a sample Python code containing the ML pipeline optimized.

As presented in the Code Snippet 5.8 to use the TPOT Classifier the dataset (raw data) was divided into independent variables (X) and the class label (y). To be coherent from our previous experiments, the k-fold value was set to 10. For the remaining TPOT Classifier parameters we keep the default values.

Code Snippet 5.8: TPOT Classifier

```
from pandas import np
from sklearn.model_selection import RepeatedStratifiedKFold
from tpot import TPOTClassifier


dataset = np.loadtxt(Constants.dataset_path_final, delimiter=',', skiprows=1)
X = dataset[:, :-1]
y = dataset[:, -1]
y = y.astype(int)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
model = TPOTClassifier(generations=5, population_size=50,
                       cv=cv, verbosity=2,
                       random_state=1, n_jobs=4,
                       max_eval_time_mins=2)
model.fit(X, y)
```

The TPOT framework return a sample code with the ML pipeline optimized for the

given dataset as shown in the Code Snippet 5.9. This code receives as input a dataset in CSV format that is divided into features (independent variables) and the label class (dependent variable). After that, the "train_test_split" function receives the features and the class label as input to create random train and test subsets. The optimized pipeline has two estimators to run the process based on the Decision Tree Classifier which was also used on the previous experiments. The use of estimators allows to combine the results of the automated feature selection process. Finally, the pipeline fits the training sets to predict the testing set.

Code Snippet 5.9: AutoML code output

```
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.tree import DecisionTreeClassifier
from tpot.builtins import StackingEstimator
from tpot.export_utils import set_param_recursive

tpot_data = pd.read_csv(Constants.dataset_path_final, sep=',',
            dtype=np.float64)
features = tpot_data.drop('Class', axis=1)
training_features, testing_features, training_target, testing_target = \
    train_test_split(features, tpot_data['Class'], random_state=1)

exported_pipeline = make_pipeline(
    StackingEstimator(
        estimator=DecisionTreeClassifier(criterion="entropy",
                                         max_depth=10,
                                         min_samples_leaf=12,
                                         min_samples_split=7)),
    DecisionTreeClassifier(criterion="gini",
                                        max_depth=8,
                                        min_samples_leaf=16,
```

```
                                    min_samples_split=5)
)

set_param_recursive(exported_pipeline.steps, 'random_state', 1)
exported_pipeline.fit(training_features, training_target)
results = exported_pipeline.predict(testing_features)
```

The results using both single run and cross validation test modes are presented in the Table 5.4.

Table 5.4: Results using the AutoML optimized pipeline

Algorithm	**Test Mode**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Time (sec)**
Decision Tree (AutoML)	Single Run	0.979644	0.979663	0.979639	0.979644	0.037014
Decision Tree (AutoML)	10-fold-cross-validation	0.962967	0.973101	0.952267	0.959949	0.025705

In Table 5.5 are presented the 15 most significant features, achieved based on the mean importance for the 10 iterations in the AutoML application.

Table 5.5: Top 15 features from AutoML

Feature	**Importance**
NumericSequence	0.7733869657
ConsoantRatio	0.0798138327
DomainLength	0.0530893664
ASN	0.0263959398
Ip	0.0262614325
StrangeCharacters	0.0147163336
TLD	0.0071405632
CountryCode	0.0062722524
SubdomainNumber	0.0021005709
NumericRatio	0.0016877222
VowelRatio	0.0016387243
MXDnsResponse	0.0013082461
SpecialCharSequence	0.0012442882
RegisteredOrg	0.0007781148
Entropy	0.0005820750

From the results presented in Table 5.5 we can conclude that after the thirteenth feature (SpecialCharSequence), the importance starts decreasing to low values and are almost insignificant to the ML model. Therefore, a manual approach to ML model creation must take into account that the number of resources should not exceed thirteen. The AutoML algorithm has choose the Decision Tree Classifier as the best algorithm and estimators to the pipeline.

As shown in Table 5.3, although the feature selection was made using the RFE method, we also get the highest values using the CART algorithm. The accuracy only differs by decimal values (AutoML = 0.962 / CART (using RFE) = 0.961). The main difference was the hyperparameter tuning used by AutoML to reach the highest score. The time resulting from AutoML (0.025705) was slightly higher than our result (0.0136005). The combination of accuracy and time are important for a further implementation of the DNS firewall in a real scenario where, for instance, the highest accuracy algorithm can be devalued in favor of a better time response algorithm.

In this chapter we presented the data analysis process used to access the usefulness of ML to develop a DNS firewall solution. First a manual data analysis approach was applied. It considered the data preparation (normalization), feature selection by importance using three distinct methods (feature importance, univariate selection, and RFE) and the application of six different ML models (SVM, LR, LDA, KNN, CART, NB) and the posterior ML classification using different algorithms. Then an automatic approach was used (AutoML). The results obtained in the first approach were compared to the results from the AutoML approach using the TPOT framework. From the results we can now choose the best model to classify new domains towards the development of a DNS firewall solution. By choosing the best algorithm, and according the results the DNS firewall will be capable of detecting malicious domains with 96% of accuracy in a time range between 0.01 and 0.02 seconds. In the next chapter, we present the conclusion and the future work.

Chapter 6

Conclusion

The increasing of cyber threats over the last decades and the use of malicious domains to perform numerous of different attacks should aware both organizations as well as individual users to be more protected over DNS requests.

The main objective of our work is to implement a firewall solution based on ML to classify DNS requests and filtering them regarding the probability of being malicious or not. The development of DNS firewall based-on ML encompasses a set of steps such as the creation of a DNS dataset from DNS logs, the exploratory analysis of the dataset, the data preparation for the ML process, the feature selection by importance and the ML algorithms application to determine the classification accuracy. To accomplish these steps we firstly created a DNS dataset containing thirty-four features enriched using OSINT and analytical methods. The resulting dataset is publicly available [44] and can be used by others researchers. After this, the dataset was analyzed to correlate features with the class label providing valuable insights on how the data is distributed by malicious or benign domains. In order to be used by ML algorithms the data was processed making use of well known methods such as label encoding and min-max functions to normalize all the values. The feature importance process was made using three distinct methods (feature importance, univariate selection, and RFE) each one with different results. The ML algorithms were selected taken into account the type of analysis (supervised) and the state of art. Six different algorithms were used: SVM, LR, LDA, KNN, CART, and NB. The results were collected and analyzed and we verified that the CART algorithm is the best algorithm considering the RFE feature selection method.

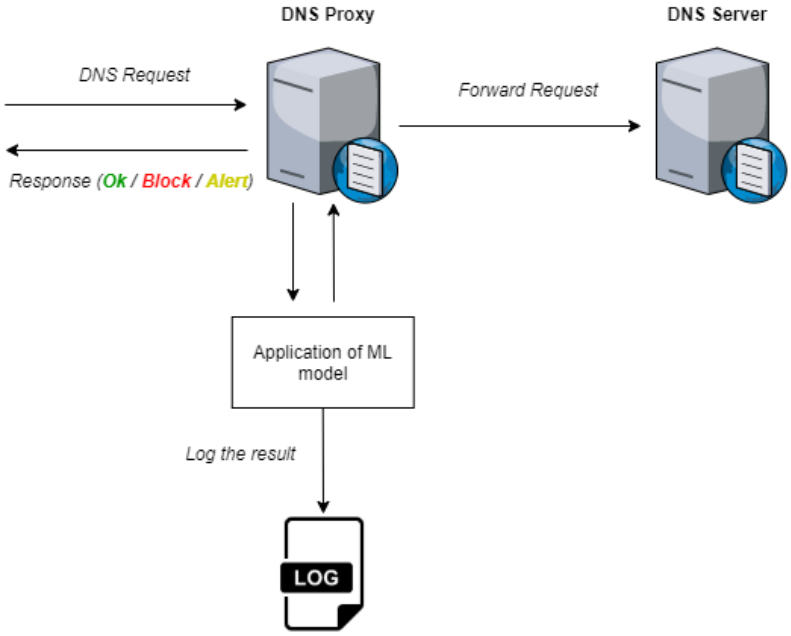
Considering the existence of automatic ML processes, the effectiveness of this type of processes was also analyzed. We used AutoML and a slight improvement in accuracy was observed but, as drawback an increase in the time to classify a set of domains was also observed.

Globally, with our approach we got accuracy results above 96% which is a good indicator to go further on the application in a real scenario. The results obtained from the ML algorithms can also be used by researchers for future research in this field.

Unfortunately, due to time constraints we did not implemented a real usage DNS firewall solution, but in our opinion all the conditions to build a viable solution are meet, even for the implementation of a in-band solution to detect malicious domains as they are requested to the DNS.

Since all the conditions to implement a DNS firewall solution based on ML are met, we plan our future work with the implementation in a real scenario as follows: (a) creation of a DNS proxy (which will work also as a firewall), (b) application of our ML model on each request, (c) log the results to a future passive analysis and (d) alert/allow/block system to inform the user. The figure 6.1 describes the proposed future work.

Figure 6.1: DNS Proxy/Firewall



This implementation can be a must to have system when applied over a current DNS firewall without ML approaches in real-time to classify domains. The possibility to improve

legacy systems is also a valuable point for users and organizations.

References

[1] *1.10. Decision Trees — scikit-learn 0.24.2 documentation.* URL: https://scikit-learn.org/stable/modules/tree.html (visited on 08/15/2021).

[2] *1.4. Support Vector Machines — scikit-learn 0.24.2 documentation.* URL: https://scikit-learn.org/stable/modules/svm.html (visited on 05/07/2021).

[3] *1.9. Naive Bayes — scikit-learn 0.24.2 documentation.* URL: https://scikit-learn.org/stable/modules/naive%7B%5C_%7Dbayes.html (visited on 07/12/2021).

[4] *2021 Cyber Attack Trends Mid-Year Report — Check Point Software.* URL: https://pages.checkpoint.com/cyber-attack-2021-trends.html (visited on 07/24/2021).

[5] *5 types of DNS attacks and how to detect them — Netsurion.* URL: https://www.netsurion.com/articles/5-types-of-dns-attacks-and-how-to-detect-them (visited on 07/22/2021).

[6] Mathias M. Adankon and Mohamed Cheriet. "Support Vector Machine". In: *Encyclopedia of Biometrics* (2009), pp. 1303–1308. DOI: 10.1007/978-0-387-73003-5_299. URL: https://link.springer.com/referenceworkentry/10.1007/978-0-387-73003-5%7B%5C_%7D299.

[7] Niels L. M. van Adrichem et al. "A measurement study of DNSSEC misconfigurations". In: *Security Informatics 2015 4:1* 4.1 (Oct. 2015), pp. 1–14. ISSN: 2190-8532. DOI: 10.1186/S13388-015-0023-Y. URL: https://security-informatics.springeropen.com/articles/10.1186/s13388-015-0023-y.

[8] *Advantages and Disadvantages of Linear Regression.* URL: https://iq.opengenus.org/advantages-and-disadvantages-of-linear-regression/ (visited on 08/05/2021).

[9] Oluwatobi Ayodeji Akanbi, Iraj Sadegh Amiri, and Elahe Fazeldehkordi. "A Machine-Learning Approach to Phishing Detection and Defense". In: *A Machine-Learning Approach to Phishing Detection and Defense* (Jan. 2014), pp. 1–100. DOI: 10.1016/C2014-0-03762-8.

[10] *Alexa - Competitive Analysis, Marketing Mix, and Website Traffic*. URL: https://www.alexa.com/siteinfo (visited on 07/22/2021).

[11] Amazon. *AWS Alexa Top 1M*. 2020. URL: http://s3.amazonaws.com/alexa-static/top-1m.csv.zip (visited on 09/25/2020).

[12] "An End-to-End View of DNSSEC Ecosystem Management". In: *26th USENIX Security Symposium* (Aug. 2017), pp. 14–21. URL: https://www.usenix.org/system/files/login/articles/login_winter17_03_chung.pdf.

[13] Chad Anderson, John 'Turbo' Conwell, and Tarik Saleh. "Investigating cyber attacks using domain and DNS data". In: *Network Security* 2021.3 (Mar. 2021), pp. 6–8. ISSN: 1353-4858. DOI: 10.1016/S1353-4858(21)00028-3.

[14] Dorde Antić and Mladen Veinović. "Implementation of DNSSEC-secured name servers for ni.rs zone and best practices". In: *Serbian Journal of Electrical Engineering* 13.3 (Oct. 2016), pp. 369–380. DOI: 10.2298/SJEE1603369A.

[15] Suranjith Ariyapperuma and Chris J. Mitchell. "Security vulnerabilities in DNS and DNSSEC". In: *The Second International Conference on Availability, Reliability and Security (ARES'07)* (Apr. 2007), pp. 335–342. DOI: 10.1109/ARES.2007.139.

[16] S. Balakrishnama and A. Ganapathiraju. "Linear Discriminant Analysis — A Brief Tutorial". In: *Institute for Signal and information Processing* (1998).

[17] Artur Barseghyan. *TLD*. 2020. URL: https://pypi.org/project/tld/ (visited on 09/28/2020).

[18] Z. Berkay Celik and S. Oktug. "Detection of Fast-Flux Networks using various DNS feature sets". In: *2013 IEEE Symposium on Computers and Communications (ISCC)*. 2013, pp. 000868–000873. DOI: 10.1109/ISCC.2013.6755058.

[19] Leyla Bilge et al. "EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis." In: 18th Annual Network and Distributed System Security Symposium, Jan. 2011.

[20] *Bind 9 - ISC*. 2021. URL: https://www.isc.org/bind/ (visited on 03/14/2021).

[21] J. Brownlee. *Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python*. Machine Learning Mastery, 2020. URL: https://books.google.pt/books?id=uAPuDwAAQBAJ.

[22] Kaouthar Chetioui, Ghizlane Orhanou, and Said El Hajji. "New protocol E-DNSSEC to enhance DNSSEC security". In: *International Journal of Network Security* 20.1 (Jan. 2018), pp. 19–24. DOI: 10.6633/IJNS.201801.20(1).03.

[23] *claudioti/machine-learning*. URL: https://github.com/claudioti/machine-learning (visited on 08/20/2021).

[24] *Common Ports*. URL: https://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-sg-en-4/ch-ports.html (visited on 08/22/2021).

[25] *Domain Name System (DNS) Security: Attacks Identification and Protection Methods*. URL: https://csce.ucmss.com/cr/books/2018/LFS/CSREA2018/SAM4137.pdf (visited on 08/22/2021).

[26] Edward R Dougherty, Jianping Hua, and Chao Sima. "Performance of Feature Selection Methods". In: *Current Genomics* 10.6 (Sept. 2009), p. 365. DOI: 10.2174/138920209789177629. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2766788/.

[27] Selma Elnasir and Siti Mariyam Shamsuddin. "Palm vein recognition based on 2D-discrete wavelet transform and linear discrimination analysis Big Data Review View project Knowledge management for adaptive hypermedia learning system View project". In: *Article in International Journal of Advances in Soft Computing and its Applications* (2014). URL: https://www.researchgate.net/publication/288225959.

[28] "FBI: Internet Crime Report 2020". In: *Computer Fraud & Security* 2021.4 (2021), p. 4. ISSN: 1361-3723. DOI: https://doi.org/10.1016/S1361-3723(21)00038-5. URL: https://www.sciencedirect.com/science/article/pii/S1361372321000385.

[29] Rongrong Fu et al. "Automatic Detection of Epileptic Seizures in EEG Using Sparse CSP and Fisher Linear Discrimination Analysis Algorithm". In: *Journal of Medical Systems 2020 44:2* 44.2 (Jan. 2020), pp. 1–13. ISSN: 1573-689X. DOI: 10.1007/S10916-019-1504-1. URL: https://link.springer.com/article/10.1007/s10916-019-1504-1.

[30] Zhihui Guo et al. "Botnet detection method based on artificial intelligence". In: *Proceedings - 2019 IEEE 4th International Conference on Data Science in Cyberspace, DSC 2019*. Institute of Electrical and Electronics Engineers Inc., June 2019, pp. 487–494. ISBN: 9781728145280. DOI: 10.1109/DSC.2019.00080.

[31] Philip Hane. *IPWhois*. 2020. URL: https://pypi.org/project/ipwhois/ (visited on 10/02/2020).

[32] Ahmad Basheer Hassanat et al. "Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach". In: *IJCSIS) International Journal of Computer Science and Information Security* 12.8 (2014). URL: http://sites.google.com/site/ijcsis/.

[33] *ICANN Research - TLD DNSSEC Report*. URL: http://stats.research.icann.org/dns/tld%7B%5C_%7Dreport/ (visited on 07/23/2021).

[34] *IDC 2020 Global DNS Threat Report — DNS Attacks Defense — EfficientIP*. URL: https://www.efficientip.com/resources/idc-dns-threat-report-2020/ (visited on 07/11/2021).

[35] *IDC 2021 Global DNS Threat Report — Network Security*. URL: https://www.efficientip.com/resources/idc-dns-threat-report-2021/ (visited on 08/22/2021).

[36] Putroue Keumala Intan. "Comparison of Kernel Function on Support Vector Machine in Classification of Childbirth". In: *Jurnal Matematika "MANTIK"* 5.2 (Oct. 2019), pp. 90–99. DOI: 10.15642/MANTIK.2019.5.2.90-99.

[37] *Internet Assigned Numbers Authority*. URL: https://www.iana.org/ (visited on 06/12/2021).

[38] Dmitry Ippolitov. *pydnsbl*. 2020. URL: https://pypi.org/project/pydnsbl/ (visited on 09/24/2020).

[39] Yong Jin, Masahiko Tomoishi, and Nariyoshi Yamai. "A Client Based DNSSEC Validation Mechanism with Recursive DNS Server Separation". In: *9th International Conference on Information and Communication Technology Convergence: ICT Convergence Powered by Smart Intelligence, ICTC 2018* (Nov. 2018), pp. 148–153. DOI: 10.1109/ICTC.2018.8539727.

[40] Manmeet Singh; Maninder Singh; Sanmeet Kaur. *TI-2016 DNS dataset*. 2019. DOI: 10.21227/9ync-vv09. URL: https://dx.doi.org/10.21227/9ync-vv09.

[41] *kNN Definition — DeepAI*. URL: https://deepai.org/machine-learning-glossary-and-terms/kNN (visited on 08/22/2021).

[42] Pierre Lison and Vasileios Mavroeidis. "Neural reputation models learned from passive DNS data". In: *Proceedings - 2017 IEEE International Conference on Big Data, Big Data 2017*. Vol. 2018-Janua. Institute of Electrical and Electronics Engineers Inc., July 2017, pp. 3662–3671. ISBN: 9781538627143. DOI: 10.1109/BigData.2017.8258361.

[43] Juwei Lu, Kostantinos N Plataniotis, and Anastasios N Venetsanopoulos. "Face Recognition Using LDA-Based Algorithms". In: *IEEE TRANSACTIONS ON NEURAL NETWORKS* 14.1 (2003). DOI: 10.1109/TNN.2002.806647.

[44] Cláudio Marques. *Dataset Creator*. 2021. URL: https://github.com/claudioti/dataset-creator.

[45] MaxMind. *GeoIP2 Databases*. 2020. URL: https://www.maxmind.com/en/geoip2-databases (visited on 09/26/2020).

[46] *Normalization*. URL: https://developers.google.com/machine-learning/data-prep/transform/normalization (visited on 06/12/2021).

[47] Gopinath Palaniappan et al. “Malicious Domain Detection Using Machine Learning on Domain Name Features, Host-Based Features and Web-Based Features”. In: *Procedia Computer Science*. Vol. 171. Elsevier B.V., Jan. 2020, pp. 654–661. DOI: 10.1016/j.procs.2020.04.071.

[48] Roberto Perdisci et al. “Detecting malicious flux service networks through passive analysis of recursive DNS traces”. In: *Proceedings - Annual Computer Security Applications Conference, ACSAC* (2009), pp. 311–320. DOI: 10.1109/ACSAC.2009.36.

[49] Daniel Plohmann et al. “A Comprehensive Measurement Study of Domain Generating Malware”. In: *Proceedings of the 25th USENIX Conference on Security Symposium*. SEC’16. Austin, TX, USA: USENIX Association, 2016, pp. 263–278. ISBN: 9781931971324.

[50] *Prevent spam, spoofing & phishing with Gmail authentication - Google Workspace Admin Help*. URL: https://support.google.com/a/answer/10583557?hl=en (visited on 07/29/2021).

[51] *Rapid7 Labs*. 2020. URL: https://opendata.rapid7.com/sonar.fdns%7B%5C_%7Dv2.

[52] *Recursive Feature Elimination — Yellowbrick v1.3.post1 documentation*. URL: https://www.scikit-yb.org/en/latest/api/model%7B%5C_%7Dselection/rfecv.html (visited on 07/02/2021).

[53] Fangli Ren et al. “A DGA domain names detection modeling method based on integrating an attention mechanism and deep neural network”. In: *Cybersecurity 2020 3:1* 3.1 (Feb. 2020), pp. 1–13. ISSN: 2523-3246. DOI: 10.1186/S42400-020-00046-6. URL: https://cybersecurity.springeropen.com/articles/10.1186/s42400-020-00046-6.

[54] *rfc4033*. URL: https://datatracker.ietf.org/doc/html/rfc4033 (visited on 08/02/2021).

[55] *SANS Internet Storm Center*. 2020. URL: http://web.archive.org/web/20200503151842/https://www.dshield.org/feeds/suspiciousdomains%7B%5C_%7DLow.txt (visited on 11/22/2020).

[56] V. Sathya Durga and Thangakumar Jeyaprakash. "An Effective Data Normalization Strategy for Academic Datasets using Log Values". In: *Proceedings of the 4th International Conference on Communication and Electronics Systems, ICCES 2019*. Institute of Electrical and Electronics Engineers Inc., July 2019, pp. 610–612. ISBN: 9781728112619. DOI: 10.1109/ICCES45898.2019.9002089.

[57] *Scikit Learn*. 2020. URL: https://scikit-learn.org/stable/ (visited on 09/15/2020).

[58] scmagazine. *Vast majority of newly registered domains are malicious*. 2019. URL: https://www.scmagazine.com/home/security-news/malware/vast-majority-of-newly-registered-domains-are-malicious (visited on 02/09/2020).

[59] Shun Segawa, Hideo Masuda, and Masayuki Mori. "Proposal and Prototype of DNS Server Firewall with Flexible Response Control Mechanism". In: *Proceedings - 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2019*. Institute of Electrical and Electronics Engineers Inc., July 2019, pp. 466–471. ISBN: 9781728116518. DOI: 10.1109/SNPD.2019.8935681.

[60] *Service Name and Transport Protocol Port Number Registry*. URL: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml (visited on 08/22/2021).

[61] A. K. Singh. "Malicious and Benign Webpages Dataset". In: *Data in Brief* 32 (Oct. 2020), p. 106304. ISSN: 2352-3409. DOI: 10.1016/J.DIB.2020.106304.

[62] *sklearn.ensemble.ExtraTreesClassifier — scikit-learn 0.24.2 documentation*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html (visited on 08/02/2021).

[63] *sklearn.feature_selection.RFECV — scikit-learn 0.24.2 documentation*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature%7B%5C_%7Dselection.RFECV.html (visited on 08/04/2021).

[64] *sklearn.feature_selection.SelectKBest — scikit-learn 0.24.2 documentation*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature%7B%5C_%7Dselection.SelectKBest.html (visited on 05/06/2021).

[65] *sklearn.model_selection.cross_validate — scikit-learn 0.24.2 documentation*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model%7B%5C_%7Dselection.cross%7B%5C_%7Dvalidate.html (visited on 08/22/2021).

[66] *sklearn.model_selection.StratifiedKFold — scikit-learn 0.24.2 documentation*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model%7B%5C_%7Dselection.StratifiedKFold.html (visited on 08/02/2021).

[67] *sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.24.2 documentation*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html (visited on 05/27/2021).

[68] *Spamhaus Botnet Threat Update: Q2-2021*. URL: https://www.spamhaus.org/news/article/813/spamhaus-botnet-threat-update-q2-2021 (visited on 08/01/2021).

[69] M. Srikanth Yadav and R. Kalpana. "Data preprocessing for intrusion detection system using encoding and normalization approaches". In: *Proceedings of the 11th International Conference on Advanced Computing, ICoAC 2019*. Institute of Electrical and Electronics Engineers Inc., Dec. 2019, pp. 265–269. ISBN: 9781728152851. DOI: 10.1109/ICoAC48765.2019.246851.

[70] *Sublist3r*. 2020. URL: https://github.com/aboul3la/Sublist3r (visited on 10/01/2020).

[71] Ikram Sumaiya Thaseen and Cherukuri Aswani Kumar. "Intrusion detection model using fusion of chi-square feature selection and multi class SVM". In: *Journal of King Saud University - Computer and Information Sciences* 29.4 (Oct. 2017), pp. 462–472. ISSN: 1319-1578. DOI: 10.1016/J.JKSUCI.2015.12.004.

[72] *The Spamhaus Project*. URL: https://www.spamhaus.org/ (visited on 03/22/2021).

[73] *TPOT*. 2021. URL: http://epistasislab.github.io/tpot/ (visited on 07/05/2021).

[74] Sriram Vajapeyam. "Understanding Shannon's Entropy metric for Information". In: *CoRR* abs/1405.2061 (2014). arXiv: 1405.2061. URL: http://arxiv.org/abs/1405.2061.

[75] Niels L.M. Van Adrichem et al. "DNSSEC misconfigurations: How incorrectly configured security leads to unreachability". In: *Proceedings - 2014 IEEE Joint Intelligence and Security Informatics Conference, JISIC 2014* (Dec. 2014), pp. 9–16. DOI: 10.1109/JISIC.2014.12.

[76] Verisign. *The Domain Name Industry Brief*. Volume 17, Issue 3. 2020.

[77] Florian Weimer. "Passive DNS Replication". In: *17 th Annual FIRST Conference* (2005). URL: www.enyo.de.

[78] *What is a top-level domain? — Cloudflare*. URL: https://www.cloudflare.com/learning/dns/top-level-domain/ (visited on 08/02/2021).

[79] *What is an autonomous system? — What are ASNs? — Cloudflare*. URL: https://www.cloudflare.com/learning/network-layer/what-is-an-autonomous-system/ (visited on 08/12/2021).

[80] *What is automated ML? AutoML - Azure Machine Learning — Microsoft Docs*. URL: https://docs.microsoft.com/en-us/azure/machine-learning/concept-automated-ml (visited on 07/02/2021).

[81] Norman Wilde et al. "A DNS RPZ firewall and current American DNS practice". In: *Lecture Notes in Electrical Engineering*. Vol. 514. Springer Verlag, June 2019, pp. 259–265. ISBN: 9789811310553. DOI: 10.1007/978-981-13-1056-0_27. URL: https://link.springer.com/chapter/10.1007/978-981-13-1056-0%7B%5C_%7D27.

[82] S. Yadav et al. "Detecting Algorithmically Generated Domain-Flux Attacks With DNS Traffic Analysis". In: *IEEE/ACM Transactions on Networking* 20.5 (2012), pp. 1663–1677. DOI: 10.1109/TNET.2012.2184552.

[83] Mattia Zago, Manuel Gil Pérez, and Gregorio Martínez Pérez. "UMUDGA: A dataset for profiling algorithmically generated domain names in botnet detection". In: *Data in Brief* 30 (June 2020), p. 105400. ISSN: 2352-3409. DOI: 10.1016/J.DIB.2020.105400.