MASTER'S THESIS

IMPROVED DGA-BASED BOTNET DETECTION THROUGH CONTEXT-RELATED FEATURE SELECTION BASED ON PACKET FLOW INFORMATION

Ruts, D

Award date:
2023

Improved DGA-based botnet detection through context-related feature selection based on packet flow information

Dominique Ruts

Date: 12/06/2023

Open Universiteit
www.ou.nl

IMPROVED DGA-BASED BOTNET DETECTION THROUGH CONTEXT-RELATED FEATURE SELECTION BASED ON PACKET FLOW INFORMATION

by

Dominique Ruts

in partial fulfillment of the requirements for the degree of

Master of Science
in Computer Science

at the Open University, faculty of Science
Master Computer Science

Course code: IM990C

Thesis committee: dr. ir. H. Vranken (chairman), Open University
dr. ir. C. Maathuis (supervisor), Open University

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis advisor, dr. ir. Harald Vranken, for his unwavering support and guidance throughout the course of this project. His valuable insights and constructive feedback have been instrumental in shaping the direction of my research.

I would also like to thank the members of my thesis committee, dr. ir. Clara Maathuis, for her thoughtful feedback and suggestions. Her expertise and knowledge have been invaluable in helping me to refine my ideas and strengthen my arguments.

I would like to thank my family and friends for their unwavering support and encouragement, both during my time at the university and throughout my life. Their love and support have been a constant source of inspiration and motivation.

Finally, I would like to thank the participants in my study for their willingness to share their time and insights. Without their contributions, this project would not have been possible.

Thank you to everyone who has contributed to this project in ways both big and small. Your support and encouragement have been invaluable, and I am grateful for all that you have done.

CONTENTS

Abstract iv

1 Introduction 1

2 Background 4
- 2.1 Botnets . . . 4
  - 2.1.1 DGA-based botnets . . . 6
- 2.2 Packet flows . . . 8
- 2.3 Machine learning . . . 8
  - 2.3.1 Baseline ML techniques . . . 10
  - 2.3.2 Ensemble methods . . . 11
  - 2.3.3 Feature selection techniques . . . 15

3 Related Work 17
- 3.1 DGA-based botnet detection . . . 17
  - 3.1.1 Selection of ML and DL methodologies . . . 17
  - 3.1.2 Focus on datasets and feature selection . . . 18
  - 3.1.3 A focus on explainable AI . . . 22
- 3.2 Non-DGA-Specific Botnet detection . . . 22
- 3.3 Building upon related work . . . 23

4 Research Design 26
- 4.1 Research Questions . . . 27
- 4.2 Research Method . . . 28
- 4.3 Data mining, validation, and analysis . . . 35
- 4.4 Infrastructure and software setup . . . 36

5 Validity and usability of available datasets 37
- 5.1 DGA and benign domain name datasets . . . 37
- 5.2 DNS registration information datasets . . . 37
- 5.3 Packet flow datasets . . . 39
  - 5.3.1 CTU-13 . . . 41
  - 5.3.2 NCC-2 . . . 45
  - 5.3.3 STA2018 . . . 47
  - 5.3.4 Merged dataset . . . 50
  - 5.3.5 IP2ASN-V4 . . . 51
- 5.4 Summary . . . 52

6 Selection of context-related features 55

7 Selection of ML methods 59

8 Performance of selected models and positioning against related work 66

9 Conclusions - Discussion, Validity, Future research, Reflection **69**
9.1 Answers to research questions 69
9.2 Contribution 72
9.3 Discussion and threats to validity 72
9.4 Future work. 74
9.5 Reflection 75

Bibliography **i**

Appendix A: Detailed process flow **vii**

ABSTRACT

The internet has brought numerous benefits to end-users, companies, and governments, but it has also led to severe threats to the security and privacy of data and systems. One such threat is botnets, which are networks of infected systems controlled by a central malicious machine. Botnets are used to spread various types of malware and attacks and can be considered very dangerous. The two most known identifiers of a botnet are its use of a C&C server to control the botnet and the need for a domain name or IP address for communication between the bots and the C&C server. Botnets nowadays use domain generation algorithms (DGA) to obscure the actual domain that will be used for communication, making it harder to detect the botnet. This research aims to improve DGA-based botnet detection based on packet flow datasets that include DGA-based botnet network activity. In doing so, the main research question is formulated to what extent machine learning models can be built for the detection of DGA-based botnets by using packet flow datasets and context-related feature selection methods. Improving the accuracy and speed of DGA-based botnet detection is crucial due to the increasing number of systems connected to the internet and the growing impact that successful attacks can have on the infected host systems. Many studies have already been performed in the field of DGA-based botnet detection by using traditional ML models, but not many of these studies use traditional ML models based on packet flow data. Especially not on a combined dataset derived from multiple other datasets. This research is conducted through an extensive literature study, a search, and consolidation to obtain workable datasets, several experiments to support the selection of feature sets and ML models, and a validation step to position the results against the outcomes of earlier work. Out of a selection of eight machine learning techniques, this study identifies three models as the best-performing models. These models are Bagging, XGBoost, and Decision Tree. The performance results are based on several experiments held against a combination of publicly available datasets and a dataset merged as part of the experiments in this study. The datasets contain packet flow information in which also the existence of DGA-based botnet behavior is available and labeled. Based on the results from the experiments, the conclusion is made that high-performing machine learning models can be built for the detection of DGA-based botnets by using packet flow datasets and the application of context-related feature selection. With only four features derived from the available datasets and extending these with two additional features based on the IP address, an XGBoost classifier is trained that reaches an accuracy of 99.72%, an AUC-score of 88.15%, and an F1-score on the positive class of 79.28%. The F1-score on positive class is deemed to be the most important measure for this study. The importance hereof is implied because the measure indicates how often a record is correctly classified as true-positive in a very imbalanced dataset. True-positive in this case represents actual DGA-based botnet behavior.

1

INTRODUCTION

Over the last decades, the Internet has grown at a fast pace when looking at the number of websites and services that are being provided to end-users, companies, and governments. The options are limitless, varying from a low-risk food ordering website of your favorite restaurant to the online banking system where you manage the financials of your entire family. Although the importance and criticality of each of these systems can be completely different, they all deliver a set of features to their end-users to improve the overall quality of life and make engagement with the company better. Unfortunately, the benefits of having all these connected services through mobile devices, laptops, and computers come hand in hand with fast-emerging and severe threats to the security and privacy of data and systems.

These threats already exist for ages, some even as old as the first computer solutions that were brought to market. They can also vary from simple misuse of the system by the end user to the unsolicited installation or activation of viruses, rootkits, worms, or botnets. A well-known example hereof is Conficker, a computer worm, with several variants, that was used in 2008 and which was targeting a flaw in the Microsoft OS software Porras [2009]. Although not directly unique in the way of infiltration the network or based on its malicious payload, the interest of researchers in Conficker mainly came from how it used different methods in such a flexible way and the fast evolution of Conficker that was witnessed. This adaptability resulted in Conficker becoming one of the widest-spread botnets in early 2009.

In a botnet, a network of infected systems is instructed via a central malicious machine, also called a Command & Control (C&C) server Beneš [2015]; Khattak et al. [2014]. This C&C server is the interface for a hacker or malicious user to control the botnet and initiate the spread of malware or other attacks throughout the internet. Because the botnet is a means to spread various types of malware and attacks, a botnet can be considered very dangerous.

Although botnets can distribute a variety of malware, or other attacks, they also have several common elements that enable the detection and prevention of a botnet. The two most known identifiers are first that they use a C&C server to control the botnet and second that a domain name or IP address is needed for the communication between the bots and the C&C server. If the botnet would make use of a hard-coded IP address or domain name for the communication between the bots and the C&C server, the botnet traffic would be easy to detect and communication could be traced quickly or even blocked entirely. Because the botnet aims to stay undetected as long as possible, most of the botnets make use of domain generation algorithms (DGA) to obscure the actual domain that will be used

for communication purposes which makes it harder to detect the botnet. More advanced botnets nowadays even use dynamic DGA, meaning that also the valid domain names will alter depending on either timestamp or other predefined parameters.

Taking down the communication between the bots and the C&C server would immediately take down the entire botnet. Due to the usage of DGA algorithms, this can be a difficult job to achieve. One of the most implemented ways of detecting and preventing attacks through a botnet is the usage of blacklists of DGA domains that are maintained by the so-called reputation providers Desmet et al. [2021]. These blacklists can then be used by security researchers, security companies, and ISPs to evaluate internet traffic against the blacklists. Even though this is a very valuable approach to detect and fight botnets, this remains a reactive countermeasure against botnets as these can only be used once internet traffic is already happening. This strategy is by now well-known by botnet creators and is losing its effectiveness due to updated strategies implemented in botnets Desmet et al. [2021]. Another approach for botnet prevention is to increase the difficulty for DGA algorithms to succeed, which is seen as a proactive methodology. To obtain a more proactive solution, researchers for example try to implement measurements that block the registration of malicious URLs.

For both the reactive and proactive approaches to botnet identification and prevention, we must act faster and more accurately to stay ahead of the rapidly changing botnets and their threats. This is the case since the creators of botnets and DGA algorithms evolve their coding and software architecture based on new technologies, system architectures, and preventive measures that become available. Due to the growing impact that successful attacks can have on the infected host systems, the detection speed of DGA-based botnets and other types of malware needs to increase. This growing impact of attacks is caused due to the increasing amount of systems that are connected to the internet and numerous solutions that are provided to end-users on which personal data, financial data, or any other type of confidential data are managed and transported. In addition, more critical transactions that can have a direct influence on people's lives are managed over the internet, e.g. health data and planning of treatments. Also, the accuracy of DGA-based botnet detection needs to be further improved. A higher accuracy will likely reduce the amount of false-positive or false-negative classifications, which in turn can help in a faster classification of malicious behavior and hence can contribute to faster detection of DGA-based botnets. This faster detection of botnets can result in earlier prevention. If we can bring the detection and prevention to a point before the botnet can execute its malicious payload, the impact on the hosts might be reduced or even eliminated.

The objective of this research is to enhance the detection of DGA-based botnets by utilizing aggregated packet flow datasets that contain instances of DGA-based botnet activity. The primary focus is on identifying and analyzing DGA-based botnet behavior within datasets that are pre-labeled, enabling the distinction of records associated with DGA-based botnets.

To provide meaningful contributions to the domain of DGA-based botnet detection, the research is conducted based on a merged dataset, derived by combining multiple packet flow datasets that have been used in earlier work. The process involves combining data from multiple sources to gain a comprehensive view of network traffic, enabling the creation of possibly more accurate machine learning models, and validating and refining existing detection techniques.

An answer is formed on how the performance metrics of ML models can be improved for DGA-based botnet detection by applying ML methods to these data sets, and how the performance of these ML models is positioned against earlier work. With performance metrics, later also referred to as performance of the ML models, the metrics "accuracy", "area under the curve (AUC)", and "F1 score" are referred.

This study does not address questions or topics related to further prevention of DGA-based botnets based on the obtained outcomes. The research was conducted through an extensive literature study, a search and consolidation to obtain workable datasets, several experiments to identify if models can be built with highly accurate results based on the context-related feature selection, and a positioning of the results of this work against the outcomes of a selection of earlier work.

In the remainder of this thesis, the second chapter provides more background information on important terminology and technologies. Section three describes the related work, what is known, and what is not known yet. The fourth section presents the research design, including the research method, research questions, and the data mining and validation approach that was applied. In section five, a summary of the datasets is provided. Section six discusses the selection of context-related features. In section seven, the literature review and experiments to select the ML methods are described. Section eight gives insight into the experiments held in this research and the results obtained from these experiments. Finally, section nine summarizes the answers to the research questions, discusses the contribution of this research, and reflects on future work.

2

BACKGROUND

This chapter provides the key concepts and theories about botnets in section 2.1, packet flows in section 2.2, and machine learning in section 2.3. It serves as an introduction to their fundamentals rather than a deep dive into their specifics. Its purpose is to provide readers with a basic understanding of the technologies before reading the remainder of this thesis.

2.1. BOTNETS

Botnets are networks of computers that have been infected with malware and are under the control of a central command and control (C&C) server. The C&C server is the interface for a hacker or malicious user to control the botnet and initiate the spread of malware or other attacks throughout the internet Vranken and Alizadeh [2022]. Botnets are a major threat to the security and stability of the internet, and they are becoming increasingly sophisticated and difficult to detect. The summary below is mostly based on the work of Khattak et al. [2014], who held in-depth research on botnets and their behavior. Figure 2.1 provides a visualization of a botnet in its easiest form, including a botmaster controlling multiple hosts (bots) via a C&C server.

Creators of botnets typically infect computers with malware through various means, such as phishing attacks, social engineering, or exploiting vulnerabilities in software. Once a computer has been infected, it becomes part of the botnet and can be controlled remotely by the botnet operator. The botnet operator can issue commands to the infected computers, such as sending spam emails, launching DDoS attacks, or stealing sensitive data. The infected computers can also be used to spread the malware to other computers, further expanding the botnet. Therefore, one of the first steps before a malicious botnet is created, is often that a hacker will try to find a use case to build and deploy his botnet. This use case is often tied to finding one or more vulnerabilities in a website, computer system, or application. A vulnerability can even be the misusage of a system by a person which exposes the users to the botnet. Once the vulnerabilities are identified, the hacker will try to build the botnet in a way that it later can exploit these vulnerabilities for the propagation of the botnet and deployment of the botnet payload.

Once the malware has been built and the first version is deployed, it will try to infect as many hosts as possible to make a large network of unsolicited bots waiting to get further instructions from the botmaster. More bots mean a higher success rate for the botnet. The

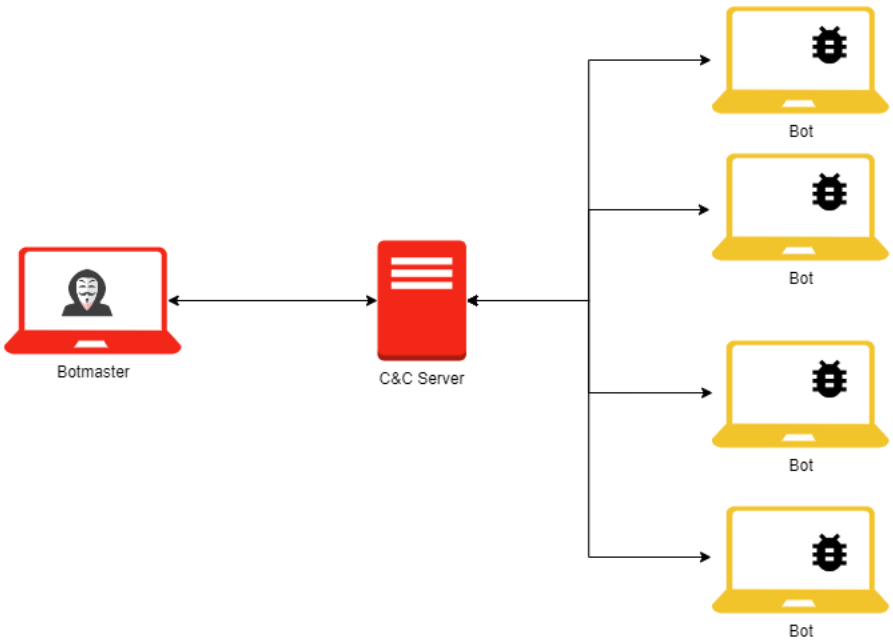


Figure 2.1: Botnet in its easiest form.

way to infect hosts can vary depending on the type of botnet. The easiest way for a botnet to propagate itself is when a user unknowingly takes an action that triggers the botnet infection, e.g. opening a link from within an e-mail or downloading software that includes a botnet payload. On the other hand, a botnet can also try to propagate itself trying to scan for backdoors or security loopholes in computer systems.

After the botnet can infect another system, the newly deployed bot will try to get in touch with the botmaster. To connect with that C&C server, the bot has to know the IP address or the domain name to connect to. In earlier versions of botnets, the IP address or domain name was often hardcoded in the malware. By using only the IP address and not a domain name, the botnet avoids using DNS lookups and hence reduces the risk of getting caught. However, once the IP address of a botnet is known, it is very easy for ISPs or network administrators to block or blacklist the respective IP address and thus shutting down the botnet.

To overcome that risk by using only the IP address, a domain name could be used instead. Although its usage could be detected quicker because of the DNS lookups, techniques like fast-flux exist to avoid detection. By applying fast-flux, the botnet uses a dynamic DNS system to change the underlying IP address to which the domain name resolves regularly. By implementing this strategy, it is no longer sufficient to block or blacklist one IP address.

A more advanced strategy, built upon fast flux, is called double flux. The principle of changing IP addresses remains the same, but in a double flux strategy also the name server that is responsible for looking up the IP addresses will dynamically and regularly be changed.

Even with fast flux and double flux, botnets can still be rather easily detected and blocked. This can be done by for example blacklisting the domain names used by the botnet or by using a DNS sinkhole. Blackisting results in all traffic related to the botnet being blocked. A DNS sinkhole does not block the traffic but instead routes the traffic of the botnet to an

IP that is not part of the botnet and thus not in control of the botmaster. The latter, using a DNS sinkhole, also reduces the risk that a botnet detects that its traffic is being blocked and additionally can help in taking over a botnet to kill it completely.

Malicious users that try to build or distribute botnets also have advanced their techniques, and that is where domain flux made its entry. By using domain flux, the botnets no longer use a single domain name for communication purposes between the bots and the C&C server. Instead, domain flux allows the botnet to continuously assign new domain names to the C&C server for its communication. Domain flux would be a difficult task without a mechanism to create new domain names randomly and automatically. This random generation of domain names is made possible by applying Domain Generation Algorithms (DGA).

2.1.1. DGA-BASED BOTNETS

Domain Generation Algorithms (DGAs) are algorithms that can automatically generate large numbers of domain names. When combined with the technique of domain-flux, which involves associating multiple domain names with a single IP address and rapidly changing the associated domain names, it is commonly used by botnets for quickly changing the domain names used for communication between the botmaster (C&C Server) and infected systems (bots) Plohmann et al. [2016]. The bots however do not know which are the valid domain names, so they will try to communicate with the C&C server based on domains in the generated list until they get a response. By applying these techniques, the botnet makes it much harder to detect and block communication with the C&C server. The information retrieved and represented about DGA in this section is derived from the work of Plohmann et al. [2016]. The generation of domain names by a DGA, when used in a botnet, is mostly based upon a variable known only to the botnet. This variable is also called a seed. The generation based on seed has to make sure that both the bots and the C&C server generate the same domain names to obtain a working communication channel with at least one of the domain names. The seeds can be either static, dynamic, and deterministic, or dynamic and non-deterministic.

Using a static seed is the easiest to detect, as the generation of domain names will always result in the same set of domains. When the seed is identified and the pattern of generating domain names is known, network administrators or security teams could generate the same list of domain names that the botnet would also generate. This generated list of domain names can then be used to block traffic from these domains and route the traffic to DNS Sinkholes, or other botnet prevention measures. Using a dynamic seed makes it a bit harder to block because the seed changes depending on certain characteristics like for example the current time. This in turn results in different sets of domain names depending on the nature of the seed. Although more difficult to block, once the deterministic pattern is known, network administrators or security experts can implement strategies to block the communication or even prevent the registration of these domain names.

The third variation of domain generation, using dynamic seeds that are non-deterministic, is the hardest to block or shut down. It uses non-deterministic characteristics like exchange rates or trending social media topics. The difficulty lies in the fact that these seeds cannot be precalculated, making blacklisting or blocking of domain names almost impossible. Although the implementation of dynamic seeds makes it harder to detect for security researchers and network administrators, the dynamic nature of the seeds also increases the

complexity for the botmaster. It will become harder to register a working domain name and open a communication channel between the C&C Server and the bots based on that domain. This is caused by the time window that is largely reduced and might not have a fixed duration for how long the seed remains valid. An example is when the trending Twitter post is being used as a dynamic seed, which can change randomly and multiple times throughout the day.

In addition to looking at how domains are generated by the use of different types of seeds, there is also a difference in the type of domain names that are being generated. Plohmann et al. [2016] have described four different types, as follows: Arithmetic-based DGA, Hash-based DGA, Permutation-based DGA, and Wordlist-based DGA. When using Arithmetic-based DGA, the domain names are generated based on either direct values or an offset thereof based on a hardcoded offset of characters based on an array. This type of DGA is also the most commonly used DGA algorithm type. The Hash-based DGA approach uses hashes to generate domain names. In particular, it is using the string representation of the hexadecimal digits of a hash. Often used hashes are MD5 or SHA256. The first two types are more random and easier to detect because of their usage of random characters and digits, making the domain names stand out against legit domain names. That is where the next two types are harder to detect. Permutation-based DGAs for example use a permutation technique on top of an initial domain name. These permutations then result in domain names that are similar or have a great resemblance to each other. Depending on the permutation scheme used and the initial domain name, this can already be more difficult to detect and stand out against valid domain names. More difficult to detect even, are Wordlist-based DGAs. These will concatenate a sequence of existing words from a wordlist. Due to this behavior, the domain names will look less random and hence camouflage the domain generation nature of DGA more than any other type of DGA. The wordlists are in turn either embedded in the botnet software (hardcoded and easier to detect or prevent) or retrieved via a public source (potentially updated from time to time, making detection and prevention more difficult).

DGA-based botnets are constantly evolving, and the algorithms used to generate the domain names are becoming increasingly sophisticated. As such, defending against botnets and DGA-based botnets requires a multi-faceted approach that includes technical measures, user education, and ongoing research and development to stay ahead of the latest threats. It is important to keep software up to date and to regularly patch known vulnerabilities. User education is equally important, as users are often the weakest link in the security chain. However, users can be trained to recognize phishing attacks, avoid clicking on suspicious links, and use strong passwords and two-factor authentication. Further technical measures include deploying firewalls, intrusion detection systems, and antivirus software to prevent infections and detect botnet activity. To support the development of these technical measures, advanced techniques such as machine learning and deep packet inspection are required. For instance, machine learning algorithms can be trained to detect patterns in the generated domain names, allowing security researchers to identify and block them. Deep packet inspection and feeding that data to ML algorithms can be used to analyze the network traffic and detect any communication with known or suspected botnet C&C servers. As botnets continue to evolve and become more sophisticated, it is important to stay vigilant and take proactive steps to protect against them.

2.2. PACKET FLOWS

Packet flow information refers to the movement of packets between clients and servers in a network. In computer networking, data is broken down into small units known as packets, each containing essential details for efficient routing and delivery across the network [1]. Packet flow data encompasses a wealth of information, including diverse network protocol headers such as IP, TCP, UDP, ICMP, and more. These headers encapsulate critical metadata that enable the identification and analysis of network traffic patterns.

In various datasets curated for network analysis, packet flow information is typically represented through bidirectional flow records. These records capture the essence of communication between network hosts, shedding light on key attributes such as source and destination IP addresses, ports, protocol types, timestamps, and other relevant features. By studying these flow records, researchers and analysts gain valuable insights into the behavior and characteristics of network communications[2].

In this study, datasets with aggregated packet flow data are used. Packet flow information with aggregated data refers to the collection and analysis of network packet-level data, where individual packets are grouped together and processed as a whole. Instead of focusing on individual packets, which can be voluminous and resource-intensive to handle, aggregated data combines multiple packets into larger units for easier analysis. This approach allows for a higher-level view of network traffic patterns, trends, and characteristics, providing insights into the overall flow of packets within a network. Aggregated data can include various metrics such as packet count, size, source and destination addresses, protocols used, and other relevant attributes.

Analyzing datasets information with aggregated packet flow data can provide valuable insights into network activity and can help to troubleshoot issues, detect security threats, or highlight suspicious activities. To detect DGA-based botnets, flow packets can be analyzed to identify patterns of communication between infected devices and the C&C servers. By analyzing the network flow data, it is possible to identify suspicious domains that are generated by DGAs and used by the botnet. Several techniques can be used to analyze network flow data for DGA-based botnet detection, including clustering, anomaly detection, and machine learning algorithms. These techniques can identify patterns of communication that are characteristic of DGA-based botnets, such as high volumes of traffic to previously unseen domains or domains with similar patterns of character generation.

2.3. MACHINE LEARNING

Machine learning is a subfield of artificial intelligence that involves training computer systems to learn from data and make predictions or decisions without being explicitly programmed to do so Kühl et al. [2019]; Wankhede [2022]. Machine learning algorithms are designed to analyze datasets, identify patterns, and make predictions based on those patterns. There are three main paradigms of machine learning: supervised learning, unsupervised learning, and reinforcement learning Kühl et al. [2022]; Praveena and Jaiganesh [2017]. Applications of machine learning include image and speech recognition, natural language processing, recommendation systems, and predictive modeling for areas such as finance and healthcare. Machine learning is a rapidly growing field with significant poten-

[1] https://en.wikipedia.org/wiki/Traffic_flow_(computer_networking)
[2] https://www.cloudflare.com/learning/network-layer/what-is-a-packet/

tial for improving many aspects of our lives.

Supervised machine learning is an ML paradigm in which the model is trained to learn from labeled data. The algorithm is provided with a set of input-output pairs, known as training data, and the goal is to learn a function that can accurately map new inputs to their corresponding outputs Dridi [2022]; Mahesh [2019]. The algorithm uses statistical techniques to identify patterns and relationships in the data and develop a model that can generalize to new, unseen data. Supervised learning is commonly used for tasks such as classification and regression, where the goal is to predict a categorical or continuous output variable based on one or more input variables. Examples of applications of supervised learning include spam filtering, image recognition, and medical diagnosis.

Unsupervised machine learning is also an ML paradigm in which the model is trained to identify patterns and structures in unlabeled data. Unlike supervised learning, there is no labeled output data provided to the algorithm Mahesh [2019]. Instead, the algorithm is given only input data and must identify patterns or relationships in the data on its own. Unsupervised learning algorithms use techniques such as clustering and Principal Component Analysis (PCA)[3] to group similar data points together or to identify the underlying structure of the data. PCA is a dimensionality reduction technique used in machine learning and data analysis to transform a high-dimensional dataset into a lower-dimensional space while preserving the most important information. Unsupervised learning is used in applications such as anomaly detection, customer segmentation, and natural language processing. It is particularly useful when labeled data is scarce or expensive to obtain.

Reinforcement learning is another ML paradigm in which a machine (also called an agent) learns to make decisions by interacting with an environment and receiving feedback in the form of rewards or punishments Mahesh [2019]. The goal of the agent is to maximize its cumulative reward over time. Reinforcement learning algorithms use trial and error to learn the optimal action to take in a given situation based on the feedback it receives. The agent learns through a process of exploration and exploitation, trying different actions to see which ones lead to the highest rewards. Reinforcement learning is used in applications such as robotics, game-playing, and autonomous vehicles. It is particularly useful in situations where the optimal strategy is not known in advance, and the agent must learn from experience.

There are many different types of machine learning models within each of the ML paradigms, each with its own strengths and weaknesses. Choosing the right model for a particular task requires careful consideration of the data, the problem, and the desired outcome. This study focuses on supervised learning techniques, given the presence of a classification problem that needs to be addressed. Experiments are held with several baseline ML techniques and ensemble methods. For the baseline ML techniques, Decision Trees, k-Nearest Neighbors, and Support Vector Machine are used. Looking at the ensemble methods, the models Ada Boost, Bagging, Histogram Gradient Boosting, Random Forest, and XBoost are used. This selection of techniques is made based on findings and relevance in the related work of van Renswou [2021] and Putra et al. [2022].

The information provided below on the different models is mostly derived from the Scikit-learn website[4].

[3] https://towardsdatascience.com/principal-component-analysis-pca-explained-visually-with-zero-math-1cbf392b9e7d

[4] https://scikit-learn.org/

2.3.1. Baseline ML techniques

Baseline ML techniques refer to simple and commonly used approaches that can serve as a starting point for the comparison and evaluation of more advanced models. Baseline methods provide a point of reference to assess the performance of new models or algorithms in a given task or dataset. Baseline models are typically straightforward and have minimal complexity. They may be selected based on their simplicity, ease of implementation, or popularity in the field.

Decision Trees

Decision trees are a type of machine learning model that uses a tree-like structure to make decisions based on a set of conditions. Each node in the tree represents a decision point, and the branches represent the possible outcomes of that decision Praveena and Jaiganesh [2017]; Talekar and Agrawal [2020]. The inner working of a decision tree is a three-step approach. The decision tree algorithm starts by splitting the entire dataset based on a selected feature and criterion such as information gain or Gini impurity. This splitting process is performed recursively, creating a binary tree structure where each internal node represents a feature and each branch represents a decision rule. The splitting continues until a stopping criterion is met. At this point, the algorithm assigns class labels to each leaf node for classification tasks or predicted values for regression tasks. To make predictions for new instances, the algorithm traverses the decision tree by following the decision rules based on the feature values of the instance. It continues until it reaches a leaf node, where the assigned class label or predicted value becomes the final prediction for that instance. This process allows decision trees to make predictions based on the learned patterns and decisions encoded within the tree structure. Decision trees offer interpretability, can handle mixed data types, and capture nonlinear relationships. To address the issues of overfitting and sensitivity to data variations inherent in decision trees, ensemble methods can be utilized, incorporating decision trees as base models to achieve enhanced performance. Decision trees are commonly used for classification and regression tasks and are easy to interpret and visualize.

k-Nearest Neighbors

K-Nearest Neighbors (k-NN) is a popular machine learning algorithm that falls under the category of instance-based or lazy learning methods Dridi [2022]. It is widely used for both classification and regression tasks and is known for its simplicity and intuitive concept.

The underlying principle of the k-NN algorithm is to make predictions based on the proximity or similarity of instances in the feature space. It assumes that instances with similar features tend to belong to the same class or have similar output values. k-NN does not involve a training phase like other algorithms; instead, it memorizes the entire training dataset for prediction purposes.

When given a new instance to classify or predict, k-NN identifies the k nearest neighbors in the training dataset based on a distance metric, typically Euclidean distance. The value of k is a user-defined parameter that determines the number of neighbors considered. The class label or output value of the new instance is then determined based on the majority vote or averaging of the labels or values of its k nearest neighbors.

k-NN offers several advantages as further discussed. It is a non-parametric algorithm, meaning it does not assume any specific underlying data distribution. This makes it robust

to outliers and versatile for different types of data. Additionally, k-NN can adapt to changing or dynamic environments since it recalculates predictions based on the current nearest neighbors.

However, k-NN's computational complexity can increase as the size of the training dataset grows since it requires calculating distances between the new instance and all training instances. This can be alleviated by using efficient data structures like kd-trees or approximate nearest-neighbor algorithms. Moreover, the choice of the distance metric and the value of k can significantly impact the algorithm's performance, and they should be carefully selected based on the characteristics of the data.

SUPPORT VECTOR MACHINE

Support Vector Machines (SVM) are used in various domains, including pattern recognition, classification, and regression tasks Dridi [2022]. SVM is particularly well-suited for solving complex problems where data points need to be classified into different categories or mapped to continuous output values.

The main principle behind SVM is to find an optimal hyperplane that separates data points of different classes with the largest possible margin. The hyperplane is a decision boundary in the feature space that maximizes the distance between the hyperplane and the nearest data points, known as support vectors. By doing so, SVM aims to achieve good generalization performance, minimizing the risk of overfitting.

One of the notable advantages of SVM is its ability to handle both linearly separable and nonlinearly separable data. This is achieved through the use of kernel functions, which transform the original feature space into a higher-dimensional space where a linear decision boundary can be found. By leveraging the kernel trick, SVM can effectively model complex relationships between features and the target variable.

SVM has proven to be robust to noise and outliers in the data due to its focus on the support vectors, which are the critical data points closest to the decision boundary. This robustness contributes to its ability to generalize well to unseen data.

While SVM has shown high accuracy and robustness, it is worth noting that the algorithm has some considerations. SVM can be computationally expensive, especially for large datasets, as it involves solving a quadratic optimization problem. Additionally, the interpretability of SVM models can be challenging due to the complex decision boundary representation in the transformed feature space.

2.3.2. ENSEMBLE METHODS

Ensemble machine learning methods are techniques that combine the predictions of multiple individual models to make more accurate and robust predictions. Instead of relying on a single model, ensemble methods leverage the diversity and collective intelligence of multiple models to improve overall performance Mahesh [2019]. Ensemble methods are widely used in various machine learning tasks, including classification, regression, and anomaly detection.

Ensemble methods, which combine the predictions of multiple models, offer several benefits. Firstly, they improve accuracy by leveraging the collective intelligence of the models, mitigating bias, reducing variance, and capturing different aspects of the data. Secondly, ensemble methods enhance robustness by being more resilient to noise and outliers. Outliers have less influence on the final prediction due to the collective decision-making

process. Additionally, ensemble methods excel in generalization by reducing overfitting through the combination of models trained on different data subsets or with different algorithms. Some ensemble methods, like Random Forest, provide feature importance measures, aiding in feature selection and data understanding.

However, ensemble methods also come with considerations. They can be computationally expensive when training and evaluating multiple models, especially for large datasets and complex models. The interpretability of ensemble models may be lower compared to individual models, as the combined decision-making process can obscure underlying factors. Hyperparameter tuning is crucial for optimal performance, as ensemble methods often have additional parameters to optimize, such as the number of models, their weights, or the combination strategy.

Random Forest

Random Forest is a versatile ensemble machine learning algorithm that has gained significant popularity in various domains, ranging from classification to regression and feature selection. It combines multiple individual models to make more accurate and robust predictions.

The core idea behind Random Forest is to create a multitude of decision trees and combine their outputs to make a final prediction. Each decision tree in the forest is built using a random subset of the original training data and a random subset of features at each split. This randomness introduces diversity among the trees and helps to reduce overfitting by capturing different aspects of the data Talekar and Agrawal [2020].

During the training phase, Random Forest constructs an ensemble of decision trees through a process called bagging (bootstrap aggregating). Bagging involves resampling the training data with replacement to create multiple bootstrap samples, which are then used to train individual decision trees. Each tree is grown until a certain stopping criterion is met, such as reaching a maximum depth or a minimum number of samples at a leaf node.

When making predictions, each tree in the Random Forest independently classifies the input data, and the final prediction is determined through a voting mechanism. For classification tasks, the class with the majority vote from the ensemble is selected as the predicted class. For regression tasks, the average or median of the individual tree predictions is taken as the final output.

Random Forest offers several advantages. It tends to have high accuracy and generalization performance due to the combination of multiple decision trees. It is robust to noise and outliers, as the ensemble-based approach reduces the impact of individual trees that may overfit on noisy samples. Additionally, Random Forest provides a measure of feature importance, allowing for feature selection and understanding of the data.

However, Random Forest also comes with some considerations. The algorithm can be computationally intensive, especially for large datasets and a large number of trees. Additionally, the interpretability of Random Forest models may be challenging, as it is not as straightforward to interpret the collective decision-making process compared to individual decision trees.

Ada Boost

AdaBoost (Adaptive Boosting) is a machine learning algorithm that belongs to the family of ensemble learning methods. It is designed to improve the performance of weak classifiers

by iteratively combining their predictions to create a stronger and more accurate model Praveena and Jaiganesh [2017].

The key idea behind AdaBoost is to focus on the misclassified samples during the training process. In each iteration, AdaBoost assigns higher weights to the misclassified samples, which emphasizes their importance in subsequent iterations. This adaptive weight adjustment allows weak classifiers to concentrate on the samples that are difficult to classify correctly.

During training, AdaBoost builds a sequence of weak classifiers, where each classifier is trained on a modified version of the training set. The modified version assigns higher weights to the misclassified samples from the previous iteration. The weak classifiers are typically simple models with limited predictive power, such as decision stumps (one-level decision trees) or shallow decision trees.

The final prediction of AdaBoost is obtained by combining the predictions of all weak classifiers. The weight assigned to each weak classifier depends on its performance during training. More accurate classifiers have higher weights in the final ensemble, resulting in a strong classifier that can provide more accurate predictions.

AdaBoost has a couple of advantages. It is capable of handling complex classification tasks and can achieve high accuracy even with a large number of weak classifiers. AdaBoost also has the ability to handle high-dimensional data and handle class imbalance issues by adjusting the sample weights. Furthermore, AdaBoost is less prone to overfitting compared to individual weak classifiers.

However, just like for other Ensemble methods like Random Forest, there are considerations to be aware of when using AdaBoost. It can be sensitive to noisy or mislabeled data, as the emphasis on misclassified samples can amplify their impact. Also, AdaBoost may be susceptible to outliers, as they can have a strong influence on the training process. Lastly, AdaBoost can be computationally expensive, especially if the weak classifiers are complex or the dataset is large.

Histogram Gradient Boosting

Histogram Gradient Boosting, also known as Histogram-Based Gradient Boosting (HGB), is a variant of the popular gradient boosting algorithm that utilizes histogram-based techniques to improve its efficiency and scalability. It is specifically designed to handle large-scale datasets with high-dimensional features and has gained attention for its ability to deliver competitive performance in various machine-learning tasks.

In traditional gradient boosting, decision trees are commonly used as weak learners to build the ensemble. However, the construction and evaluation of decision trees can be computationally expensive, especially for large datasets. Histogram Gradient Boosting addresses this challenge by utilizing histograms to approximate the splitting points of the decision trees, leading to significant speed improvements Ke et al. [2017].

The key idea behind Histogram Gradient Boosting is to discretize the feature space into bins or intervals, forming histograms that summarize the distribution of each feature. Instead of evaluating every possible splitting point for each feature, Histogram Gradient Boosting only considers splitting at the boundaries of the bins. This approximation significantly reduces the number of candidate splitting points, resulting in faster tree construction.

Furthermore, Histogram Gradient Boosting employs a technique called subsampling to further enhance efficiency. Rather than using the entire dataset to construct each tree, a

random subset of the data typically referred to as the "subsample," is used. This subsampling technique reduces both the memory and computational requirements, allowing for faster model training.

Histogram Gradient Boosting exhibits superior scalability and computational efficiency compared to traditional gradient boosting methods, making it well-suited for large datasets and high-dimensional feature spaces. Additionally, the histogram-based approach enables faster training and prediction times, enabling real-time or near-real-time applications. Despite its efficiency improvements, Histogram Gradient Boosting can still achieve competitive predictive performance on par with or even better than traditional gradient boosting.

It is worth noting that specific implementations of Histogram Gradient Boosting may have variations in their algorithmic details and additional features to further enhance performance. Examples include LightGBM and CatBoost, which are popular gradient-boosting libraries that incorporate histogram-based techniques for faster and more accurate predictions.

XGBoost

XGBoost (Extreme Gradient Boosting) is a highly efficient and powerful machine learning algorithm that is often used for both classification and regression tasks, designed to address the limitations of traditional gradient boosting algorithms. It incorporates several enhancements, including a more optimized and regularized approach, regularization techniques, parallel processing, and advanced tree construction methods, resulting in improved performance, scalability, and superior predictive accuracy Bentéjac et al. [2019].

The algorithm works by iteratively adding weak learners, typically decision trees, to the ensemble, with each subsequent learner aiming to correct the mistakes made by the previous ones. During each iteration, XGBoost calculates the gradients of the loss function and applies a form of gradient descent to minimize the loss. This process enables the model to continuously refine its predictions, resulting in a more accurate and robust final model.

XGBoost incorporates several key features that contribute to its effectiveness. First, it employs a technique called "gradient-based boosting," which optimizes the model by using gradient information to guide the learning process. Second, XGBoost employs a regularization approach known as "shrinkage" or "learning rate" that helps prevent overfitting by controlling the contribution of each tree to the overall ensemble. Additionally, XGBoost utilizes a novel technique called "column block" to enable parallel computation, making it highly efficient and scalable for large datasets.

The advantages of XGBoost are numerous. It consistently delivers state-of-the-art performance in various machine learning competitions and real-world applications. XGBoost is known for its ability to handle high-dimensional data, handle missing values, and effectively capture complex relationships between features and the target variable. Furthermore, it offers built-in capabilities for feature importance estimation and can handle different loss functions to accommodate various problem types.

However, it is important to note that XGBoost is not without considerations. It can be computationally expensive, particularly when dealing with large datasets and complex models. Careful parameter tuning is essential to achieve optimal performance, and the interpretability of XGBoost models can be challenging due to their complexity.

BAGGING

Bagging, short for Bootstrap Aggregating, is an ensemble learning technique that combines multiple individual models to improve prediction accuracy and reduce variance Kuncheva et al. [2002]. It is widely used in machine learning and has been successful in various domains, again including classification, regression, and anomaly detection.

The fundamental idea behind bagging is to create an ensemble of models by training each model on a different subset of the original training data. The subsets are created through a process called bootstrapping, where random samples are drawn with replacements from the original training set. This sampling process introduces diversity among the models, as each subset may contain different instances and exhibit variations in the feature space Kuncheva et al. [2002].

During training, each model is trained independently on its corresponding bootstrap sample. This means that each model sees a slightly different perspective of the data, and their predictions can capture different aspects or patterns within the dataset. The final prediction of the bagging ensemble is obtained by aggregating the predictions of all individual models. For classification tasks, the ensemble typically uses majority voting, while for regression tasks, it takes the average or median of the individual model predictions.

Bagging offers several advantages. It can effectively reduce the variance of the individual models, which helps to improve generalization performance and reduce overfitting. Bagging is particularly useful when the individual models have high variance or tend to overfit the training data. Additionally, bagging is robust to noise and outliers, as the ensemble decision-making process mitigates their impact on the final prediction.

However, it is important to note that bagging has its own considerations. While it can improve the accuracy and robustness of the models, bagging does not necessarily improve the interpretability of the ensemble. The combined decision-making process of the ensemble may make it challenging to understand the underlying factors driving the predictions. Additionally, the computational complexity of bagging increases with the number of models in the ensemble, making it more resource-intensive, especially for large datasets.

2.3.3. FEATURE SELECTION TECHNIQUES

Feature selection techniques are methods used to identify and select the most important features or variables from a dataset for use in machine learning models. These techniques are important because using too many features can result in overfitting, where the model becomes too complex and performs poorly on new data.

There are various types of feature selection techniques, including filter methods, wrapper methods, and embedded methods Vergara and Estévez [2015]. Filter methods involve selecting features based on statistical measures such as correlation or mutual information. Wrapper methods involve selecting features based on the performance of a particular machine learning model. Embedded methods involve selecting features as part of the model training process itself.

The choice of feature selection technique will depend on the specific problem and dataset, as well as the machine learning algorithm being used. Properly selecting and reducing the number of features can lead to more efficient and accurate models. In this study, the feature selection methods Sequential Feature Selection, SelectKBest, and Recursive Feature Elimination are used, which all can help in finding a suboptimal feature set for classification problems. In this study, the objective is to evaluate and compare the out-

comes of two different feature selection approaches: SelectKBest as a filter method and SFS as a wrapper method. To account for the computational challenges that arise with larger datasets, an additional wrapper method, RFE, is incorporated into the experiments. These methods have been selected based on a broad Internet search and reading through the Scikit-learn feature selection pages[5]. As part of the broad search, the highlighted feature selection techniques are found to be commonly used techniques. These three techniques are combined to form an ensemble approach that leverages different aspects of feature selection. The ensemble includes Sequential Feature Selection (SFS), which emphasizes the order and interactions of features, SelectKBest, which ranks features using statistical tests, and Recursive Feature Elimination (RFE), which eliminates features based on their individual contribution. By integrating these techniques, the ensemble approach provides a comprehensive analysis that considers feature order, statistical significance, and individual feature impact to create a robust and effective feature selection framework. While there are alternative feature selection techniques available, it is important to note that a comprehensive analysis of their performance has not been conducted in this study since the assessment of feature selection techniques is not its primary focus.

Sequential Feature Selection

Sequential feature selection (SFS) is a technique used in machine learning to select the most relevant features in a dataset. It works by iteratively adding or removing features and evaluating the performance of the model until a suboptimal set of features is identified. A suboptimal set of features, in the context of feature selection, refers to a feature subset that may not be the best or optimal solution but is still considered acceptable or reasonably good for a given task. Feature selection algorithms like SFS aim to identify the most relevant and informative features to improve the performance of a machine learning model. However, due to the complexity of feature selection problems and the trade-offs involved, it is often challenging to find the globally optimal feature subset that maximizes performance. This approach can improve the accuracy of the model while reducing the complexity of the dataset.

Select K Best

SelectKBest is a feature selection algorithm used in machine learning to select the K most relevant features from a dataset. It works by ranking the features based on their statistical significance, and selecting the K features with the highest score. This approach can improve the accuracy of the model and reduce the complexity of the dataset by removing irrelevant or redundant features.

Recursive Feature Elimination

Recursive Feature Elimination (RFE) is a feature selection algorithm used in machine learning to select the most relevant features in a dataset. It works by recursively eliminating the least important features based on their contribution to the model's performance until a suboptimal set of features is identified. This approach can improve the accuracy of the model and reduce the complexity of the dataset by removing irrelevant or redundant features.

[5]https://scikit-learn.org/stable/modules/feature_selection.html

3

Related Work

3.1. DGA-based botnet detection

Over the last decade, a lot of research already has been performed in the domain of botnet detection and prevention using DGA, based on machine learning or deep learning. All this research can be clustered into several subdomains, out of which most are classified into two major categories.

3.1.1. Selection of ML and DL methodologies

First of all, a big focus can be found on the research for ML / DL models to discover DGA botnets. Within this domain, a first subdivision is identified with a primary focus on research with ML models using features for botnet detection. Examples hereof are ML techniques used such as logistic regression (LR) and decision trees (DT) as proposed by Zhang et al. [2020]. In addition, and even more often applied in earlier work as indicated by Vranken and Alizadeh [2022], are the random forest (RF) and support vector machine (SVM) techniques. In recent work, RF and SVM models are for example applied in the work of Cucchiarelli et al. [2020]; Mao et al. [2020]; Zago et al. [2019]. Other models that are regularly used are gradient boost (GB) (Mao et al. [2020]; Zhang et al. [2020]), k-nearest neighbor (kNN) (Zago et al. [2019]), or multi-layer perceptron (MLP) (Cucchiarelli et al. [2020]; Mao et al. [2020]; Wu [2020]).

Next to the feature-driven ML methodologies, other research focuses more on feature-less DL algorithms Zago et al. [2019], Vranken and Alizadeh [2022]. In the field of studies that apply feature-less DL methodologies, the most common enumerator is the usage of a Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) technique. CNN's and CNN-based methods are used for example in the recent work of Highnam et al. [2021]; Namgung et al. [2021]; Ren et al. [2020]; Xu et al. [2019]. Building further upon the CNN approach as a step into dealing with temporal information and handling domain names and relevant metadata, the use and implementation of RNN (and hence Long Short-Term Memory or LSTM) is discussed in the research of Cucchiarelli et al. [2020]; Highnam et al. [2021]; Namgung et al. [2021]; Yilmaz et al. [2020]. Concluding on the research on ML and DL models, some studies consider both approaches and try to identify what the best-performing approaches are considering both ML and DL methodologies.

In addition, several literature reviews and surveys are available that focus on DGA-based botnet detection based on supervised machine learning techniques. An example

is the paper of Mac et al. [2017], who reviewed supervised detection methods and the accuracy of these methods. As part of the review, the authors also classify the methods based on the type of features used, which are either handcrafted or implicit features. The study uses datasets from Alexa (top 1 million domains) and the OSINT DGA feed from Bambenek Consulting. In addition, some DGA families were added to the dataset by generating domain names based on the algorithms used in these DGA families. Although many supervised ML algorithms can detect DGA-botnets, this study indicates that bidirectional LSTM and Recurrent SVM achieve the highest detection rate on both binary and multiclass classification problems.

3.1.2. Focus on datasets and feature selection

In contrast to the research done on ML or DL models, research is executed with an emphasis on finding or improving feature sets to make botnet identification easier, faster, and more reliable. This research domain is for example represented through studies in the area of dictionary extraction as per Pereira et al. [2018], character level-based detection as per Yu et al. [2018] and context-sensitive word embedding as per Koh and Rhodes [2018].

Context-free and context-aware features

Most of these studies that were conducted in the area of feature selection have a focus on context-free features and only a minimal amount of research is partially or fully focused on context-aware features Vranken and Alizadeh [2022]. The publications to date with a focus on context-aware features form a good foundation for further analysis but have some limitations as they are often tailored to a specific dataset or analysis methodology, for example, the usage of resolution rules specifically on the Rustock-botnet domain names as per Li et al. [2019] or the usage of temporal variation patterns (TVPs) as discussed by Chiba et al. [2017]. A temporal variation pattern refers to how the registration and usage of domain names can change over time.

Pattern recognition based on passive DNS information

When investigating literature on DGA-botnet detection in specific, many studies are found that use passive DNS data as input for feature selection and model training. The passive DNS data is either represented by datasets containing benign and malicious domain names, or datasets that provide a combination of domain names and registration data of the domain names. During the literature study, many were found in this field of play, as described in the below paragraphs.

One particular delineated domain within this field of research focuses on DGA-botnet behavior at times of domain name registration, with the attempt to identify the registration of malicious domains. These models and detection systems are trained by using for example domain profile features, registration history features, and batch correlation features in addition to DGA-based botnet data. A known approach to detect malicious domain names at the time of registration is to calculate the domain reputation. The calculated domain reputation aims to highlight if a domain might or might not be linked to malicious activities before further harm is done. A particular example hereof is the work of Hao et al. [2016]. Their model uses 22 features, derived specifically from domain profile features, registration history features, and batch correlation features, to distinguish between legitimate and abusive domain registration behavior. A Convex Polytope Machine (CPM) is used as the supervised machine learning model. CPM maintains an ensemble of linear sub-classifiers

and makes its final decision based on the maximum of all of their scores. Another approach to detecting DGA-based botnets at the time of registration involves studying large amounts of domain registration data to identify malicious campaigns. These studies focus for example on one or more specific top-level domains (TLD) and attempt to identify malicious entities that register domains for one-time use with malicious intent. An example is the study of Vissers et al. [2017], who studied 14 months of registration date from the .eu TLD and claim that they were able to identify 80.04% of these registrations and map them to large malicious campaigns using clustering algorithms. Building on top of the work of Hao et al. [2016] and Vissers et al. [2017], further research was conducted trying to answer some of the unanswered challenges like ground truth imbalance and blacklist incompleteness. One particular study took a distinct approach by also using registrant data to improve their system compared with earlier solutions. Their research resulted in a security system that detects DGA-botnets at the registration time of domain names that is called "Premadona". Desmet et al. [2021]

Classification and unsupervised clustering techniques based on passive DNS information

Another focus area to identify DGA-botnet behavior is using classification and unsupervised clustering techniques based on passive DNS information alongside features retrieved from the domain names, like Whois information, BPG information, DNS zone information, related IP addresses, AS information, and honeypot analysis data. These clustering techniques aim to identify similar characteristics and group domains according to their similarities Weber et al. [2018], Antonakakis et al. [2010]. Some specified clustering techniques are Density-based spatial clustering of applications with noise (DBSCAN), K-Means, Birch, Ward Hierarchical Clustering, and Agglomerative Clustering Weber et al. [2018]. Antonakakis et al. [2010] also calculate reputation scores for new and unknown domain names, based on the characteristics derived from the passive DNS information. A low score is given to domains linked to malicious activities, while a high score is provided to domains linked to legitimate activities. With the system they have built, called "Notos", they claim to achieve a true positive detection rate of 96.8% and a false positive detection rate of 0.38%.

Lexical features and characteristics from DNS requests and logs

A wider explored domain within this study is based on research that uses data like NX Domain traffic, DNS logs, and DNS query information. By using this kind of data, two foremost streams of studies have been identified. The first stream primarily focuses on lexical features and characteristics that are derived from DNS requests and logs. Lexical features and characteristics can be identified as the domain length, usage or non-usage of specific characters in the domain name, Shannon entropy, known word rate, and vowel rate Truong and Cheng [2016], Soleymani and Arabgol [2021]. The setup of Truong and Cheng [2016] is based on analyzing character string features. Various ML methods (Naïve Bayes, J48 (DT), Random Forest, k-NN, SVM) were used to train their model, while validation happened with the 10-fold cross-validation and the percentage split methods. The datasets used are a combination of a self-created dataset by collecting DNS traffic data in a managed network, a list of benign domains collected from Alexa, and a list of malicious domains collected from sources like malwaredomains.com and analyzers such as Anubis. The overall result indicates that J48, which is a Decision Tree algorithm, gives the best performance with an accuracy rate of 92.3%. In addition to this research, Soleymani and Arabgol [2021]

based their study on the knowledge that botnets regularly send lookup queries to DNS systems. They compare several models and their accuracy, where the models presented are a stochastic forest model, a logistic regression algorithm, and an SVM algorithm. For all models, the experiments include runs with and without using PCA (Principal Component Analysis). A general outcome presented is that the decision tree and random forest algorithms have the best accuracy and precision when using PCA. For experiments without using PCA, again Decision Tree and Random Forest generate the best performance in terms of accuracy and precision, although the results are lower than when PCA was used. Extending on this earlier research, other studies have emphasized building detection methods that can be applied in a real-world setting. One example hereof is the detection approach suggested by Manasrah et al. [2022], who claim that their DGA-based botnet detection model can be placed before a recursive DNS (RDNS) server to monitor DNS query packets coming from network hosts. The approach to making their framework scalable is to only focus on abnormal DNS queries by distinguishing regular user behavior (looking at DNS queries) from DNS queries from bot hosts. In turn, the distinguished DNS queries are then used to identify if DGA-based bot hosts are present. This identification of DGA-based bot hosts is based on 4 ML classifiers. These classifiers are k-NN, SVM, DT, and ANN. No additional information regarding the architecture of the ANN is disclosed.

Similarities and behavioral patterns of DNS requests

The second stream is looking at grouping domains based on their similarities and behavioral patterns of DNS requests. One of the primary findings is that DGA's are often used only over a short amount of time and have a similar lifetime and query style Zhou et al. [2013]. Grouping the domains based on their similarities and behavioral patterns can be done on similarity characteristics like for example, the top-level domain Yadav et al. [2012] or what is called rare destination requests Oprea et al. [2015]. A rare destination request is identified as a new domain that was not visited yet from within the network before and then contacted by a limited amount of hosts from within that network. After the grouping, metrics are computed based on alphanumeric characteristics that are fed into the ML models. Input for these studies are primarily DNS Logs from either ISPs or enterprise networks and are sometimes enriched with DGA-based domain name datasets. In particular, Oprea et al. [2015] used 2 months of DNS logs captured by LANL (Los Alamos National Lab) for APT infections and 38TB of Web Proxy logs collected by an enterprise. Yadav et al. [2012] on the contrary used an ISP dataset (network traffic), a non-malicious DNS dataset (DSN crawl), a malicious domain name dataset obtained from BotLab, and a set of algorithmically generated domain names created using Kwyjibo (tool to generate pronounceable domain name, but using words that are not in the English dictionary). Oprea et al. [2015] presents results for LANL-simulated attacks with an overall 98.33% true detection rate and a false positive rate of 3.72%. For the web proxy logs, even discoveries of unknown malicious domains (98) were found. Some studies in this stream highlight that their models do not require malware samples of malicious domains to be effective, but the limitation of their work lies in the usage of HTTP/HTTPS traffic and the time-range verification that is used Oprea et al. [2015]. A similar finding can be summarized for Yadav et al. [2012], who claim that also unknown botnets can be detected by their model, but one of the limitations is that a large number of test words is used, which could be overcome by attackers by timing the domain fluxing.

Packet flow data as input

Next to the studies that use passive DNS data, which includes data collected through DNS lookups, sensors, etc., and allows building timelines of events, a lesser amount of literature is available with a focus on DGA-based botnet detection that uses packet flow information. With packet flow information, a reference is made to data derived from DNS queries and packet flows captured at the moment of the DNS query itself or when the packet flow was logged. This is in contradiction with research that focuses more holistically on regular botnet detection where packet flow data is more often found as the used datasets. In this area, studies are more focused on finding communication patterns and behaviors of botnet attacks in the computer network based on the available packet flows and query data. An initial example hereof is the work of Putra et al. [2022], whose main contribution is found specifically in the pre-processing phase of data to achieve detect botnet detection. This pre-processing is carried out including known techniques like data transformation, feature engineering with one-hot encoding, and data normalization. In their proposed model, they use machine learning algorithms like Decision Trees, Random Forest, Logistic Regression, k-NN, and Naïve Bayes. The study is conducted on two major datasets, NCC and CTU-13. In summary, the authors claim to have a detection accuracy of 99.99%. The novelty compared to other works is that they apply and validate their model on multiple scenarios of the CTU-13 dataset and they have put the effort into describing the pattern or botnet-specific characteristics. In addition to this research, DBod was introduced by Wang et al. [2017]. It is a DGA-based botnet detection scheme that uses DNS query behavior to detect the botnets. The study is performed on a DNS dataset collected from an education network, obtained over 26 months. Only the unsuccessful DNS queries from the dataset are used, on which the researchers perform clustering and group detection algorithms. The results of both studies indicate that both known and unknown DGA-based botnets can be detected in real-world networks, without the need to have prior knowledge of a botnet.

Surveys based on DNS data analysis

Surveys based on DNS data analysis have been conducted by another group of researchers. These surveys typically categorize detection approaches based on the sources of DNS data and also review how information from other sources enriches the DNS data. The methods and evaluation strategies used are also discussed Zhauniarovich et al. [2018].

When considering DNS data sources, reviews are conducted on where and how the data is collected. The "where" question is addressed through dimensions such as host-resolver and DNS-DNS. Host resolver refers to communication between the host and the resolver, while DNS-DNS is referring to communication between DNS servers. The "how" question is answered by indicating whether studies used a passive or active way of collecting data. Enrichment data types are also considered for classification, such as geolocation, ASN, and registration records. In addition, a distinction is made between malicious and benign domain name records when using Ground Truth for the classification and evaluation of DGA-based botnet detection approaches.

Detection methods are also reviewed in other studies, with viewpoints towards the features being used, the methods applied, and the presented outcomes. In most cases, features are engineered based on the availability of raw data rather than using the raw data directly. Detection methods can be classified as knowledge-based or machine-learning-based. The latter involves researchers using datasets with both benign and malicious domain names and applying machine learning techniques. Hybrid approaches combining

both methods are also present in several cases.

The outcomes of detection methods are also analyzed, with a difference between those that detect specific malicious behavior and those that focus on the detection of associations between domain names. Metrics and evaluation methods used in DGA-based botnet detection studies are also addressed, including TPR/Recall, FPR, TNR, FNR, Precision, Accuracy, F1-score, and AUC. Evaluation methods can be grouped into different buckets such as whole dataset, one round train-test split, leave p-out cross-validation, K-fold cross-validation, cross-networks validation, cross-time validation, and cross-blacklists validation Zhauniarovich et al. [2018]. Significant challenges in the field of surveys of DGA-botnet detection based on DNS data analysis are highlighted, including data availability, lack of agreed practices to define ground truth, the resilience of features, and adaptability of algorithms. The lack of established theoretical foundations to evaluate detection approaches is also noted.

The focus of some surveys on DNS data has been extended to include reviews of DNS traffic analysis, which often involves the usage of passive DNS data. For instance, the study by Al-Mashhadi et al. [2019] examines methods that use a Honeypot and approaches that rely on an IDS (Intrusion Detection System). In their review, they summarize the models associated with each technique and highlight the primary drawback of each one. The study suggests that various DNS features can facilitate botnet detection, but the specific features used depend on the model and context, such as the time period or the scope of available datasets.

3.1.3. A FOCUS ON EXPLAINABLE AI

On a more holistic level, there is also the study from Piras et al. [2022]. They performed a study towards explainable AI when looking at the detection of DGA-based botnets using ML methodologies. They contribute model explanations with a local or global nature, where a local explanation is made on a specific sample rather than an entire dataset. A global explanation in turn has a focus on an entire dataset or collection of samples. The techniques used for the experiments in this study are feature statistical analysis, partial dependence plots, summary plots, and force plots. They used DNS traffic as input data alongside domain lists from Alexa for benign domains and DGAArchive for malicious domains. The conclusions of their study, first of all, indicate that they have obtained a better understanding of feature distribution and the correlation between features. In addition, there seems to be a direct correlation as well between the selection of features and the ML models that are used, including also the availability or usage of data. All the conclusions brought forward demonstrate how context-dependent DGA botnet detection can be.

3.2. NON-DGA-SPECIFIC BOTNET DETECTION

Although research on non-DGA-specific botnets is slightly deviating from the purpose of this study, many of these studies contain important information to consider. On top, more studies can be found in this domain that use active DNS information to detect botnets and is a source of relevant information to work with. Several studies are using TCP, UDP, and IP traffic and header information as a foundation for their analysis and model training. There is a distinction to make between researchers that emphasize packet flow information van Roosmalen et al. [2018], van Renswou [2021], Kwon et al. [2016], Liang et al. [2022]

and researchers that tried to distinguish themselves using only low-level network features like TCP, IP, and UDP header information Poon [2018]. While some of the studies apply the more traditional machine learning models to detect the presence of botnets and botnet behavior in network traffic, others applied deep learning methodologies like neural networks, recurrent neural networks, and ladder networks for the same purpose van Roosmalen et al. [2018], Poon [2018]. Concluding on results, the best results were obtained by researchers that used the actual packet information. As an example, van Roosmalen et al. [2018] were able to report a detection accuracy of 99.7%. On the contrary, the study from Poon [2018] using only header information summarized that the accuracy of their proposed approach was lower than expected compared to earlier work and that the number of false positives was too high for the model to be seen as trustworthy. Given the huge datasets and large volumes of DNS traffic that is flowing through networks every day, Kwon et al. [2016] proposes a detection approach called PsyBog. It is a detection approach specifically tailored to detecting malicious behavior within large volumes of DNS traffic. PsyBog uses a technique called power spectral density (PSD) to discover major frequencies that can be a result of DNS queries performed by botnets. The scalability of PsyBog on large data volumes originates in the usage of timing information of DNS query generation primarily and mostly ignoring other information like the actual domain name. The datasets used for their research consist of a generated set of DNS traces in a controlled environment and a large number of real-world DNS traces collected from a recursive DNS server, an authoritative DNS server, and Top-Level Domain (TLD) servers. In summary, PsyBog detected 23 unknown and 26 known botnet groups with 0.1% false positives. This result also indicates that PsyBog does not need to have prior knowledge of a botnet or a botnet's behavior to detect botnets. From a slightly different angle, Liang et al. [2022] has emphasized the group characteristics of botnets for their proposed detection method. With this approach, they use the sequence of packet length as one of the major characteristics, based on which the similarity between sequences is calculated to identify botnet behavior. Through this methodology, the researchers claim to be independent of any protocols used by the botnet and they are also not affected by the encryption of network traffic. The dataset used for their research is the ISCX dataset. Their proposed model can be implemented at the boundary of a network to monitor network traffic that enters or leaves the designated network, and hence protect the network before a botnet passes the boundary.

3.3. Building upon related work

Concluding an extensive literature review, several papers provide a summary or survey of earlier work that can be an entry point for future research. Earlier, the paper of Vranken and Alizadeh [2022] was highlighted, which provides a summary of studies concerning botnet and DGA-botnet detection. Table 3.1 provides a brief overview of recent related work discussed in the preceding sections, serving as a baseline for this study. Although newer research is available, literature is selected that either adopts a similar methodology or has influenced the way of thinking. Previous studies have employed various models to identify DGA-domain names and botnet behavior, such as clustering, classification, neural networks, and Long Short-Term Memory (LSTM). The related work showcases a mix of studies that utilize machine learning models with feature selection and featureless deep learning models. Nonetheless, this study will concentrate on machine learning models that utilize feature selection. The application of supervised machine learning methods offers

the advantage of reducing training time and providing greater control in assessing model performance during the research. Feature selection plays a crucial role in enhancing the effectiveness of machine learning models and is performed prior to model training. By eliminating redundant or irrelevant features, feature selection ensures that only valuable features are considered. This not only improves model performance but also reduces the effort and time required for model training, enabling faster feedback loops throughout the entire process.

Related work	Year	Primary Approach	Main Data used
Desmet et al. [2021]	2021	Clustering, Classification	domain records, registrant information
Hao et al. [2016]	2016	Convex Polytope Machine (CPM)	domain profile, registration history and batch correlation
Vissers et al. [2017]	2017	Clustering, Classification	Domain records, DNS registrant information
Weber et al. [2018]	2018	Clustering, Classification	Passive DNS records, domain records
Antonakakis et al. [2010]	2010	Clustering, Classification	Passive DNS records, Zone- & network based features
van Roosmalen et al. [2018]	2018	Deep Neural network, Ladder Network	TCP, UDP, IP botnet traffic
Poon [2018]	2018	Bidirectional LSTM (BLSTM)RNNs	TCP, UDP, IP headers
van Renswou [2021]	2018	Clustering, Classification	TCP, UDP, IP headers
Putra et al. [2022]	2022	Clustering, Classification	TCP, UDP, IP headers

Table 3.1: Summary of related work with the primary approach and the type of data used.

The previous studies on DGA-based botnet detection have used various datasets, such as domain records, DNS records, registration information, and TCP/UDP/IP traffic. They have applied feature selection techniques based on domain length, character/digit frequency, registrant information, nameserver reputation, domain age, and IP information. However, these studies also suggest that more attention is needed for feature selection and identification of new features.

This research aims to answer to what extent machine learning models can be built for the detection of DGA-based botnets by using aggregated packet flow datasets and context-related feature selection methods. Context-related features are features that are selected and designed specifically for a particular context or problem domain. In light of this research, the problem domain is phrased as DGA-based botnet detection based on aggregated packet flow data that includes DGA-based botnet behavior.

The contribution of this study is found in the merging of multiple packet flow datasets and the application of context-related feature selection based on that merged dataset. By combining multiple datasets, the volume, and diversity of data increases. This increased diversity of data improves the feature selection process and enhances the accuracy and re-

liability of DGA-based botnet detection. This approach helps to address the challenges associated with DGA-based domain detection, such as the variability and complexity of DGA algorithms, and the noise and ambiguity in packet flow data. It provides a fresh view on selecting relevant features for ML model training based on an enriched dataset. In addition, this study provides another view on the evaluation and performance analysis of the trained models. Through extensive experiments and evaluations using packet flow datasets, the effectiveness and robustness of the DGA-based botnet detection method are assessed by means of the F1 performance metric on the positive target class. This provides valuable insights into the strengths and limitations of the approach and establishes a benchmark for future research in the field.

This study builds on earlier work, including the usage of passive DNS data for malicious behavior prediction Xu et al. [2014] and the proposal of a model to prevent malicious domain registration Desmet et al. [2021]. The work of Putra et al. [2022] and van Renswou [2021] are primarily used as a baseline in this study, with an aim to identify new findings and contribute to the domain of DGA-based botnet detection. The two studies are selected as the most relevant related work because of their insights and contribution to the existing research domain. Although they both focus on botnet detection in a broader sense, they demonstrate promising results for detecting botnets based on packet flow information. In addition, they use similar datasets as the ones selected for this research. They also use ML models for clustering and classification purposes, whereas other recent related work based on packet flows often uses deep learning methods.

Studying these works provides insights into the strengths and limitations of different detection methods, and identifies opportunities for future research to extend in the domain of DGA-based botnet detection. Furthermore, the selected works provide valuable references and can be used as benchmarks for evaluating the effectiveness and efficiency of results obtained in this work.

4

RESEARCH DESIGN

Many studies have already been performed in the field of DGA botnet detection, often with promising results. Each research has its boundaries in terms of selected botnets, used datasets, feature selection, or model implementation to make the task at hand manageable in the ever-changing environment where botnets reside in. The evolving environment and the fast pace with which botnets are changing make it difficult for researchers and security experts to stay ahead. Given the abundance of existing research and proposed preventive measures in the field of DGA-based botnet detection, malicious users can leverage this knowledge and established machine learning techniques to evade detection. Therefore, researchers must constantly explore novel approaches to identify and block (DGA-based) botnets and their activities. In this study, an answer is provided to how different ML models can be built for DGA-based botnet detection and how the results of these models can be positioned against earlier work. Throughout the study, ML techniques are trained and evaluated based on aggregated packet flow datasets. Better model performance can help in a better and faster classification and detection of malicious botnet behavior. The packet flow information is used to highlight patterns and DGA-specific behavior in the captured network information. This faster detection can in turn contribute to earlier prevention mechanisms when used in future work.

This study's findings enhance DGA-based botnet detection by combining several packet flow datasets labeled for DGA-based botnet behavior. Additionally, a context-related feature selection approach is utilized, and ML techniques are selected based on the derived features to improve botnet detection. The context-relativity refers to feature and model selection through multiple supervised (e.g. Select K Best and Sequential Feature Selections) selection techniques depending on the availability, accuracy, and size of the datasets. To zoom in on the context-relativity, usable features are identified from each specific dataset individually, and how ML models perform when they are trained with these feature sets. To bring more value to context-relativity, experiments are held on how ML models perform when evaluated against other datasets than the one from which the feature sets are derived.

The research is characterized as a combination of extensive literature review and empirical research. This approach entails a comprehensive examination of existing scholarly works and research studies related to the topic at hand, providing a strong foundation of theoretical knowledge. Additionally, empirical research is conducted to gather firsthand data and evidence, enabling the study to draw practical insights and make informed con-

clusions. By merging the insights gained from the literature review with the findings obtained through empirical research, a well-rounded and holistic understanding of the subject matter is achieved.

4.1. Research Questions

The above-mentioned research aim and target contribution translate well into the main research question of this study, formulated as:

To what extent can machine learning models be built for DGA-based botnet detection by using aggregated packet flow datasets and context-related feature selection methods?

To formulate clear answers and recommendations that are beneficial for this research and potential further research, several subquestions will be handled to deliver specific and tangible results.

SRQ1: Which datasets are suitable to derive context-related features from packet flow information?

The first subquestion targets to find a list of useable packet flow datasets that include DGA-based botnet data. Packet flow data is a rich source of information for network analysis, containing important details about network protocols and traffic patterns. In datasets used for network analysis, packet flow information is often presented in the form of bidirectional flow records. These records provide valuable insights into host communication by capturing essential attributes like source and destination IP addresses, ports, protocols, timestamps, and other relevant features. A useable dataset for this research indicates a few characteristics. First of all, the dataset should include relevant data containing information about DGA botnets or their behavioral network patterns, to perform further experiments. Ideally, it should not be older than approximately five years to work with relevant information. In addition, the data in the different datasets have to demonstrate commonalities so that findings from one dataset can be transposed in some way on another dataset. In some cases, it is also required to merge datasets based on one or more features (e.g. data from multiple packet flow datasets have to be mergeable to generate a more complete and holistic dataset). Thirdly, the dataset needs to have sufficient records to train and test the ML models that will be used later. And finally, the datasets also need to have sufficient features with qualitative data to work with, as further discussed in section 4.2.

SRQ2: Which context-related features can be derived from packet flow datasets to improve the overall performance and accuracy of DGA botnet detection?

Subquestion two targets to find an answer on which context-related features can be derived from the datasets to improve the overall detection and accuracy of DGA-based botnet detection. Context-related features refer to features that are carefully chosen and designed to address the specific context and problem domain. In light of this study, these features are tailored to capture the unique characteristics of DGA traffic and network data, enabling more effective detection of DGA-based botnets. The feature selection is supported through the usage of multiple feature selection techniques tailored to the availability, accuracy, and size of the datasets. By selecting the features based on the context of the DGA-based botnet detection, the best fit-for-purpose feature sets can be used when training the models. Even though the feature selection is defined as a separate subquestion, the steps required

for answering this subquestion will involve performing experiments as further described for SRQ4.

SRQ3: WHICH ML METHODS CAN BE USED TO IMPROVE DGA-BASED BOTNET DETECTION WHEN USING CONTEXT-RELATED FEATURES DERIVED FROM PACKET FLOW DATASETS?
Similar to the second subquestion, a third question is defined but with a focus on the selection of machine learning techniques. The goal is to draw a conclusion on which ML methods can be used to improve DGA detection when using context-related features that are derived from the selected data sets. Initially, multiple ML models will be considered, implemented, tested, and validated based on the available datasets and features. Because finding the ML models with the best outcomes depends on the derived features, experiments will have to be held. Hence, the work performed for answering this SRQ goes hand in hand and will be answered through the experiments as described below for SRQ4. The outcome of this experiment will be used to select the best suitable models based on the derived features from the earlier described feature selection as part of SRQ2.

SRQ4: HOW DO THE PROPOSED MODELS PERFORM AND HOW CAN WE POSITION THE OUTCOMES OF THESE MODELS AGAINST RESULTS FROM EARLIER WORK?
The final component of this research to answer is whether a model can be built combining the outcome of the former two subquestions and what the performance and accuracy of detecting botnets look like with these models. The aim is to build models based on context-related feature selection with good performance rates. In addition, the evaluation will be made on how the outcomes of the proposed model can be positioned against the performance of ML models from earlier work. This positioning happens based on the outcomes of experiments in this study against the claimed results from two most related works.

4.2. RESEARCH METHOD

This research consists of several steps that contribute to answering the main research questions. These steps include an extensive literature review, a search and consolidation to obtain workable datasets, a validation phase on the conclusions of earlier research, and experiments to define the outcomes of the posed research questions.

To obtain a set of related literature, the literature review is handled in a controlled manner, with a focus on literature published in the last three to five years. Several known scientific databases such as IEEE Xplore, ScienceDirect, and ACM Digital Library are consulted as starting points to minimize the risks of getting too much or irrelevant literature. In addition, Scopus and Google Scholar are used for more generic searches and for finding additional literature. When performing the literature study, a defined set of keywords is used. These keywords include terms like "DGA", "Domain Generation Algorithm", "Botnet detection", and "Malicious domain" combined with "Machine Learning". The keywords are also extended with terms like "packet flow", and "netflow" because context-related feature sets based on packet flow data are created.

To facilitate further reuse and references of literature in this study, the Mendeley Reference Manager is used to store all papers with their respective reference. Specific tags are given to the papers to categorize them into their domain expertise, findings, and approaches. The metatags are mainly used to keep a condensed overview of the type of research conducted, the key ML methods implemented, the datasets used, and the relevant

outcomes the article has. By using these established literature collections and the Mendeley reference manager, both the risks introduced by the high amount of papers and the risk of limiting the reach of this study are minimized.

SRQ1: Availability and usability of datasets

The first subquestion to answer relates to the availability of datasets suitable for the study. Suitable datasets should provide enough information to merge with other data or already have DGA-based botnet behavior labeled data. Criteria used during the search for datasets are relevance, quality, size, diversity, availability, and ethics. Relevant datasets contain useful information, while high-quality datasets are accurate and representative of the problem domain. The dataset should be large enough to provide statistical power and diverse enough to avoid bias and overfitting in the machine learning models. Sufficient records is a rather hard-to-answer concept when looking for datasets because the amount of records required very much depends on the problem to be solved and the complexity of the challenge and therefore used ML models. An example hereof is the need for a few hundred records to obtain an optimal detection performance as proposed by Zhu et al. [2015]. In comparison, Figueroa et al. [2012] performs experiments by fitting 568 models to predict the sample size for classification performance. As a result, they can derive the need to have between 80 and 560 tagged samples depending on the dataset and sampling method. Additionally, the dataset should be easily accessible and available for other researchers to promote transparency and reproducibility in research. Obtaining the dataset in an ethical and legal manner without sensitive or personally identifiable information is also important.

The initial activity to address this question is addressed via an extensive literature review. Because of the quickly evolving landscape of DGA detection and prevention, the main focus is on literature published in the last three to five years to minimize the risk of working with outdated information and obtain more accurate results in the proposed experiments, not excluding relevant older work. Datasets for DGA-based botnet detection ideally should come from the same time period, as the DGA algorithms used by botnets can change over time. Literature however demonstrates that the creation of new datasets does not follow the same time evolution. Also in more recent research, packet flow datasets are used that include data that is ten to fifteen years old. Using datasets from different time periods may result in inaccurate or inconsistent results, as the DGA algorithms may have evolved or become more sophisticated over time. For example, if you are training a machine learning model to detect DGA-based botnets, using a dataset from several years ago may not accurately represent the current state of DGA-based botnets. In this case, the model may not be able to detect newer and more advanced botnets, leading to potential security vulnerabilities. Even though it is not possible to obtain all datasets from the same time period, it is still possible to use datasets from different time periods. In this case, it is important to carefully select the features, try to avoid using time-specific features, and compare these characteristics of the DGA-based botnets used in different time periods. Many groundworks have already been done on the literature of the last decades, including studies specifically tailored to the literature review of other papers. Two main studies in this domain are the ones from Zago et al. [2019] and Vranken and Alizadeh [2022], which are used as starting points for this study and are further used in the literature review. The scope of the literature study for this subquestion is to find datasets in the most recent related work and to identify if they are suitable for my research.

Next to the literature review, a broad search is performed to obtain qualitative datasets.

This search is primarily conducted on data repositories like Kaggle and Google Dataset Search, using search terms like DGA, Botnet, DNS, and WhoIs Information. In extension, a free text search on the Google search engine is performed with terms and dataset names obtained during the literature review. The search for datasets includes the datasets used in earlier research as well as datasets found during the specific dataset search. On the datasets that are found during the literature study and the extensive search, further investigation is needed by means of experiments to identify if these datasets contain the correct data and can be used for further research. For this part of the investigation, the data first has to be cleaned, processed, and analyzed. The preprocessing involves validation, standardization, formatting, transformation, matching, and cleaning of the data Alasadi [2017]. Data validation ensures the integrity and validity of the data, and it ensures the reliability of the results obtained. This is done by identifying missing values, outliers, and inconsistencies. These issues are then addressed by removing or handling the problematic data points appropriately. Next, data standardization ensures consistent representation of variables across datasets, including converting date formats, normalizing numerical values, and encoding categorical variables consistently. Data formatting ensures compatibility and consistency of data formats, with necessary conversions of data types like string to numeric or datetime formats. Data transformation involves necessary changes to make the data suitable for merging, such as aggregating data, creating derived variables, or encoding variables for compatibility. Data matching identifies common variables or identifiers for merging, ensuring consistent and compatible values. Finally, specific data cleaning techniques are applied, such as removing duplicates, handling missing values through imputation or deletion, and addressing outliers Alasadi [2017].

To get a better understanding of the cleaned data, an initial investigation of the data and identification of patterns happens through a structured Exploratory Data Analysis (EDA) approach. EDA is an approach, promoted by John Tukey in 1970, used to analyze data sets and identify the main attributes or features Cox and Jones [1981]. This approach includes anomaly detection, verification of outliers, and a check on several assumptions. Exploratory Data Analysis (EDA) goes beyond identifying data quality issues and encompasses a diverse set of data analysis techniques aimed at extracting descriptive statistics to uncover significant features and characteristics of the data. These statistics include essential measures such as the mean, median, mode, standard deviation, and variance of specific features. They provide valuable insights into the central tendency, variability, and distributional properties of the data. Further details and comprehensive discussions on these descriptive statistics are covered when describing the datasets, allowing for a more in-depth understanding of the data's key attributes and facilitating subsequent analytical tasks.

SRQ2: Context-related feature selection

For the second subquestion, an attempt is made to identify which context-related features can derived from the collected datasets to improve the overall performance and accuracy of DGA domain detection. Context-related features are features that are selected and designed specifically for a particular context or problem domain. In the context of DGA-based botnet detection, context-related features are selected based on the characteristics of DGA traffic and network data. Context-related features include various types of information, such as the domain name syntax, DNS query patterns, and network traffic characteristics. One important aspect of context-related features is that they are specific to the domain be-

ing studied. This means that the same features may not be applicable or effective for other domains or problem contexts. These features are used to train machine learning models that accurately classify DGA traffic and detect botnet activity. Performing feature selection before model training is important because it can help reduce dataset complexity by removing irrelevant or redundant features, improve model accuracy by training on a more focused set of data, and prevent overfitting by selecting the most important features. Overfitting can occur when a model performs well on training data but poorly on new, unseen data due to including too many features in the model.

Moreover, literature study is conducted to identify methodologies for feature selection and experiments are performed with the identified methodologies to derive usable feature sets from the datasets. This experiment runs in parallel with the search for datasets, as the usability of the datasets and their features is verified. Feature sets are created that are derived directly from the data sets without feature creation. The context-relativity of feature selection means the possibility to use certain features depending on the availability, accuracy, and size of data elements. Features are selected using a combination of supervised feature selection techniques to obtain usable feature sets depending on the availability and usability of the input data. Equally important to selecting the most relevant features is the identification of less useful features and the removal of these features from the feature sets. This identification of less useful features is performed by calculating the almost constant features and performing a feature correlation analysis. The goal is to apply an ML model with the highest performance depending on the selected features. Combined with the outcome of the literature study, this activity aims to form a conclusion on which context-related feature sets can be derived from the data. On a high-level note, there is not a single answer to the question as the context-relativity will be driven by the availability and accuracy of the data or depending on the architecture and approach of certain botnets.

SRQ3: ML METHOD SELECTION

An explicit choice is made to execute this research with ML techniques and predefined feature sets, instead of feature-less deep learning methodologies. The first decision herein is one of control, as the aim is to work with a controlled data set and to write a clear explanation about the use of packet flow information and the impact of context-related feature selection. Additionally, the usage of supervised ML methods allows for a shorter time in training models throughout the research, which contributes to shorter experiment loops and faster results to work with.

To answer the third subquestion related to the identification of ML methods that can be used in combination with the derived feature sets, a similar approach is held. For this purpose, the focus of the literature study goes to detection methods using ML and methodologies that are used for further validation and experiments. The list of ML methods includes ML algorithms that already proved successful in the domain of botnet and DGA-botnet detection in the related work of this study, as well as potentially less-used methods in earlier research towards DGA detection. The performance of the selected ML methods is validated throughout several experiments, where the earlier derived datasets and features are used as input.

SRQ4: PERFORMANCE OF PROPOSED MODELS AND POSITIONING AGAINST RELATED WORK

During the experiments to identify which ML methods can be used in combination with the derived feature sets, MLFlow[1] is used to capture the results of each experiment. MLFlow automatically registers all performance indicators of model training and evaluation, including parameters such as accuracy, recall, and F1-score, which give insight into how well the trained models classify DGA-botnet data. The tool also records parameters like run time and memory consumption of the models. In addition, MLFlow offers an easy-to-use comparison tool that allows for the comparison of outcomes of different models both in a textual and visual representation. This comparison within MLFlow makes it easier to analyze the results and detect the best-performing model.

The comparison of performance indicators, specifically performance indicators that relate to the model's ability to make correct classifications, should be treated and analyzed with care. It is not advisable to look at one particular performance indicator like accuracy or ROC score as it could be too limited and give a false indication of what a good model looks like. In this research, the focus is on three performance indicators combined to identify if a model is performing well or not: area under the curve (AUC), accuracy, and F1 score on the positive target class. AUC and accuracy are used to get a general understanding of the trained model's performance, as they indicate how well the model is performing on classifying data within the full data provided, for both negative and positive classes. Since highly imbalanced datasets are used, and the aim is to correctly identify DGA-botnet behavior, the F1-score on the positive class is added to the evaluation to measure how well the model identifies true positive records, which are the actual DGA-botnet records.

Accuracy measures the overall correctness of the classifier by considering the ratio of correctly classified instances (both positive and negative) to the total number of instances Sokolova and Lapalme [2009]. It provides a general measure of how well the model performs in terms of correct predictions. Accuracy is expressed as a value between 0 and 1, where 1 represents a perfect classifier with 100% accuracy, and 0 represents a completely inaccurate classifier.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

In this equation, True Positives (TP) represent the number of correctly classified positive instances, True Negatives (TN) represent the number of correctly classified negative instances, False Positives (FP) represent the number of negative instances that are incorrectly classified as positive, and False Negatives (FN) represent the number of positive instances that are incorrectly classified as negative.

F1 score on the positive class provides a measure of model performance specifically for correctly identifying positive instances, taking into account both precision and recallSokolova and Lapalme [2009]. The F1 score for the positive class is calculated using precision and recall specifically for the positive class. The mathematical equation for the F1 score is as follows:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4.2}$$

Precision is the ratio of true positives (TP) to the sum of true positives and false positives (TP + FP).

[1] MLFlow: https://mlflow.org/

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

Recall is the ratio of true positives to the sum of true positives and false negatives (TP + FN).

$$Recall = \frac{TP}{TP + FN} \tag{4.4}$$

The AUC (Area Under the Curve) is a metric used to evaluate the performance of a binary classification model Fawcett [2006]. It represents the area under the Receiver Operating Characteristic (ROC) curve. The mathematical equation for AUC is not a single formula. To calculate the AUC, the ROC curve is first constructed by plotting the True Positive Rate (TPR, also called Recall) against the False Positive Rate (FPR) at various classification thresholds. TPR is the ratio of true positives to the sum of true positives and false negatives, while FPR is the ratio of false positives to the sum of false positives and true negatives. Once the ROC curve is obtained, the AUC is calculated as the area under this curve. It represents the probability that a randomly chosen positive instance will be ranked higher than a randomly chosen negative instance by the model. An example ROC curve is provided in figure 4.1, collected from Wikipedia[2]. In this visualization, AUC is not explicitely shown, but it is respectively the area under each ROC curve.

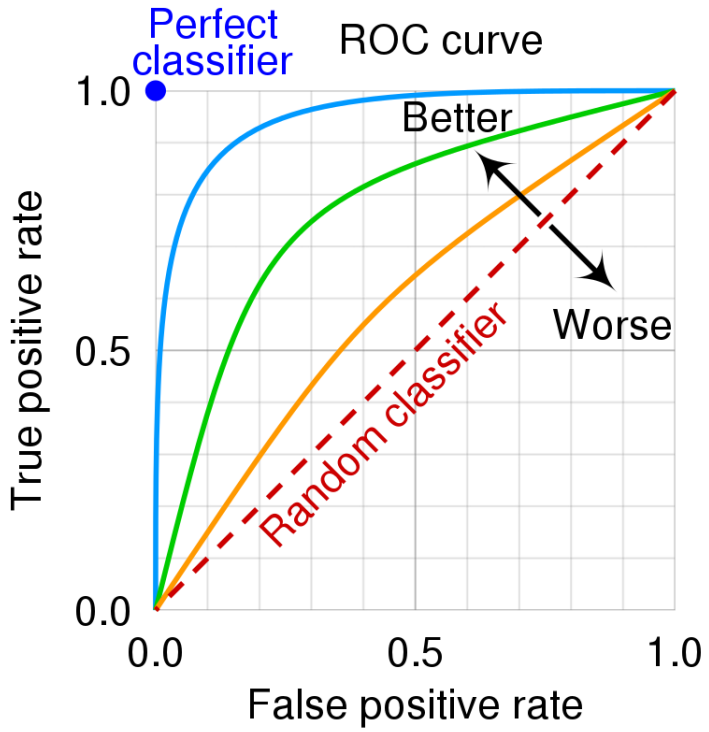


Figure 4.1: Example ROC curve visualized. Figure retrieved from the Wikipedia website.

This part of the experiment contributes to answering the main research question based on the earlier steps in the research. The aim is to answer to what extent models can be built for DGA-based botnet detection through applying machine learning techniques on

[2]https://en.wikipedia.org/wiki/Receiver_operating_characteristic

packet flow data, and which feature sets can be deduced as input for the machine learning techniques. In addition to finding an answer to the main research question, the experiment also contributes to finding potential shortcomings, pitfalls, and attention points of the proposed methodology of using a context-related approach for feature selection. This will be used to shed visibility on potential disadvantages and mitigate that going forward in further research.

Figure 4.2 represents a high-level schematic overview of the described steps in this and the previous two subsections, dealing with feature selection, model selection, and model evaluation. A detailed process flow is available in Appendix 1.

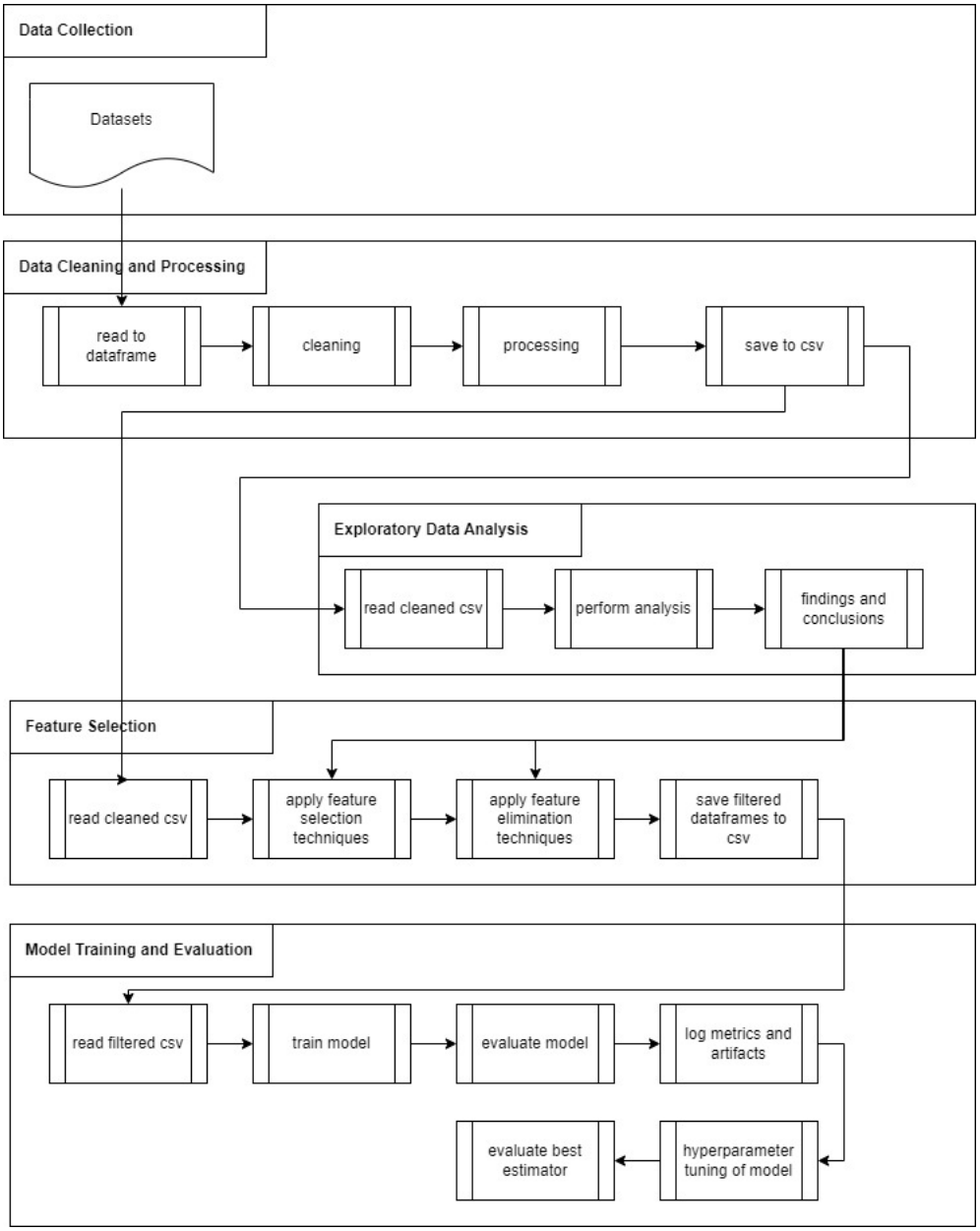


Figure 4.2: High level end-to-end ML process: From data cleaning to model evaluation

To conclude the evaluation of the proposed models and form an answer on the research contribution, a summary is made on how the proposed models are positioned against the

outcomes of earlier work, specific to recent related work. As related work, only academic papers are used, and it is assumed that the results of related work are valid and solid conclusions were presented in the literature. Due to time constraints, only two promising related works are considered for validation, and no further validation of earlier work is conducted. A comprehensive review is undertaken to identify relevant references that can be used to position this research against, focusing on the utilization of ML models and the availability of data and code in related works. The outcomes of this limited set of related work are subsequently used to position the outcomes of this research against, aiming to determine whether models can be proposed that have a high performance for DGA-based botnet detection and that potentially can improve the outcomes of earlier findings.

4.3. Data mining, validation, and analysis

Both the literature review and the gathering of datasets are forms of data mining. Data mining involves retrieving information from large datasets or collections of data, including literature. In the literature, data mining aims to gain knowledge of previous work on ML methodologies, datasets, and research outcomes, as well as relationships and dependencies between related works. When it comes to gathering datasets, data mining involves finding correlations or patterns between datasets themselves and how they can be used for the next steps in the research.

To follow a structured approach for these data mining activities, starting at the very beginning of the study up to writing the conclusion, the CRISP-DM[3] methodology is used. The model proposes 6 steps to execute in an iterative manner: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. For this research, only the first 5 steps of this model are used, where the last step called 'Deployment' is reflected in the summarized results of the research.

Validation of earlier literature conclusions is performed to act as a validity check on the used literature to ensure a solid and trustworthy baseline is in place. The baseline serves to evaluate which other research can be considered as actual related work. All academic papers are assumed to be trustworthy and have solid conclusions, making the hypothesis to be validated. It is not the intention to fully redo the work of other researchers, as it would be time-consuming and error-prone, with no positive contribution derived by doing so. The validation includes two promising related works, and their results are used to position the outcomes of experiments from this study against.

After the literature validation is completed, the preparation of data and analysis thereof happens. In the preparation phase, one or more workable datasets need to be generated out of the available datasets. The preparation steps include further data ingestion, cleaning and formatting of the data, and combining data elements. These steps are a critical part to make a workable and error-free dataset for further processing is available. After the preparation step, further analysis of the prepared dataset happens to validate the usability and correctness of the dataset. This analysis happens via the earlier mentioned Exploratory Data Analysis (EDA) approach. It can help to formulate an answer to questions like what algorithm to use on a dataset, how to refine features for a given use case, and when are you ready to start applying these ML models and features on the dataset. In addition, EDA also helps in demonstrating what data features or characteristics are better not

[3]Cross-industry standard process for data mining: https://www.datascience-pm.com/crisp-dm-2/

used or not as valuable as expected. The way how these outcomes are highlighted is often represented through data visualization techniques like statistical graphics. Examples of statistical graphics on data sets are box plots, histograms, bar charts, scatter plots, and heat maps.

4.4. INFRASTRUCTURE AND SOFTWARE SETUP

The validation and experimentation steps in this research are carried out using Jupyter Notebooks[4], within the Anaconda distribution[5]. The implementation is done in Python due to the numerous libraries available that have implemented different classifiers. The scikit-learn library is used specifically for evaluating the different algorithms. This library is widely used in the machine learning community and has a wide range of classifiers and utilities that can be used to evaluate the performance of the algorithms. Furthermore, it supports all the classifiers mentioned earlier, making it easy to evaluate the different algorithms. The experiments are conducted on a local machine, an HP Pavilion laptop, built with an Intel(R) Core(TM) i7-9750H CPU @ 2.60GH processor and 32GB internal RAM memory.

[4]Jupyter Notebooks: https://jupyter.org/
[5]Anaconda Distribution: https://www.anaconda.com/products/distribution

5

VALIDITY AND USABILITY OF AVAILABLE DATASETS

5.1. DGA AND BENIGN DOMAIN NAME DATASETS

To perform research on DGA-based botnets, datasets are obtained that have both benign and malicious domain names. Benign domain name datasets are retrieved from the open list TRANCO[1], which includes the top one million and most popular domains. The dataset only has domain names, so data processing is required on the data to get useful features. TRANCO generates its dataset by averaging the ranks from Alexa, Cisco Umbrella, Majestic, and Quantcast's source lists over the past 30 days. Even though benign domain name datasets are not directly related to DGA or botnets, they are required to train the ML methods. The malicious domain name datasets are obtained from multiple sources, such as Netlab[2], UMUDGA[3], and DGArchive[4]. The DGA datasets are downloaded on different occasions, including 21 October 2021, 21 November 2021, and 22 April 2022, to create a consolidated set for further research. The attributes found in these datasets are foremost domain names and a target label that indicates if the domain name is benign or related to malicious activity.

During the dataset search, it was discovered that some sources, such as DGArchive and Bambenek[5], no longer provide publicly open datasets. Explicit access and permission to retrieve the data from these sources need to be provided by the owners. In addition to the pure benign or DGA datasets, several datasets are obtained from previous research which include preprocessed lists of both benign and malicious DGA domains Cucchiarelli et al. [2020], Tran et al. [2018], and Marques et al. [2021]. An overview of domain name datasets that are identified and explored is summarized in Table 5.1.

5.2. DNS REGISTRATION INFORMATION DATASETS

DNS datasets are available on multiple sources and cover multiple time periods in the last decades, even up to the most recent days. In these datasets, a list of domain names is found

[1] https://tranco-list.eu
[2] https://data.netlab.360.com/feeds/dga/dga.txt
[3] https://data.mendeley.com/datasets/y8ph45msv8/1
[4] https://dgarchive.caad.fkie.fraunhofer.de/
[5] https://osint.bambenekconsulting.com/feeds/

Name	Type of domains	Records	Unique attributes	Data Time Period
Netlab	DGA	1,476,507	4	2021-2022
UMUDGA	DGA	+30,000,000	1 (plain lists)	2020
DGAArchive	DGA	Not Counted	5	2019-2020
Bambenek	DGA	Not Counted	2	2021-2022
Tranco Lists	Benign	1,000,000	1	2021 - 2022
Cucchiarelli et al.	Benign + DGA	674,898	3	2020
Tran et al.	Benign + DGA	169,847	8	2018
Marques et al.	Benign + DGA	90,000	34	2020

Table 5.1: Summary of investigated datasets with benign and DGA-based domain names.

together with attributes like the registration date, registrant information, technical contact details, and name server information. One example hereof is the datasets from Getwhoisdata[6], which are updated daily and available after payment. The freely available version is a limited dataset and contains information from 2017. The freely available datasets contain 150,910 records with relevant information spread across 90 attributes that can be mapped with other datasets of benign domain names. Another source for DNS datasets can be found on Whoxy[7]. It is similar in nature and also updated daily. The freely available dataset is smaller in size and contains approximately 20,000 records.

In addition to datasets that only contain a list of domain names, several datasets exist that already contain ready-to-use attributes and contain pre-processed information. As an example, the OpenIntel dataset includes DNS records for all domains in the .se, .nu, .ee, .sk, and .fr ccTLDs and all US Federal domain names from the .gov and .fed.us generic TLDs. Data is collected daily. Each file in this dataset includes 100000 domains, where each record represents a DNS query. This dataset consists of 106 unique attributes.

A summary of explored datasets with DNS registration information is found in table 5.2.

Dataset	Records	Unique attributes	Data Time Period
Ant[8]	Not counted	Depends on dataset	2016
Getwhoisdata (1 file)[9]	150,910	88	2017
Rapid7 (1 file)[10]	90,000	34	2021
Whoxy (1 file)[11]	21,319	58	2017

Table 5.2: summary of explored datasets with DNS registration information

Equal to datasets with benign and malicious domain names, also the datasets with DNS information have a large number of records. However, the merge of DGA datasets with DNS data does reduce the amount of valuable information a lot. This merge happens based on common variables, where variables present in both the DGA and DNS datasets are used as identifiers or keys for merging. For example, common variables include domain names, timestamps, IP addresses, or other network identifiers. Both DGA and DNS datasets contain missing, incomplete, or non-matcheable data. Missing values in key variables used for merging can lead to difficulties in matching and combining the datasets effectively. The

[6]https://www.getwhoisdata.com/
[7]https://www.whoxy.com/whois-database/

generated and merged datasets still contain a couple of hundreds or thousands of records that are useful for this research, but the remaining amount of data limits the usability for further experiments a lot.

One approach to merging the datasets is to identify matching identifiers like the domain name itself. Because the DGA datasets are mostly lists of domain names without additional attributes, it is too difficult to aggregate data based on other attributes.

5.3. PACKET FLOW DATASETS

Several datasets discovered during the literature study have been obtained, which already provide a combination of aggregated packet flow information and indications of (DGA-based) botnet behavior. Table 5.3 provides a summary of the datasets utilized in this research, namely CTU-13, NCC-2, and STA2018. It is worth noting that all of these datasets consist of data that ranges from eleven and twelve years old. These datasets have been chosen because they already contain aggregated and labeled packet flows with a truth label indicating whether the records are associated with malicious traffic, particularly botnets.

All three datasets incorporate several protocols for network traffic analysis. The top 3 represented protocols across the datasets are UDP, TCP, and ICMP. For both CTU-13 and NCC-2, most records are related to the UDP protocol, followed by TCP and ICMP. Most records in STA2018 are related to the TCP protocol, followed by respectively UDP and ICMP. TCP stands for Transmission Control Protocol, which ensures reliable and sequential delivery of data packets. Another protocol used is UDP (User Datagram Protocol), which offers a connectionless and less reliable means of transmitting data packets. Additionally, ICMP (Internet Control Message Protocol) is employed primarily for error reporting and diagnostics, such as ping requests and responses Kozierok [2005].

Datasets with aggregated packet flow data typically don't include Time-To-Live (TTL) attributes. When studying DGA, understanding the TTL value associated with a DNS record can help to distinguish between legitimate and malicious domain names. DGA algorithms generate a significant volume of domain names, some of which may not yet be registered or have a short lifespan. As a result, the DNS records associated with these domains either have a very short Time-to-Live (TTL) value, or attempts to resolve non-existing domains lead to an NX-domain response. In contrast, legitimate domains typically have much longer TTL values, commonly measured in days or weeks Sivaguru et al. [2020].

In the context of researching DGA-based botnets with aggregated packet flow data, Time-to-Live (TTL) attributes are not available. Previous successful work in the field also indicated that TTL-attributes are not a hard requirement for DGA-based botnet detection and that other feature sets should be explored Sivaguru et al. [2020]. By not using TTL-values, this research aims to direct its focus towards alternative distinguishing attributes and leverage machine learning techniques to analyze aggregated packet flows, aiming to identify and exploit other characteristic patterns associated with DGA-based botnet traffic.

Dataset	Source	Records	Unique attributes	Data Time Period
CTU-13 (Scenario 5 + 13)	CTU University, Czech Republic	2.054.981	15	2011
NCC-2	Mendeley Data	14.779.083	18	2011
STA2018	University of St Andrews	1.537.285	550	2012

Table 5.3: Summary of packet flow datasets.

Based on the retrieved datasets, an experiment is performed to evaluate the usability of the datasets. The first part of the experiment involves cleaning and processing the raw data of each dataset, without adding more attributes or merging datasets. To begin with, the raw data files are read into a Pandas dataframe, first rows are checked to understand the data, and column names are transformed to lowercase for easier handling in later steps. Next, the dataset is split into a train, validation, and test set in a 70%, 15%, and 15% ratio, respectively, since only the training data will is cleaned. Then, a check for null or NaN (Not a Number) values is performed and these are replaced with a predefined value for easier handling. For attributes related to the number of bytes, the predefined value was set to 0.0. This is done to preserve the data integrity and that the missing values are explicitly accounted for. For other numerical attributes, the Pandas 'fillna'[12] function was used with the 'ffill' method. Hereby NaN or null values are replaced with the last valid observed value in that column. For a categorical feature like 'state', an arbitrary string like 'CON' was used. Null and NaN values can even generate errors during the training of some ML models. After cleaning up the null and NaN values, verification was performed to identify outliers. Outliers are often considered to be noise or anomalies that do not align with the targeted data. Removing them might help to reduce variability and avoid distorting the data distribution, hence improving the accuracy and reliability of the analysis. For outliers in the packet flow datasets in the context of DGA-based botnet detection, the outliers are not removed as they could be genuine instances of DGA-based botnet activity and can be useful for training the ML models. In the datasets used in this study, outliers were only found in approximately 0.1% of all records. The outliers were identified using the Tukey's fences method[13]. Tukey's fences method uses the interquartile range (IQR), which is the range between the first quartile (Q1) and the third quartile (Q3) of the dataset. Data points that fall below Q1 - k * IQR or above Q3 + k * IQR (where k is a constant typically set to 1.5 or 3, and IQR is Q3-Q1) are considered outliers. Another compelling argument for retaining outliers in the dataset is the utilization of various ensemble methods that demonstrate increased resilience to outliers. By incorporating ensemble techniques into the analysis, the models can effectively mitigate the potential adverse effects of outliers on the overall performance. Therefore, retaining outliers in the dataset is a viable strategy for leveraging the robustness offered by ensemble methods in handling outlier instances.

Depending on the available labels or target indicators in the dataset, a target class is added to indicate whether a record is linked to benign or malware behavior. As an example, for the CTU-13 dataset the feature "label" was used. If the label contains the word "virut", the target class is set to "1" to indicate that the record is associated with a DGA-

[12]https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html
[13]https://en.wikipedia.org/wiki/Outlier#Tukey's_fences

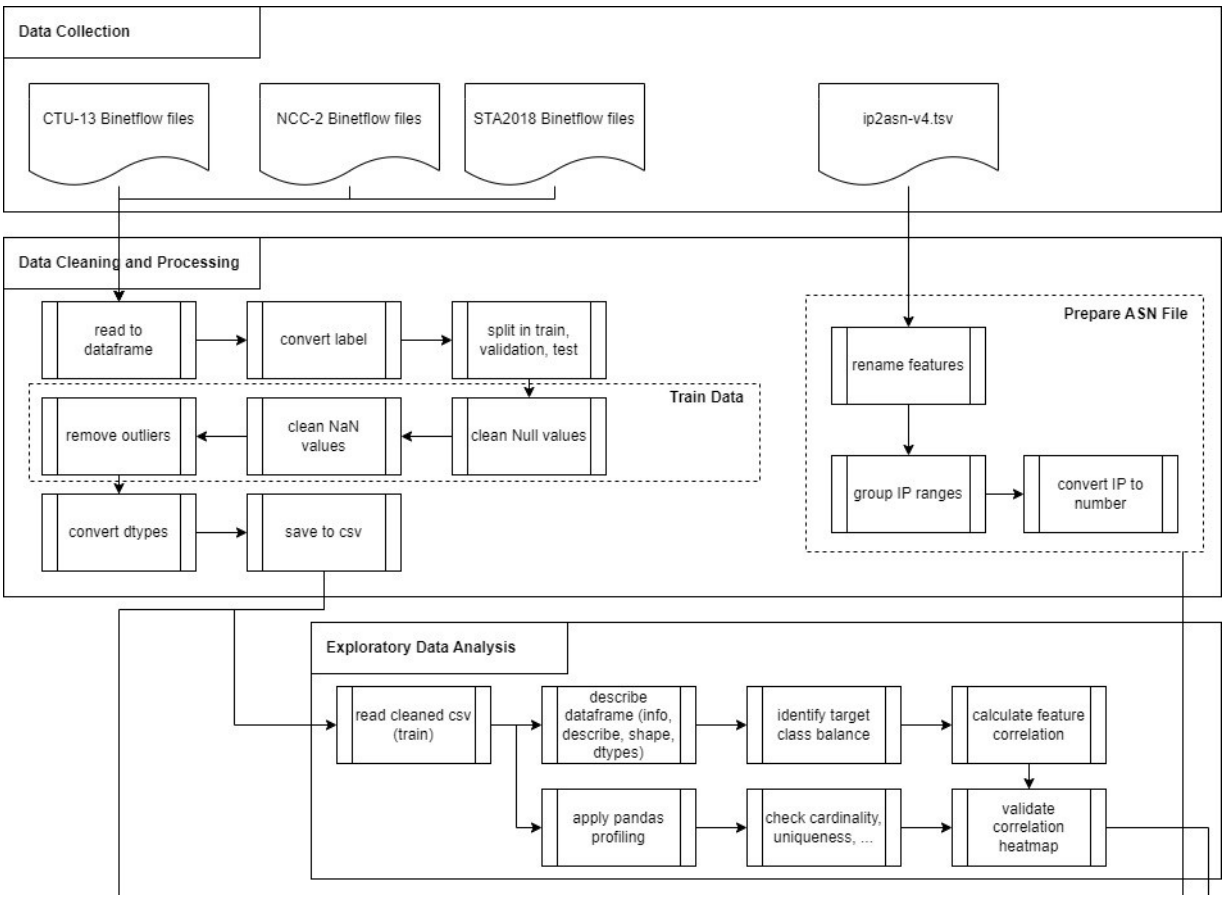


Figure 5.1: Data cleaning, processing and exploratory data analysis steps

based botnet. The class balance was identified and the datatype information of each feature was determined, including which attributes are categorical and which are numerical. All of the above steps are part of the Exploratory Data Analysis (EDA) approach used in this study. The results of the data cleaning and processing were saved in CSV format for further use in subsequent experiments. Figure 5.1 provides a high-level flow of the cleaning and processing steps performed on the collected datasets.

5.3.1. CTU-13

The CTU-13 dataset, provided by the Czech Technical University (CTU), remains valuable despite being captured in 2011, making it 11 years old. This dataset continues to hold significance as it contains a valuable mix of botnet traffic and regular benign network traffic, providing relevant information for research and analysis in the field. It is a widely used benchmark dataset in the field of cybersecurity and botnet detection, providing a standardized evaluation platform for comparing different detection techniques. Its popularity and extensive use make it a valuable reference point for researchers. The dataset contains a diverse range of botnet activities and network traffic scenarios. The dataset includes various types of botnet traffic, including DGA-based botnets, which are still prevalent in today's cybersecurity landscape. This diversity allows researchers to study and develop detection methods that can generalize to different botnet behaviors. CTU-13 contains information from 13 scenarios, where each scenario includes one or more types of cyber attacks. Sev-

eral scenarios include traffic from different botnets, which are used for this research. The information in this dataset is already labeled depending on the type of traffic, resulting in records labeled as botnet, normal, or background. Because the focus of this study is on DGA-botnets, a primary focus for CTU-13 will be given to scenarios 5 and 13, which include respectively traces of the Virut. Virut is also identified as a DGA-botnet by DGAArchive and Netlab and is included with the known domain names in their datasets. Virut operates as a time-dependent and deterministic DGA-based botnet. The algorithm used for generating domain names in Virut incorporates a changing seed, resulting in dynamically evolving domain outputs. These generated domains possess a high probability of colliding with existing or legitimate domains due to their specific format, which consists of six alphabet characters followed by the ".com" top-level domain (TLD) Kocinec [2022]. The other scenarios are dealing with other, non-DGA botnets. They will not be used in this study. An overview of the scenarios with a high-level view of the content is presented in table 5.4.

Scenario	Bot	Is DGA?	Benign and Malicious data?
1	Neris	No	Yes
2	Neris	No	Yes
3	Rbot	No	Yes
4	Rbot	No	Yes
5	**Virut**	**Yes**	**Yes**
6	Menti	No	Yes
7	Sogou	No	Yes
8	Murlo	No	Yes
9	Neris	No	Yes
10	Rbot	No	Yes
11	Rbot	No	Yes
12	NSIS.ay	No	Yes
13	**Virut**	**Yes**	**Yes**

Table 5.4: Summary of the CTU-13 dataset content per scenario.

Scenario 5 contains 129832 records with data across 15 attributes and scenario 13 contains 1925149 records across 15 attributes. These attributes, which are equal for all scenarios, are presented in Table 5.5 including a short description for each feature. In terms of the attributes of the CTU-13 dataset that are relevant for DGA botnet detection, the dataset includes a number of different types of data that may be useful. For example, the dataset includes flow-level data, which includes information about the size, duration, and direction of each network traffic flow. This data may be useful for identifying patterns or attributes that are indicative of DGA botnet activity. The CTU-13 dataset also includes packet-level data, which includes information about the content of each network packet. This data may be useful for analyzing the specific types of data that are being transmitted as part of a DGA botnet attack, and for identifying patterns or characteristics that are specific to DGA botnets.

Out of the 15 attributes, 1 attribute called 'label' identifies whether or not the record is related to malicious behavior. This label data can be used to define a target class which in turn can be used to train and evaluate machine learning models for DGA botnet detection, by providing a labeled dataset for training and testing purposes.

Attribute	Description
starttime	The timestamp when the network flow was captured.
dur	The duration of the network flow in seconds.
proto	The protocol used for the network flow (TCP, UDP, etc.).
srcaddr	The source IP address of the network flow.
sport	The source port number of the network flow.
dir	The direction of the network flow.
dstaddr	The destination IP address of the network flow.
dport	The destination port number of the network flow.
state	The state of the network flow.
stos	The type of service in the IP header of the last packet in a network flow, referring to the source.
dtos	The type of service in the IP header of the first packet in a network flow, referring to the destination.
totpkts	The total number of packets transmitted during a network flow, in both directions.
totbytes	The total number of bytes that were transmitted during a network flow in both directions.
srcbytes	The number of bytes transmitted from the source IP address to the destination IP address during a network flow.
label	**The label or ground truth for the dataset, indicating if the network flow is related to benign of malicious behavior.**

Table 5.5: Summary of the CTU-13 dataset attributes with short description.

In addition to identifying data quality issues and analyzing the high-level attributes in the dataset, EDA involves conducting a range of data analysis steps to extract descriptive statistics that reveal key attributes and characteristics of the data. These statistics encompass a variety of measures such as the mean, median, mode, standard deviation, and variance of specific attributes. The mean provides insights into the central tendency of the data, indicating its average value. The median serves as a measure of the data's central position, unaffected by extreme values. The mode represents the most frequently occurring value in the dataset. Standard deviation quantifies the degree of dispersion or spread of the data, while variance measures the average squared deviation from the mean. Collectively, these statistics offer valuable insights into the distributional properties, variability, and shape of the data, aiding in the identification of patterns, anomalies, and trends. Figure 5.2 provides a holistic view of the descriptive attribute statistics for CTU-13 - Scenario 5 and Scenario 13, created via the Pandas 'describe' function. This function provides a summary of descriptive statistics for columns in a DataFrame. It computes various statistical measures that help in understanding the distribution and properties of the data. The output includes the following statistics for each numerical column:

•Count: The number of non-null values in the column.
•Mean: The average value of the column.
•Standard Deviation: A measure of the dispersion or spread of the values.
•Minimum: The minimum value in the column.
•25th Percentile (Q1): The value below which 25% of the data falls.
•50th Percentile (Median or Q2): The middle value of the data, separating the lower and

upper halves.
•75th Percentile (Q3): The value below which 75% of the data falls.
•Maximum: The maximum value in the column.

Additionally, if the column contains non-numeric values, 'describe' provides the count, unique count, and the most frequent value of the non-numeric values. To enhance readability, all measures have been converted to integer representation.

	dur	stos	dtos	totpkts	totbytes	srcbytes
count	129832	128974	122936	129832	129832	129832
mean	77	0	0	35	28836	5082
std	282	3	0	2494	2962769	517968
min	0	0	0	1	60	0
25%	0	0	0	2	214	76
50%	0	0	0	2	266	81
75%	1	0	0	6	741	460
max	1806	192	3	694553	853704136	136546777

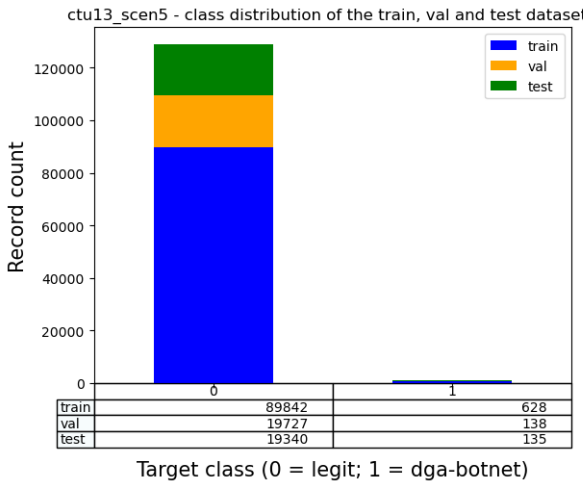
(a) CTU-13 Scenario 5

	dur	stos	dtos	totpkts	totbytes	srcbytes
count	1925149	1898070	1803339	1925149	1925149	1925149
mean	328	0	0	26	18460	3441
std	888	4	0	2049	1931164	672001
min	0	0	0	1	60	0
25%	0	0	0	2	214	77
50%	0	0	0	2	268	83
75%	2	0	0	4	600	262
max	3600	192	3	1185896	1403006471	505523769

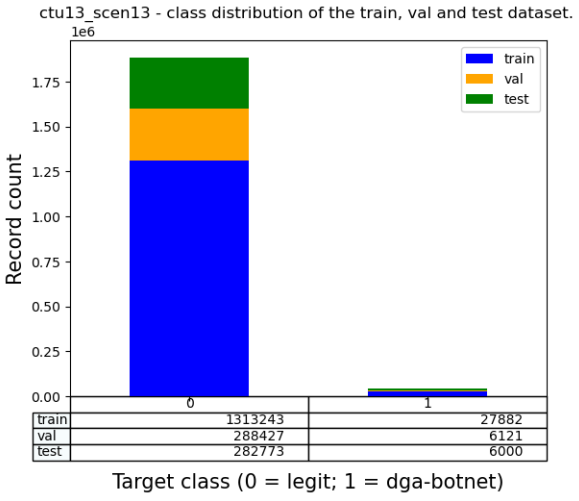
(b) CTU-13 Scenario 13

Figure 5.2: CTU-13 Scenario 5 & 13: attribute statistics

After the cleaning and processing, the class distribution for the unbalanced dataset for both scenarios was calculated as visualized in Figure 5.3, respectively representing CTU-13 scenario 5 and CTU-13 scenario 13.



(a) CTU-13 Scenario 5



(b) CTU-13 Scenario 13

Figure 5.3: CTU-13 Scenario 5 & 13: class distribution before balancing

Overall, the CTU-13 dataset has a number of attributes that makes it a well-suited dataset for research on DGA botnets. Its flow-level, packet-level, and label data provide a rich source of information for developing and evaluating methods for detecting and analyzing DGA-based botnets.

5.3.2. NCC-2

The NCC-2 dataset is a curated dataset that combines activity patterns from the CTU-13 and NCC datasets. It was created by simulating botnet attacks in July 2022 and published in September 2022. Although the simulation is based on the extracted activity patterns from the original datasets, the data itself is considered to be from 2011. Unlike the CTU-13 dataset, the NCC-2 dataset does not include detailed packet-level information but instead focuses on network header data. As a result, it may not provide specific insights into the types of data transmitted during a DGA botnet attack. However, it remains valuable for identifying patterns and characteristics related to DGA botnets at the network header level. The added value of NCC-2 is that it generates a view of simultaneous attack activity based on the CTU-13 data, resembling sporadic attacks, and NCC data which resembles periodic attacks. Because the NCC-2 dataset is based on CTU-13 and NCC, it also includes information on DGA-based botnets. This study focuses on the data related to the Virut DGA-botnet. Virut is found in all three sensors of the NCC-2 dataset. In the context of packet flow captures, a sensor refers to a network device or software component that captures and monitors network traffic. It acts as an observer and collects packets flowing through a network segment or interface. Table 5.6 provides a holistic overview of the content per sensor.

Sensor	Bot	Contains DGA?	Benign and Malicious data?
1	Rbot, Neris, Sogou, NSIS.ay, **Virut**	**Yes**	**Yes**
2	Rbot, Neris, Menti, **Virut**	**Yes**	**Yes**
3	Rbot, Neris, Murlo, NSIS.ay, **Virut**	**Yes**	**Yes**

Table 5.6: Summary of the NCC-2 dataset content per sensor.

NCC-2 includes 18 attributes available for the identification of network activity and covers a total of 14779085 records collected through three different sensors. These attributes, which are equal for all sensors in this dataset, are presented in Table 5.7 including a short description for each attribute. In terms of the attributes of the NCC-2 dataset that are relevant for DGA botnet detection, the dataset includes a number of different types of data that may be useful. For example, the dataset includes flow-level data, which includes information about the size, duration, and direction of each network traffic flow. This data may be useful for identifying patterns or attributes that are indicative of DGA botnet activity.

Due to the size of the dataset and to prevent excessive compute time, a focus was put on data coming from sensor 1, representing a total of 4895158 records. The dataset from sensor 1 represents a similar target class balance of benign versus malicious records. Looking at the descriptive statistics from sensor 1 as described in figure 5.4, it becomes clear that further processing and cleanup are needed.

Attribute	Description
starttime	The timestamp when the network flow was captured.
dur	The duration of the network flow in seconds.
proto	The protocol used for the network flow (TCP, UDP, etc.).
srcaddr	The source IP address of the network flow.
sport	The source port number of the network flow.
dir	The direction of the network flow.
dstaddr	The destination IP address of the network flow.
dport	The destination port number of the network flow.
state	The state of the network flow.
stos	The type of service in the IP header of the last packet in a network flow, referring to the source.
dtos	The type of service in the IP header of the first packet in a network flow, referring to the destination.
totpkts	The total number of packets transmitted during a network flow, in both directions.
totbytes	The total number of bytes that were transmitted during a network flow in both directions.
srcbytes	The number of bytes transmitted from the source IP address to the destination IP address during a network flow.
label	**The label or ground truth for the dataset, indicating if the network flow is related to benign of malicious behavior.**
activitylabel	A more descriptive label descriving the type of activity of the network flow.
botnetname	**The name of the botnet to which the malicious behavior is related. Used in combination with the label to identify DGA-based botnet behavior.**
sensorid	The ID of the sensor that captured the network flow.

Table 5.7: Summary of the NCC-2 dataset attributes with short description.

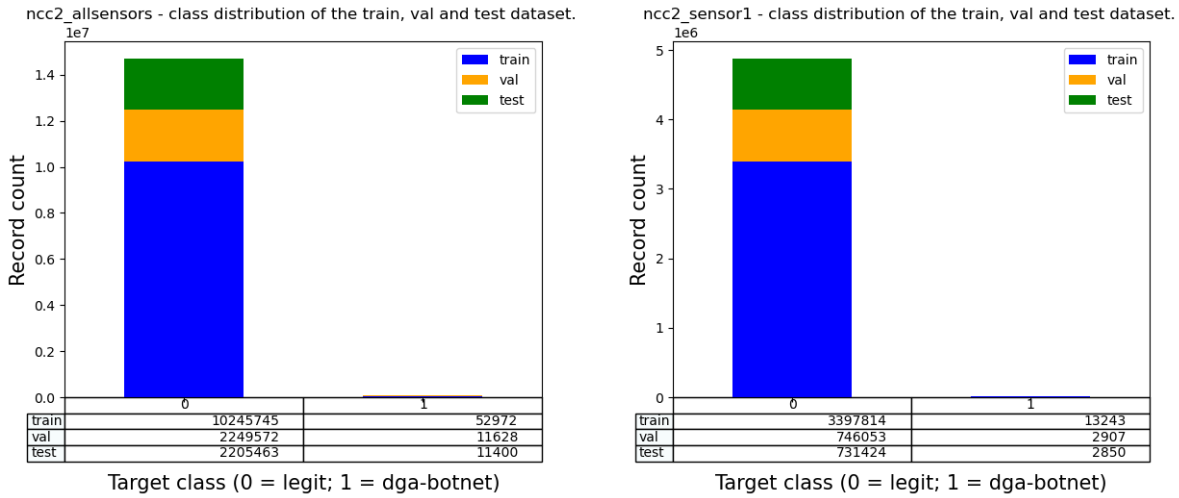
	dur	stos	dtos	totpkts	totbytes	srcbytes	activitylabel	sensorid
count	4895158	4857412	4526957	4895158	4895158	4895158	4895158	4895158
mean	345	0	0	76	60467	35323	0	1
std	902	3	0	4414	4052237	2356303	0	0
min	0	0	0	1	60	0	0	1
25%	0	0	0	2	214	78	0	1
50%	0	0	0	2	270	84	0	1
75%	6	0	0	6	797	446	0	1
max	3600	192	3	2686731	2689640464	2635366235	1	1

Figure 5.4: NCC-2 Sensor 1: attribute statistics

A thorough examination of outliers is conducted to assess their distribution within the dataset and to evaluate their potential influence on the training and evaluation of the ML models. This analysis aims to gain insights into the extent of outlier presence and understand their potential impact on the performance of the models during the training and evaluation stages. Because outliers are only identified in approximately 0.1% of the records and these can potentially indicate DGA-based botnet behavior, the decision is made to keep the outliers in the dataset. Null data is removed and attributes are converted into usable datatypes for further analysis. As highlighted before, null values and NaN values have to be removed or replaced because they can increase variability in the data or distort the distribution of data. This can impact the training and evaluation of the ML models, and negatively influence the classification of DGA botnet behavior.

Out of the 18 attributes, 1 attribute called 'label' identifies whether or not the record is related to malicious behavior. This label data can be used to define a target class which in turn can be used to train and evaluate machine learning models for DGA botnet detection, by providing a labeled dataset for training and testing purposes. In addition, another feature also indicates the botnet name related to the malicious activity.

After the cleaning and processing, the NCC-2 dataset showed a class distribution as shown in Figure 5.5 .



	0	1
train	10245745	52972
val	2249572	11628
test	2205463	11400

(a) NCC-2 - All sensors

	0	1
train	3397814	13243
val	746053	2907
test	731424	2850

(b) NCC-2 - Sensor 1

Figure 5.5: NCC-2 All Sensors & Sensor 1: class distribution before balancing

The NCC-2 dataset includes multiple attributes that make it well-suited for research on DGA botnets. Its flow-level and label data provide a rich source of information for developing and evaluating methods for detecting and analyzing DGA-based botnets.

5.3.3. STA2018

The STA2018 dataset is a rather new dataset published in 2020 by researchers at the University of St Andrews in partial fulfillment of a thesis. The dataset uses the UNB ISCX Intrusion Detection Evaluation Dataset from 2012 as a baseline and includes 1537285 records of network traces that are transformed into 193 basic attributes (derived directly from the network traces) and then extended to 550 attributes. These additional attributes are generated

by using Onut's feature classification schema Onut and Ghorbani [2007].

The data comprises network traffic of five simulation days. It is a combination of normal and malicious network activity, out of which botnet activity can be distilled. However, DGA-botnet behavior cannot easily be identified out of the regular botnet behavior as there is no identification as such available in the dataset. A high-level overview of the content is presented in table 5.8.

Sensor	Bot	Contains DGA?	Benign and Malicious data?
11 jun	Unknown	Unknown	No
12 jun	Unknown	Unknown	Yes
13 jun	Unknown	Unknown	Yes
14 jun	Unknown	Unknown	Yes
15 jun	Unknown	Unknown	Yes
16 jun	Unknown	Unknown	No
17 jun	Unknown	Unknown	Yes

Table 5.8: Summary of the STA2018 dataset content per day.

In terms of the attributes of the STA2018 dataset that are relevant for (DGA) botnet detection, the dataset includes a number of different types of data that may be useful. For example, the dataset includes flow-level data, which includes information about the size, duration, and direction of each network traffic flow. This data may be useful for identifying patterns or attributes that are indicative of DGA botnet activity. Out of all the attributes, 1 attribute called 'class' identifies whether or not the record is related to malicious behavior. This label data can be used to define a target class which in turn can be used to train and evaluate machine learning models for botnet detection, by providing a labeled dataset for training and testing purposes. A partial overview of the most relevant attributes is provided in Table 5.9. Attributes that have not been described in detail, are grouped and indicated separately in the table.

Attribute	Description
start_time	The timestamp when the network flow was captured.
duration	The duration of the network flow in seconds.
ipVersion	IP version (IPv4 or IPv6) of a connection.
protocol	The protocol used for the network flow (TCP, UDP, etc.).
src_ip	The source IP address of the network flow.
src_zone	The topological zone of the source host (GLOBAL, MULTICAST, UNICAST, UNKNOWN, LOCAL, LAN1, LAN2, LAN3, LAN4, LAN5, LAN6)
src_prt	The source port number of the network flow.
dst_ip	The destination IP address of the network flow.
dst_zone	The topological zone of the destination host (GLOBAL, MULTICAST, UNICAST, UNKNOWN, LOCAL, LAN1, LAN2, LAN3, LAN4, LAN5, LAN6)
dst_prt	The destination port number of the network flow.
conn_state	The state of the network flow.
service	The application protocol of a connection as detected.
synthetic	One (1) if synthetic connection and zero (0) otherwise.
origOrder	The original sequence number of connections and 0 for all synthetic connections
src_pkts	Total packets sent by source host.
dst_pkts	Total packets sent by destination host.
dst_ip_bytes	Total IP bytes sent by destination host.
src_ip_bytes	Total IP bytes sent by source host.
dst_bytes	Total payload bytes sent by destination host.
src_bytes	Total payload bytes sent by source host.
urg	Number of total urgent TCP packets within a connection.
bro_*	9 attributes starting with bro_, referring to attributes that are captured by Bro[14]'s Internal Engine.
conn_*	20 attributes starting with conn_, providing additional insight on the connections as computed by Bro's statistics event.
ip4_*	29 attributes starting with ip4_, providing additional insight on the IPv4 packet information of a connection.
ip6_*	7 attributes starting with ip6_, providing additional insight on the IPv6 packet information of a connection.
tcp_*, udp_*	49 attributes starting with tcp_ or udp_, providing additional insight on the TCP or UDP packet information of a connection.
DFMC_*, DFMCB_*, DFMCOS_*, DFMCOD_*, DFMCG_*, DFMT_*, DFMTB_*, DFMTOS_*, DFMTOD_*, DFMTG_*	405 attributes providing additional insight on the number of connections, number of packets and number of bytes in connection windows of 5 or 100 seconds.
class	**The connection label, indicating if the network flow is related to benign of malicious behavior.**

Table 5.9: Partial summary of the STA2018 dataset attributes with short description.

Figure 5.6 represents a subset of the descriptive statistics from STA2018. Also for this dataset, the conclusion is made that further processing and cleanup is needed.

	starttime	sport	dport	dur	ipversion	service	synthetic	origorder	src_ip_bytes	dst_ip_bytes	...	dfmtg_29	dfmtg_30	dfmtg_31
count	1537285	1537285	1537285	1537285	1537285	1537285	1537285	1537285	1537285	1537285	...	1537285	1537285	1537285
mean	1276398158	20613	1457	14	4	13250	0	98973	1955	20632	...	0	487	33
std	102687	21802	5757	254	0	18786	0	116537	624443	864854	...	0	3104	162
min	1276225266	3	0	0	4	0	0	0	0	0	...	0	0	0
25%	1276292039	2845	53	0	4	1519	0	0	48	20	...	0	0	0
50%	1276396614	4495	80	1	4	3774	0	59984	125	399	...	0	0	9
75%	1276496395	45410	80	4	4	18402	1	156064	590	2487	...	0	0	28
max	1276570866	65535	65535	86383	6	65535	1	442705	754011288	390644056	...	34	35163	3386

8 rows × 541 columns

Figure 5.6: STA2018: subset of the attribute statistics

After the cleaning and processing, the STA2018 (11 through 14 Jun) dataset showed a class distribution as shown in Figure 5.7.

Overall, the STA2018 dataset has a number of attributes that make it well-suited for research on DGA botnets, given that DGA behavior can be identified for training the model. Its flow-level and label data could provide a rich source of information for developing and evaluating methods for detecting and analyzing DGA-based botnets.

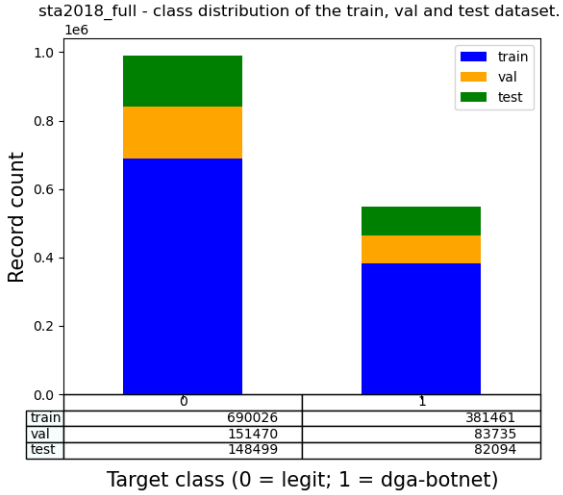


Figure 5.7: STA2018 - 11 through 14 Jun dataset: class distribution before balancing

5.3.4. MERGED DATASET

The merged dataset is derived from the CTU-13 Scenario 5, CTU-13 Scenario 13, and NCC-2 - Sensor 1 datasets. It is a combination of normal and malicious network activity, out of which botnet activity can be distilled. Because it is based on CTU-13 and NCC-2, the labeled information is available and DGA-botnet behavior is already identified. The STA2018 dataset is excluded from the merge, due to the lack of correct labeling of DGA-based botnet behavior. The merged dataset contains 6948547 before the removal of duplicates, across 19 attributes (including the target class). Other than removing duplicate records, no further

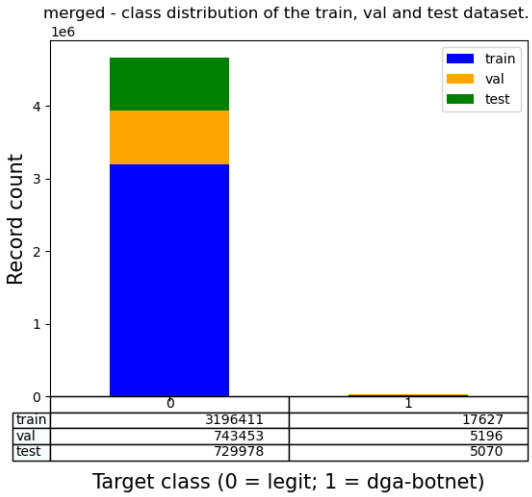


Figure 5.8: Merged dataset: class distribution before balancing

data cleaning was required as each of the individual datasets was already cleaned. Identification and removal of duplicate records happened by checking for duplicates based on the combined attributes 'dur', 'proto', 'srcaddr', 'dstaddr', 'totbytes', and 'target'. The 'target' class identifies whether or not the record is related to DGA-based botnet behavior. The attribute 'starttime' was excluded from the merging process as it had been removed from the individual datasets during cleaning. This attribute indicates the initiation of a packet flow capture and is not pertinent to the actual behavior of DGA-based botnets. The class distribution for the merged dataset is shown in Figure 5.8.

The merged dataset has a number of attributes that make it well-suited for research on DGA botnets. Its flow-level and label data are a good source of information for developing and evaluating methods for detecting and analyzing DGA botnets.

5.3.5. IP2ASN-V4

IP2ASN-v4[15] refers to a database that is used to map IP addresses (IPv4 addresses) to Autonomous System Numbers (ASNs). An Autonomous System (AS) is a collection of IP networks under the control of a single administrative entity, such as an Internet Service Provider (ISP) or a large organization. ASNs are globally unique identifiers assigned to these entities by regional Internet registries. The IP2ASN-v4 database provides a means to determine the corresponding ASN for a given IPv4 address. It enables the identification of the organization or network associated with a specific IP address, which can be useful in various applications and analyses, such as network monitoring, security investigations, traffic analysis, and routing optimizations. The database includes information such as the IP address ranges allocated to each ASN, the organization or entity associated with the ASN, and other relevant metadata. It allows users to map IP addresses to ASNs and obtain valuable insights about the ownership and routing of internet traffic.

The IP2ASN-V4 dataset contains 466798 records with data across 5 attributes. These attributes are presented in Table 5.10 including a short description for each attribute.

In this study, the IP2ASN-V4 dataset is used to map the IP addresses found in the packet flows to the actual ASN number, the country name, and the ISP name. Even though the

[15]https://iptoasn.com/

Attribute	Description
start_ip_range	The first IP of the IP-range that belongs to a particular ASN or ISP
end_ip_range	The last IP of the IP-range that belongs to a particular ASN or ISP
ASN	The ASN-number that is linked to the IP-range
country	The country of the ISP that is linked to the IP-range
isp	The ISP-name that is linked to the IP-range

Table 5.10: Summary of the IP2ASN-V4 dataset attributes with short description.

IP2ASN dataset has been retrieved in 2023 and includes the most up-to-date data at the time of retreival, it is still deemed relevant for this study. This is based on the knowledge that ASN number assignments do not change frequently. Once an organization is allocated an ASN, it typically retains that number for an extended period, often years or even decades. ASN assignments are managed by Regional Internet Registries (RIRs) such as ARIN, RIPE NCC, APNIC, LACNIC, and AFRINIC. There are reasons why an ASN assignment can change, like a merger, a network divestiture, or policy changes. Because these happen less frequently, the assumption is made that this has less of an impact and the data is still relevant. The conclusion above about the changing nature of ASN numbers is based on several sample lookups that were performed. The lookups were executed via an online tool[16], based on randomly selected ASN numbers found in the dataset. Table 5.11 shows a few results of the lookups.

ASN-number	Country	Allocated	Updated
13335	United States	Jul 14, 2010	Feb 17, 2017
29208	Czech Republic	Jul 01, 2003	Apr 03, 2023
5610	Czech Republic	Jul 22, 2002	Jan 10, 2023
38803	Australia	Oct 12, 2007	Aug 26, 2020
9808	China	Jan 10, 2000	Oct 27, 2021

Table 5.11: Summary of the ASN lookup examples performed via ipinfo.io.

5.4. Summary

This study intends to identify how DGA-based botnet detection can be improved based on packet flow information. To support this research, the usability of different dataset types is investigated as described in the previous sections. A conclusion is made that the merge of domain name datasets with datasets that contains DNS information is very difficult due to the lack of information that overlaps and can be linked.

To overcome this challenge, existing packet flow datasets are used for this study for which the data is already labeled with a truth label. The truth labels indicate whether or not the records are generated by botnets and DGA-based botnets. The packet flow datasets that form the basis for all further steps in this research are CTU-13, NCC-2, the combined dataset, and for parts of the experiments STA2018. The STA2018 dataset can be used for further analysis on botnet detection and can be included when merging datasets to increase the overall data diversity, but further research is needed to identify if DGA-based botnet

[16]https://ipinfo.io/

flows can be derived or filtered out of the network traffic. A summary of the datasets with their respective attributes is presented in table 5.12.

Dataset	Record count	Unique attributes	Records after cleaning	botnet records	DGA-botnet records
CTU-13 - Scenario 5	129.832	15	129.810	0.7%	0.7%
CTU-13 - Scenario 13	1.925.149	15	1.924.446	2%	2%
NCC-2 - Sensor 1	4.895.158	18	4.894.291	3%	0.4%
NCC-2 - All Sensors	14.779.083	18	14.776.785	5%	0.5%
STA2018	1.537.285	550	1.537.285	36%	Unknown
Merged dataset	6.948.547	19	4.697.735	Not counted	0.6%
IP2ASN-V4	468.433	5	466.798	N/A	N/A

Table 5.12: Summary of datasets and a holistic usability view for research.

Each of the datasets, with the exception of the IP2ASN-V4 dataset, contains packet flow data and includes DGA-based botnet traces. They also contain similar attributes like 'duration', 'ip addresses', 'ports', and 'protocol'. Based on these attributes, the datasets can be merged into one bigger dataset. Although the time of publication for the different datasets varies from 2011 to 2022, more important is that the actual data across all datasets are dating back from 2011 and 2012. When merging datasets, it is important to ensure that they are compatible and representative of the same context or problem domain. If datasets from different time periods are merged without considering these changes, it can lead to inaccurate or inconsistent results. In the case of studying DGA-based botnet behavior, the algorithms used by botnets can evolve and become more sophisticated over time. Therefore, using datasets from different time periods may not accurately capture the current state of DGA-based botnets. The model trained on such merged datasets may fail to detect newer and more advanced botnets, which can pose security vulnerabilities. Figure 5.9 provides a visual representation of the datasets used for this study.

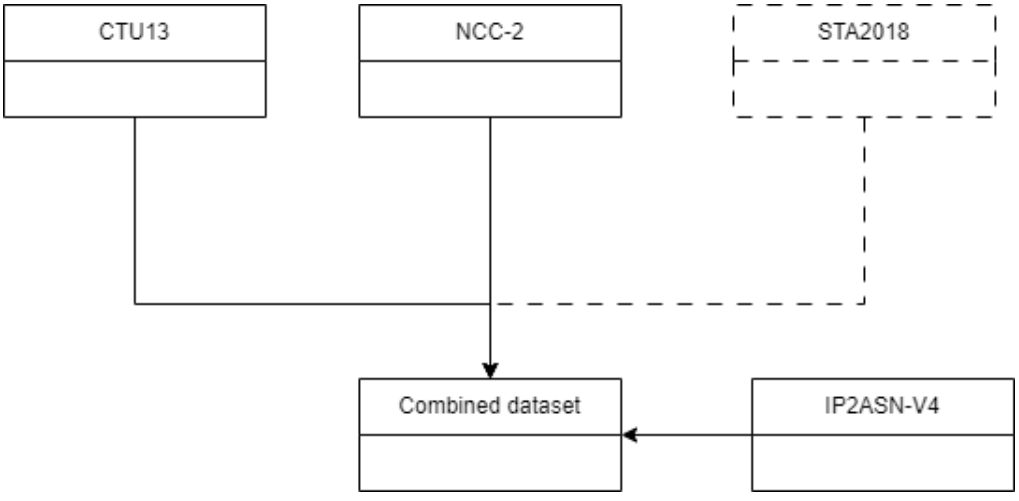


Figure 5.9: Overview of the datasets used in this study.

Several processing steps are performed on the datasets to ensure that they contain usable and accurate information. Duplicate records are removed to ensure that the data is

as clean and accurate as possible, individually on the separate datasets and again on the merged dataset afterward. In addition, a final verification is performed to verify records that have empty or incorrect data, such as null values, and incorrect classifications. Because the merged dataset is based on already cleaned data, no additional operations are needed for this dataset other than the removal of duplicate records. The datasets are normalized to get representative data across the different datasets. This is done to ensure that the final combined dataset is comprehensive and usable for ML model training.

Various techniques such as downsampling, upsampling, and applying sample weights were investigated to overcome the target class imbalances. The most successful approach involved applying sample weights, with a weight ratio of 1 to 4 for negative versus positive target classes, where a positive target class represents DGA-based botnet behavior. Regarding resampling, the dataset was upsampled using the RandomOversampler[17] from the imbalanced-learn package, while downsampling was achieved using the resample[18] method from sklearn. Dealing with highly imbalanced data is crucial to ensure that the trained models correctly classify benign and malicious network flows and that the models do not overfit or present incorrect outcomes. These processing steps are crucial in ensuring that the final datasets are accurate, clean, and representative of the data that is collected.

In summary, the datasets utilized in this study to address the subquestions pertaining to dataset availability, feature selection, and model selection include CTU-13, NCC-2, the merged dataset combining CTU-13 and NCC-2, and STA2018 for specific parts of the experiments. The merged dataset, derived from CTU-13 and NCC-2, serves as the primary dataset for addressing the final subquestion related to the training and evaluation of the machine learning models developed in this research. However, the STA2018 dataset is excluded from the merge due to its inadequate labeling of DGA-based botnet behavior.

[17] https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html
[18] https://scikit-learn.org/stable/modules/generated/sklearn.utils.resample.html

6

SELECTION OF CONTEXT-RELATED FEATURES

The data in all datasets is first cleaned and processed to make sure feature selection techniques can work with adequate data. The data cleaning and preparation are already performed and described as part of the experiments to search for usable datasets. This part of the experiment aims at finding relevant and usable features in the selected datasets. The high-level flow of the feature selection process is visualized in figure 6.1.

The focus is placed on already available features in the non-resampled datasets. This part of the experiment is mainly supported through a combination of supervised methods. The respective supervised methods that are used in support of the presented classification problem, are SelectKBest and Recursive Feature Elimination (RFE) were used. The calculation for SelectKBest was performed using the scoring function chi2. The Chi2 score measures the relevance of each feature to the target variable. The higher the chi2 score, the more relevant the feature is to the target variable. For RFE, a logistic regression model was applied. An explicit choice was made to not use the Sequential Feature Selection (SFS) throughout all experiments in this research, as initial experiments on the CTU-13 dataset pointed out that this method has a much higher computational complexity than RFE on larger datasets. From an unsupervised method perspective, correlation matrices and correlation analysis are used to check which highly correlated features can be eliminated. For the correlation calculation, the Pearson method is used, which resembles a correlation mea-

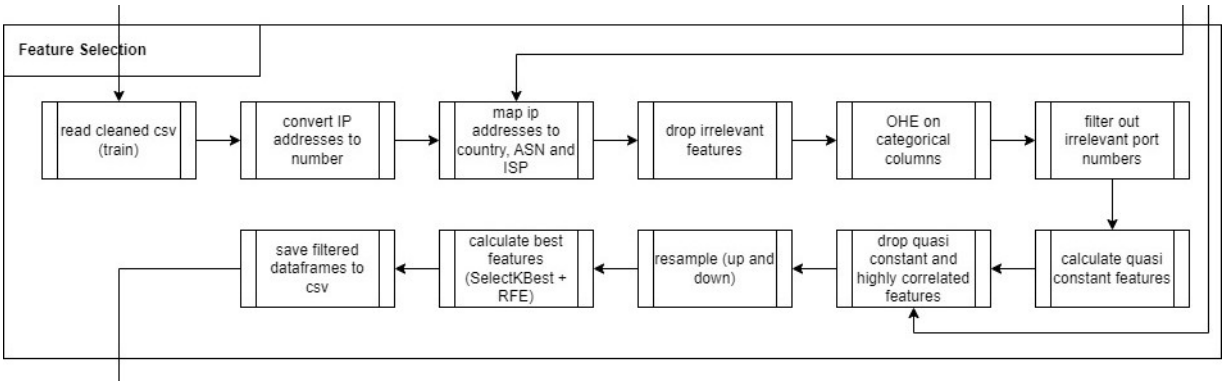


Figure 6.1: Feature selection process steps

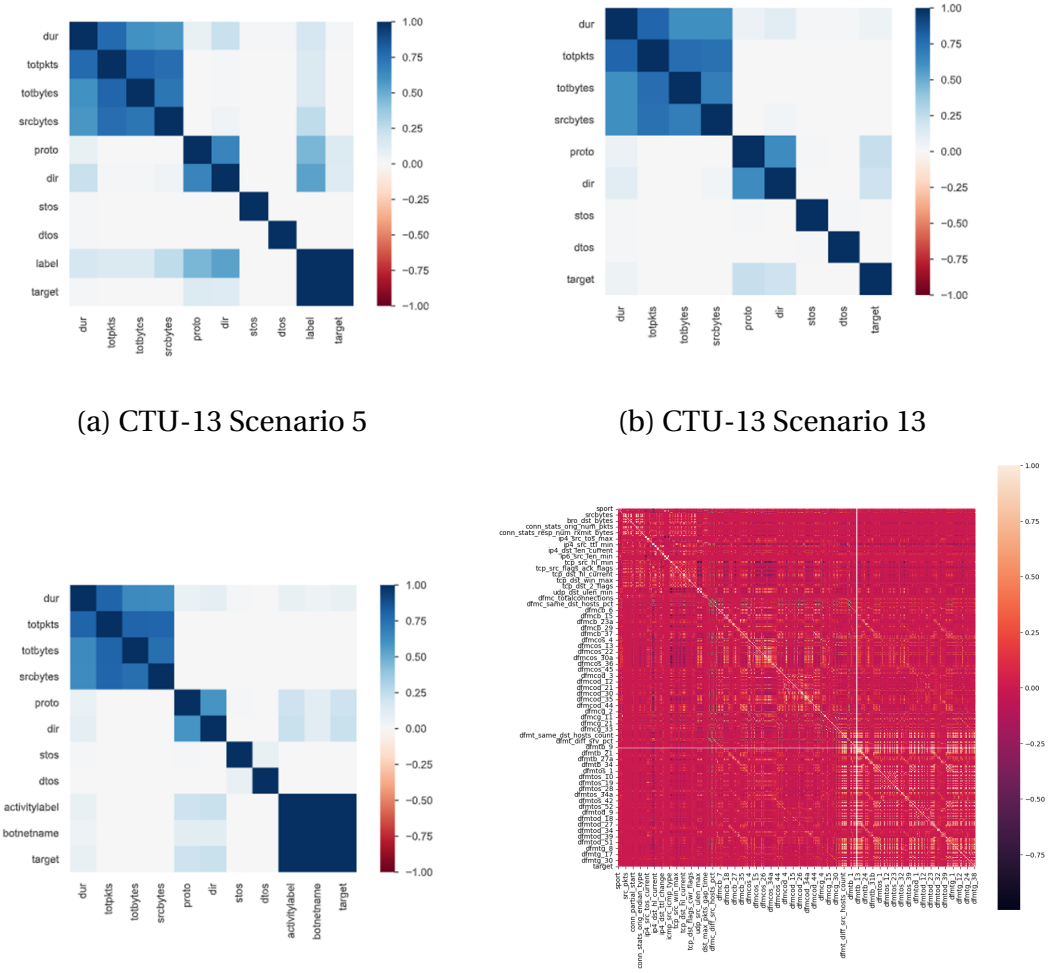
sure based on a linear relationship between two features and highlights to what extent the features are related to each other.

By defining a variance threshold to identify almost constant features and performing a feature correlation analysis, features can be identified as being of less importance. This means, being of less importance within the current feature set available in the dataset. The variance threshold verification is configured to use a threshold value of 0.01, which means that some features could still pass the test although they have many constant values. A higher value can be used, which would result in even fewer features being passed to the models. However, we want to avoid stripping out too many features and leave some decision logic to the ML model during training. Because the data already went through data cleaning and processing, the categorical features are one hot encoded. For example, this results in a feature vector of 4303 features for the CTU-13 Scenario 13 dataset out of which 4284 are identified as being almost constant Examples hereof are features like 'state__SPA', 'dir_ who', and 'new_country_AD'. In the case of the NCC-2 dataset, 7009 out of 7029 features are found to be almost constant. Thus, the above-mentioned technique help us in removing many less relevant features from the feature vector.

The feature correlation matrices of respectively CTU-13 - Scenario 5, CTU-13 - Scenario 13, NCC-2 Sensor 1, and STA2018 are shown in Figure 6.2. Due to the big feature vector of the STA2018 dataset, the full correlation matrix is shown as a reference but not readable in its totality. The feature correlation matrices provide information about the relationships between the different features through a calculated correlation coefficient. The correlation coefficient measures the strength and direction of the linear relationship between the features. The correlation coefficient can range from -1 to +1, where -1 indicates a strong negative correlation, +1 indicates a strong positive correlation, and 0 indicates no correlation. By examining the correlation matrices, the strongly correlated features can be identified, meaning they tend to vary together. This information is useful for feature selection and dimensionality reduction. Highly correlated features may provide redundant information, and keeping only one of them can simplify the model and improve its interpretability. On the other hand, features with low or no correlation may offer unique insights and contribute independently to the model's predictive power. Visualization of the feature correlation matrices for the different datasets indicates that the features totpkts and totbytes are highly correlated. Because of this very high correlation, the dimensional complexity of the dataset can be reduced by keeping only one, which will make training of the selected models in further experiments faster. The same applies to the features 'label' and 'target', which are both representing the ground truth of the data.

Based on the literature review, the feature 'starttime' has been removed from all datasets. It is an identifier related to the moment when the packet flow capture started and is not related to the actual DGA-based botnet behavior. Also dataset-specific features like 'sensorid', 'activitylabel', and 'botnetname' are removed because they are only used to label the data and build up the target class, without further use during feature selection of ML model training.

During the analysis, RFE is performed on different feature sets and the features 'state_XXX', 'new_country_XXX', and 'new_ASN_XXX' are often identfied as relevant features. Here, XXX represents a specific state, country, or Autonomous System Number (ASN) abbreviation. These states, countries, and ASN numbers are derived from one hot encoding the original features 'state', 'country', and 'ASN'. It is worth noting that states refer to the state of a net-



(a) CTU-13 Scenario 5

(b) CTU-13 Scenario 13

(c) NCC-2 Sensor 1

(d) STA2018

Figure 6.2: Feature correlation matrices of CTU-13 Scenario 5, CTU-13 Scenario 13, NCC-2 Sensor 1 & STA2018

work connection and that ASN is a unique identifier that is assigned to a network operated by a single organization. In addition to RFE, also the SelectKBest feature selection method is used to identify the most relevant features in some datasets. This method demonstrates that for some datasets, the most relevant features are derived from both 'country' and 'ASN'. Additionally, the features 'ports' and 'src_bytes' are also important to consider in certain cases. A sequential feature selector (SFS) is only used on CTU-13 - scenario 5 and CTU-13 - scenario 13. It yields a slightly different result, where the most important features are indicated as 'dur', 'totpkts', and 'ports'. As highlighted earlier, given the computational time complexity on larger datasets, this feature selection technique is not further used during the experiment and is left out of the summary table.

In summary, different feature selection techniques are utilized, including RFE and SelectKBest, to identify the most relevant features in the datasets. The results show that features related to 'state', 'country', 'ASN', 'ports', and 'src_bytes' are frequently returned as relevant. SFS is also considered, but given its computational time complexity, it is only used on a limited number of datasets.

Each feature set to further use during experiments contains between 18 and 26 columns depending on the dataset from which they are derived, due to the one-hot encoded nature of these features. From the original features, only four are used. They are extended by the country and ASN information derived from the destination IP address. The four original features are 'dur', 'srcbytes', 'dport', and 'state'. The details of the top 3 most relevant features as a result of this experiment are captured in table 6.1.

Dataset	Technique	Preprocessed + OHE Features (top 3)	Derived from (original features)
CTU-13 - Scenario 5	RFE	dir_ ->, new_country_DE, new_country_None	dir, country
	SelectKBest	ports, src_bytes, new_country_DE	dport, src_bytes, country
CTU-13 - Scenario 13	RFE	state_INT, state_SRPA_FSPA, new_ASN_36351	state, ASN
	SelectKBest	ports, src_bytes, new_ASN_36351	dport, src_bytes, ASN
NCC-2 - Sensor 1	RFE	state_FSA_FSA, state_SRPA_FSPA, new_country_CZ	state, country
	SelectKBest	ports, srcbytes, new_country_US	dport, srcbytes, country
STA2018	RFE	state_SH, new_country_None, new_ASN_611	state, country, ASN
	SelectKBest	ports, srcbytes, dur	ports, srcbytes, dur
Merged dataset	RFE	state_INT, state_S_, new_ASN_43037	state, ASN
	SelectKBest	ports, src_bytes, state_S_	dport, src_bytes, state

Table 6.1: Summary of most relevant features based on RFE and SelectKBest techniques.

7

SELECTION OF ML METHODS

Classifying datasets is a fundamental task in machine learning, and there are many algorithms that can be used to accomplish this. These algorithms can be broadly categorized into two groups: baseline and NN-based classifiers. Baseline classifiers include a wide range of algorithms such as Decision Trees, Naive Bayes, and k-Nearest Neighbors. On the other hand, neural networks are modeled after the human brain, making them particularly suited to tasks such as image and speech recognition. In addition, ensemble methods are strategies employed to enhance model accuracy by aggregating multiple models instead of relying on a single model. These methods leverage the collective insights of diverse models, enabling them to make more accurate predictions and achieve better performance. By combining the predictions of individual models, ensemble methods can effectively reduce errors, increase robustness, and capture a broader range of patterns and relationships in the data.

In this study, initial experiments are performed using three classifiers that have performed well in previous studies: Support Vector Machines (SVM), Random Forest Classifier (RFC), and Gradient Boosted Trees (in specific XGB) van Renswou [2021]. Additionally, Putra et al. [2022] focused their research primarily on the classifiers Decision Tree, Random Forest, Logistic Regression, kNN, and Naïve Bayes. To limit the scope of this research, no further focus was given to Naïve Bayes and Logistic Regression. Most classifiers from both studies have been extensively tested and have proven to be effective in many different applications. SVM is particularly useful for datasets that are not linearly separable, while RFC is known for its ability to handle large datasets and GBT is a powerful algorithm that can achieve high accuracy on many datasets.

The derived feature sets from earlier steps of the experiments, which were discussed in the previous section, are utilized to enhance the performance of the classical classifiers. These feature sets involve a subset of the features that are most relevant to the classification task and derived from the context of the earlier obtained datasets.

In addition to the highlighted classifiers above, further experiments were performed on less-used algorithms like bagging, kNN, Ada Boost, and Histogram Gradient Boost. Even though less used in the field of play, these are all machine learning techniques that are commonly used for classification and prediction tasks and might yield good results with specific context-related feature sets. kNN and Histogram Gradient Boosting are both good candidates when working with large-scale datasets and many features, while Ada Boost has

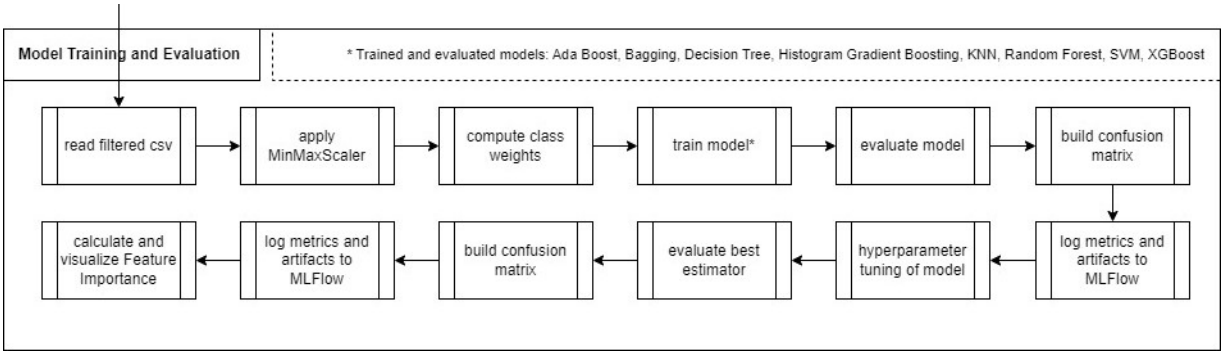


Figure 7.1: Model training and evaluation process steps

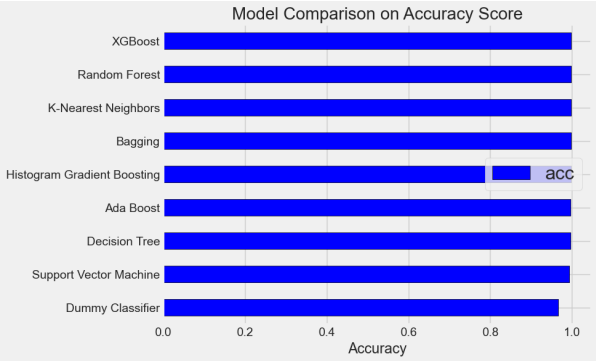
a better focus on unbalanced data and Bagging (in this research using a Decision Tree Classifier) in general can help to reduce overfitting by reducing the variance of its predictions. These classifiers are all tested in the experiments to identify if good model performance could be obtained based on the context-selective features that are used, equal to earlier proven methodologies. A graphical overview of the model training and evaluation process is presented in figure 7.1.

A summary of all the models that were tested during the experiments can be found in table 7.1.

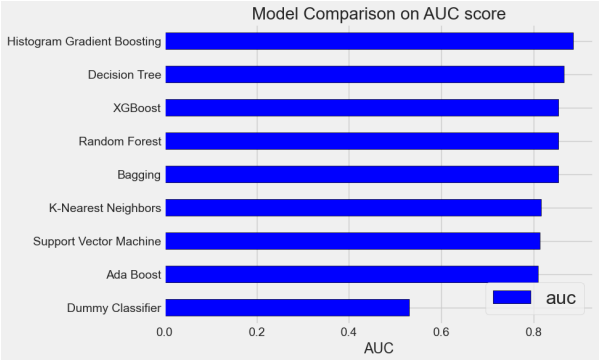
Model	Type
Ada Boost	Ensemble - Boosting (can be any type of classifier)
Bagging	Ensemble (can be any type of classifier)
Decision Trees	Baseline - Decision Trees
Histogram Gradient Boosting	Ensemble - Boosting (based on Decision Trees)
kNN	Baseline - Nearest Neighbors
Random Forest	Ensemble (based on Decision Trees)
Support Vector Machine	Baseline - SVM
XGBoost	Ensemble - Boosting (based on Gradient Boosting)

Table 7.1: Summary of tested models.

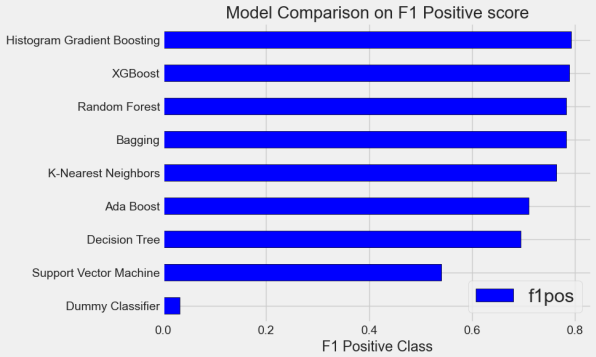
From a programming perspective, the implementation was done in Python due to the many available libraries that have implemented different classifiers. In particular, the scikit-learn library was used for evaluating the different algorithms. This library is widely used in the machine learning community and has a wide range of classifiers and utilities that can be used to evaluate the performance of the algorithms. Additionally, it supports all the earlier-mentioned classifiers, making it easy to evaluate the different algorithms. To make a first selection of models, each model was trained and evaluated against feature sets of the CTU13 dataset (respectively scenarios 5 and 13), without further hyperparameter tuning. The dataset was also not resampled, but a sample weight of 1 (negative class) compared to 4 (positive class) was provided. The actually calculated sample weights for the classes were 0.5 for the negative class against 63 for the positive class, but experiments indicated that the 1 compared to 4 weights gave better results for the models. The decision to use datasets that are not resampled was made based on experiments on both the CTU-13 Scenario 5 and CTU-13 Scenario 13. In these experiments, higher performance results were



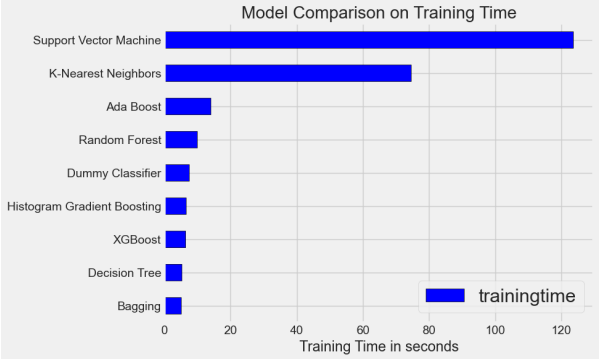
(a) Model comparison by accuracy



(b) Model comparison by AUC



(a) Model comparison by F1 Pos



(b) Model comparison by compute time

Figure 7.2: CTU-13 Scenario 5: Model performance comparison

obtained in runs when using sample weights compared to the runs where downsampled or upsampled datasets were used. From the initial runs as indicated above, the accuracy score, AUC score, F1 Positive score and compute time were calculated to perform an initial assessment. The comparison of the above-mentioned scores is respectively represented in figure 7.2.

On accuracy scores, XGBoost, Random Forest, and kNN come out as the top 3 models. For AUC scores, this is respectively Histogram Gradient Boosting, Decision Tree, and XGBoost. Taking the F1 Positive score into consideration, Histogram Gradient Boosting, XGBoost, and Random Forest come out as best performing models. Finally, looking at computing time, Bagging, Decision Tree, and XGBoost come out as best performing. On the other spectrum, looking at the least performing models, SVM scores lowest on almost all criteria and has the highest compute time. Based on this initial analysis, a preliminary decision was made to continue further experiments with the models XGBoost, Histogram Gradient Boosting, Random Forest, and Bagging. The main driver for this decision is the best performance on F1 Positive Score combined with a high AUC score and the low compute time that the models require. The results of this experiment against the CTU-13 dataset are provided in table 7.2.

To validate the obtained outcomes of the first experiment on models, the results and selection of the models to use have been compared with the outcome of another run of all models, now using the NCC-2 Sensor 1 dataset. Because the intention was to find the best suitable ML algorithms, the models were trained again with the NCC2 Sensor 1 dataset. The

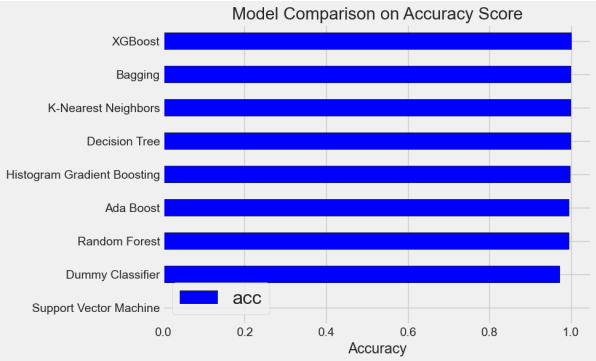
Model	Accuracy	AUC	F1 Pos score	Training Time (seconds)
Histogram Gradient Boosting	0.997	0.886	0.792	6.9s
XGBoost	0.998	0.854	0.789	6.8s
Random Forest	0.997	0.854	0.783	10.6s
Bagging	0.997	0.854	0.783	5.4s
kNN	0.997	0.816	0.763	78s
Ada Boost	0.997	0.81	0.71	14.6s
Decision Tree	0.996	0.866	0.695	5.5s
Support Vector Machine	0.993	0.814	0.541	126s

Table 7.2: CTU13 Scenario 5: Performance of trained models without hyperparameter tuning, sorted by F1 Pos score in descending order.

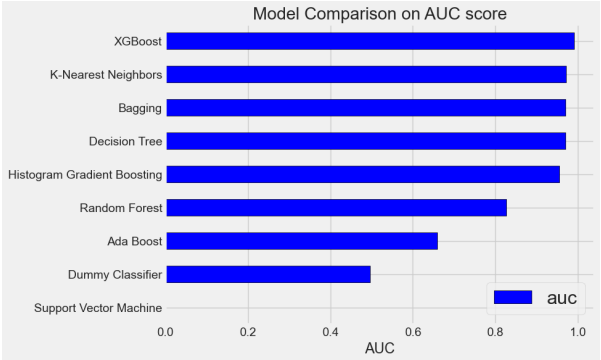
same sample weights to train the models were not changed, meaning the same weight of 1 against a weight of 4 is used for respectively the negative and positive target classes. In this run, 1912437 records were used to train the models. From the validation runs, where a bigger dataset was used, the impact and performance of different ML models were very clear. Although kNN came out better in AUC and F1 Pos scores compared to the previous run and even compared to some other models, a decision was made to keep it out of scope for further analysis. The main driver for the decision was the computing time, where the actual training and evaluation time did exponentially explode. k-NN is a non-parametric algorithm that requires comparing a new data point with all existing data points in the training set to determine the K nearest neighbors. This big increase in training time makes the models less usable and hence also supports the earlier provided decision on ML models. Also for the SVM model, the significant increase in training time, resulting from the dataset's size and the decision boundary's complexity due to the numerous features, led to the exclusion of the model for subsequent experiments. This decision was reinforced by the validation run on the NCC2 Sensor 1 dataset, which had to be terminated after continuously running for more than 24 hours. For Decision Tree, a major performance improvement was seen, and was therefore also added to the selection of models to further explore. The results of this experiment that ran against the NCC2 Sensor 1 dataset can be found in table 7.3. A graphical overview of the results is also visualized in figure 7.3, in which the values of SVM have been set to 0 manually as this model was excluded from the experiment.

Model	Accuracy	AUC	F1 Pos score	Training Time (seconds)
XGBoost	0.99968	0.99028	0.97356	74.10s
Bagging	0.99935	0.97021	0.94546	53.64s
kNN	0.99929	0.97158	0.94127	17919.02s
Decision Tree	0.99929	0.97018	0.94072	13.55s
Histogram Gradient Boosting	0.99671	0.95515	0.76871	18.94s
Random Forest	0.99336	0.82569	0.54148	151.81s
Ada Boost	0.99387	0.65915	0.38443	287.58s

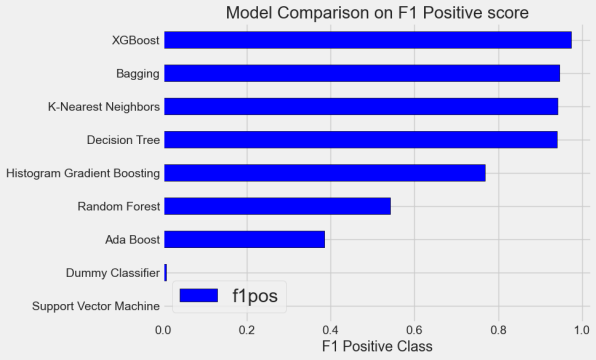
Table 7.3: NCC2 Sensor 1: Performance of trained models without hyperparameter tuning, sorted by F1 Pos score in descending order.



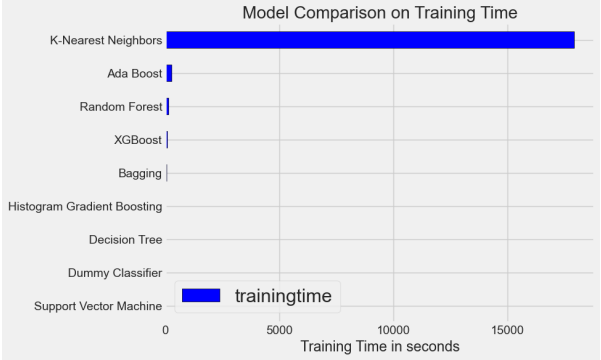
(a) Model comparison by accuracy



(b) Model comparison by AUC



(a) Model comparison by F1 Pos



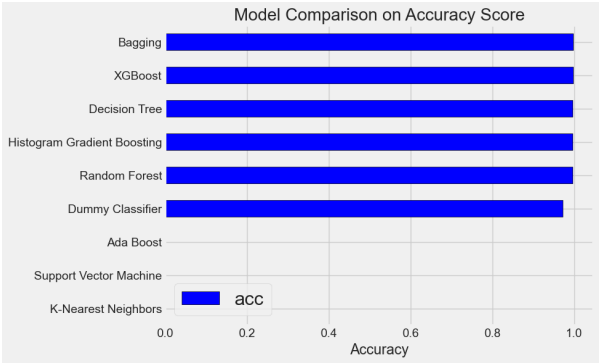
(b) Model comparison by compute time

Figure 7.3: NCC2 Sensor 1: Model performance comparison

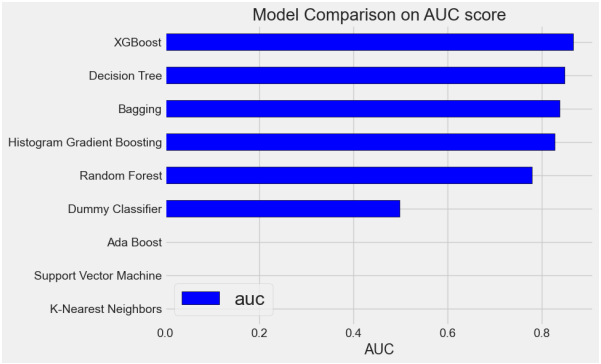
In the third round of this experiment, the same process is applied to the combined dataset generated out of the CTU-13 and NCC-2 datasets. Also in this step, the models were separately trained and evaluated to identify which models perform best. To obtain a fair point for comparison, again sample weights are used to score the models during training. In this run, 3214038 records were used to train the models. Based on the outcomes of the runs on CTU-13 and NCC-2 data, the models AdaBoost, kNN, and SVM are no longer considered. On this dataset, Bagging, XGBoost, and Decision Tree come out as the top three best-performing models when considering the F1 Pos score combined with the AUC score. Also for the combined dataset, a Decision Tree Classifier was used for Bagging. The results of this experiment can be found in table 7.4. Figure 7.4 provides a visual representation of these results, in which the values of Ada Boost, kNN, and SVM have been set to 0 manually as these models were excluded from the experiment.

Model	Accuracy	AUC	F1 Pos score	Training Time (seconds)
Bagging	0.99727	0.83895	0.77566	146.91s
XGBoost	0.99699	0.86671	0.77228	158.02s
Decision Tree	0.99664	0.84819	0.74282	29.55s
Histogram Gradient Boosting	0.99654	0.82740	0.72464	46.19s
Random Forest	0.99629	0.77893	0.67669	322.21s

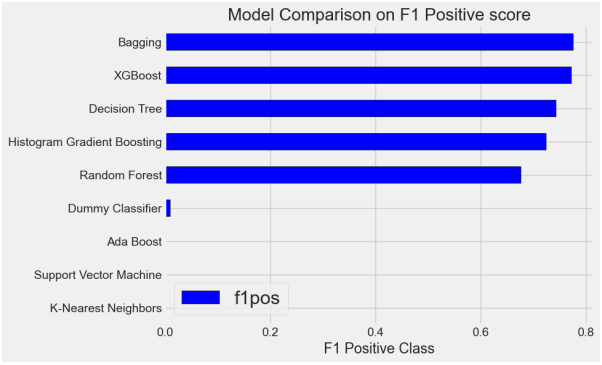
Table 7.4: Merged dataset - CTU-13 & NCC-2: Performance of trained models without hyperparameter tuning, sorted by F1 Pos score in descending order.



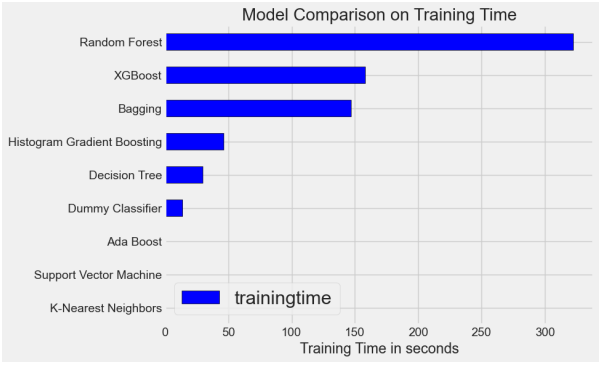
(a) Model comparison by accuracy



(b) Model comparison by AUC



(a) Model comparison by F1 Pos



(b) Model comparison by compute time

Figure 7.4: Merged dataset - CTU-13 & NCC-2: Model performance comparison

It is clear that there is a big difference between the obtained accuracy scores and the visibly lower F1 Positive scores. Accuracy is a straightforward metric that calculates the ratio of correctly classified instances to the total number of instances. It provides a general overview of the model's overall correctness. However, accuracy alone is not a reliable measure when dealing with imbalanced datasets, where one class significantly outweighs the other. In such cases, a high accuracy can be achieved by simply predicting the majority class most of the time, while the minority class remains misclassified. On the other hand, the F1 score considers both precision and recall, which are calculated based on true positive, false positive, and false negative predictions. Precision represents the proportion of correctly classified positive instances out of all instances predicted as positive, while recall measures the proportion of correctly classified positive instances out of all actual positive instances. The F1 score combines these two measures into a single value, which provides a balanced evaluation of the model's performance, particularly in cases where there is an imbalance between the classes. Because the models are trained based on highly imbalanced datasets, the accuracy and F1 scores differ significantly. The model's ability to correctly classify positive instances differs from its ability to correctly classify negative instances. The F1 score places more emphasis on the model's performance in correctly identifying positive instances, making it a more suitable metric in the given scenarios of this research, where the detection of positive cases is of higher importance.

All of the hereby mentioned experiments have been performed on ML models without considering hyperparameter tuning. The actual tuning and effect on the performance of the ML models is discussed in the next section which deals with improving the performance of the selected models and how these results can be positioned against related work.

8

PERFORMANCE OF SELECTED MODELS AND POSITIONING AGAINST RELATED WORK

The previous experiment had a focus on selecting ML models that are best fit-for-purpose in the context of this research. The experiments are continued based on the outcomes of the model selection, formulating an answer on the performance of the classifiers Bagging, Decision Tree, and XGBoost.
A short overview of the top three selected models is summarized again in Table 8.1, ranked on their respective F1 Pos-Score. These scores represent the model performance before hyperparameter tuning.

Data	Model	Accuracy	AUC	F1 Pos-score (DGA)
Merged dataset	Bagging	0.99727	0.83895	0.77566
Merged dataset	XGB	0.99699	0.86671	0.77228
Merged dataset	Decision Tree	0.99664	0.84819	0.74282

Table 8.1: Top three results ranked by F1 Pos-score, obtained during model selection, before hyperparameter tuning.

In an attempt to increase and assess the performance of the selected models, hyperparameter tuning is performed. Hyperparameter tuning is an important step in improving the performance of a machine-learning model. The process involves defining the range of hyperparameters, selecting a search method, recursively evaluating the model on a validation set, and selecting the best set of hyperparameters. For this study, the explicit choice is made to use grid search for the tuning steps. One advantage of grid search is that it is guaranteed to find the best set of hyperparameters within the predefined range, given enough computational resources. This immediately also highlights the potential challenge of grid search, as it can be computationally expensive and time-consuming, especially when dealing with a large number of hyperparameters or a large range of values for each hyperparameter. As the number of hyperparameters and the range of values increase, the search space can become exponentially large, leading to an explosion in the number of models that need to be trained and evaluated. Alternatives to the grid search, are for example random search or Bayesian optimization. These methods are however not further explored during this research to limit the scope of the study.

To validate the outcomes from different experiments during hyperparameter tuning, a cross-validation approach was employed Cucchiarelli et al. [2020]. The cross-validation is done as part of the grid search execution. In grid search, the k-fold cross-validation technique is used. K-fold cross-validation[1] involves dividing the dataset into k equal-sized subsets or folds. The grid search algorithm then trains and evaluates the model k times, each time using a different fold as the validation set and the remaining folds as the training set. The performance of the model is averaged over the k iterations to provide an overall evaluation. This helps to assess the model's generalization performance and find the optimal hyperparameters for the given grid of parameter values. This method was used to compare the efficiency of the models based on the earlier described performance metrics and helped to reduce the risk of overfitting. This is in contrast with the initial training of the models, for which a cross-validation approach was not used. Instead, a choice was made to use a fixed dataset split between train, test, and validation sets for a few reasons. Firstly, the fixed split allows for consistency in evaluating model performance. By keeping the same data points in the validation set throughout the training process, we can compare the model's performance on the validation set across different iterations or experiments. Secondly, a fixed split can be more efficient when working with limited computational resources or time constraints. Cross-validation involves repeating the training and evaluation process multiple times, which can be computationally expensive, especially for larger datasets or complex models. With a fixed split, we can train the model once on the training set, evaluate its performance on the validation set, and then use the independent test set for the final evaluation.

Once the hyperparameters are tuned and the final models are trained, it is important to assess their performance. Assessing the performance of the tuned models is done with unseen data, in this particular case on a test set that was earlier derived from the merged dataset. After tuning, XGBoost obtains the best performance increase with an end result in F1 Pos-score of 79.28%, AUC-score of 88.15%, and accuracy score of 99.72%. The Bagging classifier also shows an increased performance after tuning but with a smaller performance gain. Only for the Decision Tree, the performance score reduces when looking at the F1 Pos-score. Table 8.2 summarizes the performance of the earlier selected models after hyperparameter tuning. Important to note that the results after hyperparameter tuning are based on an evaluation dataset with previously unseen data. This makes us conclude that the baseline Decision Tree classifier likely demonstrates overfitting and that it learned the training data too well, including noise and irrelevant patterns, which leads to poor generalization on unseen data. In contrast, methods like XGBoost and Bagging employ regularization techniques such as shrinkage and tree pruning, which help prevent overfitting and improve generalization performance. Another reason can be the existence of non-linear relationships in the data. Decision trees are limited in their ability to capture complex non-linear relationships between features and the target variable. XGBoost for example, being an ensemble of decision trees, can effectively model non-linear relationships by combining multiple weak learners to form a strong learner. This enables XGBoost to capture more intricate patterns and achieve better performance on the validation set.

In a previous study by van Renswou [2021], a Random Forest Classifier achieved an accuracy score of 99.63%, followed by a Gradient Boost model with an accuracy of 97.40%, and an SVM with 82.42% accuracy. The Random Forest Classifier achieved the best re-

[1] https://scikit-learn.org/stable/modules/cross_validation.html

Data	Model	Accuracy	AUC	F1 Pos-score (DGA)
Merged dataset	XGB	0.99724	0.88148	0.79279
Merged dataset	Bagging	0.99738	0.84610	0.78497
Merged dataset	Decision Tree	0.99471	0.85719	0.65209

Table 8.2: Top three results ranked by F1 Pos-score, obtained after hyperparameter tuning.

sult when trained with an oversampled dataset. However, the selected models, after hyperparameter tuning, performed equally well or even slightly better than the models in Renswou's study based solely on the accuracy metric. There are no other performance metrics used in Renswou's study that can be used to compare results with. Renswou used a Matthews Correction Coefficient (MCC-score), while in this study an AUC-score and F1-score is used. Although MCC is linked to an F1-score, MCC considers all four scores of the confusion matrix, while in this study only F1 for the positive class is considered for scoring due to the highly imbalanced dataset. An important note to highlight when comparing machine learning model performance between studies is that the performance metrics and evaluation methodologies used may vary. It is crucial to consider the specific metrics used, the dataset characteristics, the preprocessing techniques applied, and the experimental setup. Therefore, direct comparisons between different studies should be made cautiously, taking into account these factors to ensure a fair and meaningful comparison.

Another study by Putra et al. [2022] used similar datasets and ML models, such as Decision Trees, Random Forest, Logistic Regression, k-NN, and Naïve Bayes. They claimed to have achieved a detection accuracy of 99.998% for the CTU-13 dataset and 99.999% for the NCC dataset. However, they only described an average accuracy score and did not provide details about individual models. Moreover, their study evaluated models on each dataset separately, without training or evaluating on a combined dataset. Therefore, a detailed comparison of their results with the results from this research is not possible based solely on the average accuracy score.

9

CONCLUSIONS - DISCUSSION, VALIDITY, FUTURE RESEARCH, REFLECTION

9.1. ANSWERS TO RESEARCH QUESTIONS

To produce clear answers and recommendations that will benefit this research and potential future studies, several sub-questions are addressed to provide specific and tangible outcomes.

SRQ1: WHICH DATASETS ARE SUITABLE TO DERIVE CONTEXT-RELATED FEATURES FROM PACKET FLOW INFORMATION?

In chapter 5.3, three datasets are described that are identified as relevant and suitable data sources for this study. These three datasets were created by peer researchers and used in this research are:

1. CTU-13
2. NCC-2
3. STA2018

Further experiments on these datasets are performed and described in section 6.1. The combined datasets consist of 18.371.351 records that are used in this research, which consist of both benign and malicious network flow data. Out of the full datasets, only an average of 0.6% is related to DGA-based botnet behavior. Unfortunately, the data is relatively old considering the domain of DGA-based botnet detection, and data is not unique across the different datasets. Based on the above-mentioned datasets, a merged dataset is also generated as part of this study. It combines data from CTU-13 (both scenarios) and NCC-2 - Sensor 1. The STA2018 dataset is excluded from the merge, due to the lack of correct labeling of DGA-based botnet behavior. During experimentation, duplicate records are discarded in the original datasets and the merged dataset. The merged dataset resulted in 6.948.547 records, out of which 4.697.735 unique records are obtained for usage in further experiments. The merged dataset also represents 0.6% DGA-based botnet data related specifically to Virut.

SRQ2: WHICH CONTEXT-RELATED FEATURES CAN BE DERIVED FROM PACKET FLOW DATASETS TO IMPROVE THE OVERALL PERFORMANCE AND ACCURACY OF DGA BOTNET DETECTION?
The context-relativity for feature selection refers to selecting features specifically in the particular domain of DGA-based botnet detection. In the context of DGA-based botnet detection, context-related features are selected and designed based on the characteristics of DGA traffic and network data. To support that context-relativity, a combination of supervised feature selection techniques is used to obtain relevant features based on the available input data. Section 6.2 described the selection of context-related features in more detail. In summary, four main features are derived from the datasets, being 'dur', 'srcbytes', 'dport', and 'state'. The feature set with these four features is further extended by adding two features related to the country and ASN information derived from the destination IP address. Table 9.1 provides a quick overview of these features with their respective description.

Feature	Description
dur	The duration of the network flow in seconds.
srcbytes	The number of bytes transmitted from the source IP address to the destination IP address during a network flow.
dport	The destination port number of the network flow.
state	The state of the network flow.
country	The country of the ISP provider related to the destination IP address.
ASN	The unique network identifier provided to the ISP provider related to the destination IP address.

Table 9.1: Summary of most relevant features derived from the datasets based on the context of DGA- based botnet detection.

When considering DGA-based botnet detection, the highlighted features can exhibit specific characteristics related to DGA activities. They are general network traffic features that can be used for various types of network traffic analysis, including but not limited to DGA detection. However, this fits the purpose of this study is to identify how these features can improve DGA-based botnet detection. The breakdown of how the above-mentioned features may be relevant:

• Dur (Duration): The duration of a network connection can provide insights into DGA activities. DGA-generated domain names often exhibit short-lived connections or short durations, as they are frequently changing and dynamically generated. Monitoring the duration of network connections can help identify suspicious or anomalous behavior associated with DGA-generated domains.

• Srcbytes (Source Bytes): Srcbytes represent the number of bytes sent by the source in a network connection. In the case of DGA, analyzing the srcbytes feature can reveal patterns or abnormalities in the size of data transmitted to or from DGA-generated domains. Unusual variations or significantly different srcbytes values may indicate DGA-related activities.

• Dport (Destination Port): The destination port in network traffic data can provide insights into the specific services or protocols being targeted by DGA activities. DGA may attempt to communicate with specific ports associated with known services or exploit vulnerabilities in certain protocols. Analyzing the destination port can help detect suspicious traffic patterns associated with DGA.

• State: The state feature refers to the state of a network connection (e.g., established, closed, etc.). DGA-generated domains may exhibit specific connection states that differ from normal traffic patterns. Monitoring the state feature can help identify abnormal connection states that may be indicative of DGA activities.

• Country: Analyzing the country associated with the network traffic can provide geographical insights into DGA activities. DGA campaigns often involve malicious actors distributing their activities across various countries to evade detection. Detecting connections to DGA-generated domains originating from multiple or unexpected countries can help identify potential DGA-related traffic.

• ASN: The ASN represents a unique identifier associated with an Autonomous System (AS) in the internet routing infrastructure. Analyzing the ASN of network traffic can help identify patterns or clusters of DGA activities originating from specific ASNs or network providers. Unusual or unexpected ASN values associated with DGA-generated domains can be indicative of malicious traffic.

SRQ3: Which ML methods can be used to improve DGA botnet detection when using context-related features derived from packet flow datasets?

Section 6.3 provides insight into the experiments held to identify which ML techniques can be used to improve DGA botnet detection. Eight different classifiers are investigated based on multiple datasets and derived feature sets. Following the outcomes of several experiments, Bagging, XGBoost, and Decision Tree are identified to be the best-performing ML methods for the current scope and context of DGA-based botnet detection. kNN provided better results during some of the experiments but was not further considered due to the compute time required for training the model. SVM was also quickly kept out of scope, due to the combination of a long training time and less-than-optimal results derived from the initial runs.

The best results are obtained when using an imbalanced dataset, but applying sample weights during the training and evaluation of the ML model. For the merged dataset, the Bagging classifier can get an accuracy of 99.73%, an AUC-score of 83.90%, and an F1-score for the positive class of 77.57%. XGBoost scores marginally lower with an F1-score of 77.23% and Decision Tree demonstrates an F1-score of 74.28%. The emphasis is put on the F1-score rather than the accuracy score, as this measure provides more meaningful results given the class imbalance and the aim to classify the actual DGA-based botnet behavior in the dataset.

SRQ4: How do the proposed models perform and how can we position the outcomes of these models against results from earlier work?

Hyperparameter tuning is used to further improve the performance of the trained ML models. Hyperparameter tuning involves defining hyperparameter ranges, selecting a search method, and recursively evaluating models on a validation set to choose the best hyperparameters. Grid search is used in this study as it guarantees to find the best hyperparameters, although it can become computationally expensive if the feature sets are not selected carefully. The tuned models are evaluated based on an unseen test set derived from the same dataset as earlier experiments. As a result of the hyperparameter tuning, most of the performance metrics can be increased for the selected models. For the Bagging classifier, results are obtained that demonstrate an accuracy of 99.74%, an AUC-score of 84.61%, and an F1-score for the positive class of 78.50%. For XGBoost, the accuracy score is improved with a

marginal 0.1%, but the F1 Pos-score with 2%, resulting in 79.28%. On AUC-score, XGBoost now demonstrates a score of 88.15% and on accuracy a score of 99.72% Only for the Decision Tree method, both the accuracy score and F1 Pos-score decrease. The accuracy score reduces by 0.2%, but the F1 Pos-score reduces by almost 9%. This is likely caused by overfitting or a limited search space of the model during training, while unseen data is introduced for evaluation and the model starts classifying the unseen data incorrectly. Overfitting or a limited search space can have a big impact on classifier models like Decision Trees. In summary, depending on the used dataset and feature selection in the provided setup, XGBoost is the best-performing model.

An attempt has been made to position the results of this research against the outcomes of van Renswou [2021] and Putra et al. [2022]. The first study achieved high accuracy scores with Random Forest Classifier, Gradient Boost, and SVM models. The performance indicators respectively demonstrate an accuracy score of 99.63%, 97.40%, and 82.42%. Based on only the accuracy metric, a conclusion can be made that the presented models perform equally well or even slightly better. The second study claimed to have achieved even higher accuracy scores with similar models but did not provide details on individual models or evaluate the models on a combined dataset. Therefore, a detailed comparison of results between these studies is difficult due to differences in datasets, feature engineering techniques, and performance metrics. In general, comparisons of results between different studies must be done with the utmost care and caution.

9.2. Contribution

In this study, we investigate if the DGA-based botnet detection can be improved by applying context-related feature selection and training traditional ML models based on these feature sets. The contribution lies in the usage of traditional ML models trained by data derived from packet flow datasets, where the data is labeled as DGA-botnet related or not. In addition and a more specific contribution, we use merged datasets to increase the data diversity during training and evaluation. As part of the research, we identified the feature importance of the selected features and how the usage of these features contributes to more accurate detection of DGA-based botnet behavior. We perform this analysis by training multiple models and selecting the highest-performing models. The concept of highest-performing models, in combination with the feature selection and merged datasets, is to my knowledge not performed in a similar way in other studies. In this study, the primary performance metrics used are F1-score, AUC-score, and Accuracy. Most studies use the accuracy score as the leading indicator, while this research uses F1-score as the leading indicator. The models obtained the best performance metrics when using sample weights to score the training and evaluate the models. Most other related work use either downsampled or upsampled datasets.

9.3. Discussion and threats to validity

This study highlights the possibility of machine learning techniques to improve the speed, effectiveness, and accuracy of detecting DGA-based botnets. During the experiments, only a small subset of features is derived from packet flow datasets. It is important to note that machine learning techniques are not foolproof and can be susceptible to evasion tactics by sophisticated botnets. Therefore, it is necessary to continuously improve and update

these techniques to keep up with evolving botnet strategies. Furthermore, the use of machine learning algorithms for detection purposes may also introduce potential biases and limitations. The effectiveness of machine learning algorithms depends on the quality and quantity of training data used to develop the model. If the training data is limited or biased, the resulting model may not accurately generalize to new or different environments. In addition, several ML models like SVM have been disregarded after the initial experiments due to the time complexity during training. Although models like SVM might have an exponential training time, the detection accuracy of these models could potentially be higher than the selected models in this study.

Another significant risk involves the potential of outdated literature, as new information is frequently published, and the field of study is continuously progressing. This created difficulties in validating the research outcomes and conducting a positioning against the newly acquired information. To mitigate this risk, the study primarily focused on papers published within the last three to five years, although literature from the last ten to fifteen years was also considered. However, limiting the amount of reviewed literature and the number of validated results presented another potential risk. This risk suggests that the baseline for future comparisons could be based on a limited dataset. To manage this risk, known scientific databases were used in conjunction with the Mendeley reference management tool to maintain a comprehensive overview of the literature used in the study.

The literature and presented outcomes were extensively validated to ensure the legitimacy and accuracy of the input used for the research. This included the literature, outcomes, and datasets used. However, in the case of DGA-based botnet detection and prevention, it is challenging to establish a golden standard for comparison due to the field's rapid evolution. Consequently, much earlier research may not apply in the current context with new datasets. On the other hand, the continuous evolution of the domain results in an ongoing stream of new research papers published each year, which can help with the first level of validation - quantitative research. To increase the likelihood of finding correct and legitimate information, the extensive literature study confirmed information in multiple, non-related sources. Related work for validation was derived from the analysis of feature selection and choice of machine learning (ML) techniques used. However, different usage of feature sets or ML methodologies can result in different related works to validate against.

The potential risk of datasets becoming outdated or unavailable was present, which can be problematic when working in a fast-changing field. Outdated or irrelevant datasets can affect the research's accuracy and relevance. However, the availability and quality of the data were already extensively covered during the literature study and dataset search. In addition to the quantitative literature research used for the first level of validation, qualitative validation was performed on the obtained datasets. This involved a combination of techniques such as building confusion matrices and checks against overfitting. Confusion matrices were used, with the rows representing actual target classes and the columns representing predicted classes from applied machine learning (ML) models. To validate against overfitting, where the ML model models the training set too well, metrics such as accuracy, AUC, and F1 score were calculated and monitored to identify and prevent/correct overfitting.

The CTU-13 dataset, and on limited occurrences the NCC-2 dataset, are widely used in research on botnet and DGA-botnet detection, but there are potential threats to the validity of the results obtained from these datasets. Generalizability is one threat, as the CTU-13

dataset is collected from a single organization and may not be representative of other networks or organizations. To address this, the NCC-2 and STA2018 datasets were included in the experiments. Another threat is data quality, as real-world network traffic may contain errors or inconsistencies that can lead to inaccurate results if not accounted for. Data relevance is a third threat, as the datasets are relatively old and may not reflect current network security or threats faced by organizations. To mitigate this, the more recent STA2018 dataset was added to the experiments. Class imbalance is a fourth threat, as both datasets are heavily skewed towards benign traffic, which can make it challenging to train models and obtain accurate results. This is especially problematic when studying malicious traffic, as the majority of the data is benign. To address this, methods such as downsampling, upsampling, and applying sample weights were explored. The best results were obtained by applying sample weights with a weight of 1 to 4 for negative versus positive target classes, where a positive target class indicates DGA-based botnet behavior. Although the datasets are valuable resources for DGA-botnet detection research, it is essential to recognize the potential threats to validity and consider them when interpreting the results.

One final point to consider regarding the datasets is the utilization of specific features derived from the network packets. For example, the destination port is a fixed port that a server listens on to receive a connection request from a client. This port number can be particular to a specific botnet and therefore significant to detect it. However, it is also easily modifiable and probably varies for different types of botnets. The use of destination ports can create issues when attempting to detect novel botnets. While it may be relevant for a particular dataset, it will likely differ in other contexts, rendering its usefulness questionable. Consequently, it may be beneficial to exclude certain features from the feature vector in new research to develop an algorithm that can more effectively identify botnets in new environments.

Part of the research focused on positioning the outcomes against those of earlier work. As the literature review was conducted using academic research, the qualitative outcomes of related work were assumed to be valid. Reproducing or validating earlier work would have been too time-consuming and beyond the scope of this research. Instead, the validation of this research involved positioning the outcomes and conducting small experiments on smaller datasets to determine if similar results could be achieved.

In addition to the qualitative and quantitative approaches, this work also included an exploratory research layer. While the qualitative research had already included some experiments to validate outcomes, the focus of the exploratory research was to provide accurate and legitimate answers to the research questions posed in this work. To conduct these experiments, Jupyter Notebooks[1] were used. Since Jupyter Notebooks are widely used in the data analysis and machine learning domains, the validity of the tools themselves was assumed. Furthermore, many packages, libraries, and ML methods already exist for these tools, which could be used as starting points for the experiments.

9.4. FUTURE WORK

This study demonstrates that DGA-based botnet detection can be improved through adaptive feature selection based on packet flow information. At the same time, it also introduces several areas of research that could be explored in the future to improve the accuracy and

[1] Jupyter Notebook: https://jupyter.org/

effectiveness of these methods even further.

In this study, we contribute to the research domain by merging multiple datasets and using the merged dataset for feature selection and model training. The used datasets are quite old, which raises the question of whether the network packets contained within the datasets are still useful. To ensure the effectiveness of newly developed algorithms for classifying new data, it is recommended that fresh datasets are captured or constructed. Additionally, we can look at scaling to larger datasets. As the size of network traffic datasets continues to grow, future work could focus on developing machine learning models that can efficiently scale to handle larger datasets without sacrificing accuracy. Finally, the currently used datasets only contain information about a limited set of DGA families. Datasets could be searched or created that include information from multiple DGA families.

The selection and engineering of features are critical in developing accurate machine-learning models. Future work could focus on identifying new features that may be more effective in detecting DGAs and botnets. In this study, we kept the focus on a limited set of features based on the feature selection techniques RFE and SelectKBest. Contextual information based on the IP addresses was added to the datasets. For the feature selection process, additional feature selection techniques could be explored or other thresholds could be explored to obtain different feature vectors.

In this study, an initial selection of 8 machine learning methods has been chosen. This selection is done based on literature studies to identify methods that have been used before, forming a baseline for comparison. Future work could extend the research by exploring other, newer, and less used machine learning or deep learning methods. From a different angle, although hyperparameter tuning is performed as part of this study, more research could go into the actual hyperparameter tuning to identify how detection effectiveness can be improved. Considering the outcomes achieved through grid search, the utilization of Bayesian optimization, as an alternative optimization technique, becomes relevant. This choice is warranted when faced with potential overfitting, constraints imposed by the limited search space, or computational limitations.

9.5. REFLECTION

Careful planning was essential to achieve the expected outcomes and meet the target deadlines. One major planning challenge was the amount of time I spent on certain tasks, which had an impact on the overall timeline of the study. There have been two main areas that influenced the duration of the research.

The first area is the literature review. The amount of literature available on the topic of DGA-based botnet detection is quite extensive and has many variations that can lead in different directions. This made it difficult to manage the amount of time spent on the literature review and to stay on track with the overall timeline of the research. To mitigate this challenge, I had to set clear boundaries for the literature review, and a clear but strict threshold had to be defined for how many results I would validate my results against, a limited amount of papers selected during the literature review.

Linked to that first area, the second area is the validation of results from earlier work. This could have been a very time-consuming activity that even leads to deviation from the expected outcomes if not managed properly. To minimize this risk, I limited my research phase to datasets that are already used in other research and that have proven to be trustworthy and reliable. During the pre-work of this research, several datasets have already

been obtained from earlier work and reviewed on their usefulness.

A crucial note to make, is the decision to shift from the initial approach of merging DGA and DNS datasets to using packet flow datasets. It has proven to be a significant turning point in my research on DGA-based botnet detection. This change in approach has provided several notable benefits and addressed several challenges that were encountered during the earlier stages of the project.

One of the primary advantages of using packet flow datasets is the availability of more comprehensive and detailed information about network traffic patterns. These datasets encompass a wealth of network protocol headers, including IP, TCP, UDP, ICMP, and more. This rich data allows for a more thorough analysis of DGA-based botnet behavior, enabling me to identify distinct patterns and characteristics associated with malicious activities.

By working with packet flow datasets, I have also been able to overcome the challenges related to merging DGA and DNS datasets. The previous approach involved handling missing, incomplete, or non-matchable data, which posed significant difficulties and limited the reliability of the merged dataset. The transition to packet flow datasets eliminates these challenges, providing a more reliable and consistent source of data for my research.

In retrospect, the decision to shift to packet flow datasets has proven to be a pivotal step in my research. It has provided a solid foundation for my analysis of DGA-based botnet detection. The availability of detailed network traffic data, coupled with context-related feature selection techniques, has empowered me to develop a more accurate and efficient detection approach. The insights gained from this transition have not only improved the quality of my research but also expanded my understanding of DGA-based botnet behavior in real-world network environments.

Another reflection is on the time and resources required to run the ML algorithms. As the datasets used in this research are relatively large, it did take a significant amount of time and resources to calculate results and to make alterations to the model for validating and improving results. I tried to minimize this risk by spending plenty of time on data preprocessing and the selection of ML methods to use. Additionally, it is important to note that only 1 DGA-botnet in CTU-13 (virut) is available which is contradictory to the expected outcome for the datasets, which might limit this research.

I would also like to emphasize the support and help I received from Harald Vranken and Clara Maathuis. The constructive periodic discussions proved helpful in keeping me motivated and focused on continuing. Without their regular feedback, it would have been significantly more challenging to adhere to the planned course of action.

BIBLIOGRAPHY

Saif Al-Mashhadi, Mohammed Anbar, Shankar Karuppayah, and Ahmed K. Al-Ani. A review of botnet detection approaches based on dns traffic analysis. *Lecture Notes in Networks and Systems*, 67:305–321, 2019. ISSN 23673389. doi: 10.1007/978-981-13-6031-2_21/COVER. URL https://link.springer.com/chapter/10.1007/978-981-13-6031-2_21. 22

Suad Alasadi. Review of data preprocessing techniques in data mining. *Journal of Engineering and Applied Sciences*, 12:4102–4107, 09 2017. URL https://www.researchgate.net/publication/320161439. 30

Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. Building a dynamic reputation system for dns. In *19th USENIX Security Symposium (USENIX Security 10)*, pages 273–290, Washington, DC, 09 2010. USENIX Association. URL https://www.usenix.org/conference/usenixsecurity10/building-dynamic-reputation-system-dns. 19, 24

Martin Beneš. Botnet detection based on network traffic classification. Master's thesis, Masaryk University, 2015. URL https://is.muni.cz/th/hoh4l/masters.pdf. 1

Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. A comparative analysis of xgboost. *ArXiv*, abs/1911.01914, 2019. URL https://www.researchgate.net/publication/337048557_A_Comparative_Analysis_of_XGBoost. 14

Daiki Chiba, Takeshi Yagi, Mitsuaki Akiyama, Toshiki Shibahara, Tatsuya Mori, and Shigeki Goto. Domainprofiler: toward accurate and early discovery of domain names abused in future. *International Journal of Information Security*, 17:661–680, 2017. ISSN 16155270. doi: 10.1007/s10207-017-0396-7. 18

Nicholas Cox and Kelvyn Jones. Exploratory data analysis. *Quantitative Geography, London: Routledge*, pages 135–143, 01 1981. 30

Alessandro Cucchiarelli, Christian Morbidoni, Luca Spalazzi, and Marco Baldi. Algorithmically generated malicious domain names detection based on n-grams features. *Expert Systems with Applications*, 170:114551, 12 2020. doi: 10.1016/j.eswa.2020.114551. 17, 37, 67

Lieven Desmet, Jan Spooren, Thomas Vissers, Peter Janssen, and Wouter Joosen. Premadoma: An operational solution to prevent malicious domain name registrations in the .eu tld. *Digital Threats: Research and Practice*, 2(1):1–24, 1 2021. ISSN 2692-1626. doi: https://doi.org/10.1145/3419476. URL https://lirias.kuleuven.be/3613850. 2, 19, 24, 25

Salim Dridi. Supervised learning - a systematic literature review, Apr 2022. URL osf.io/tysr4. 9, 10, 11

Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2005.10.010. URL https://www.sciencedirect.com/science/article/pii/S016786550500303X. ROC Analysis in Pattern Recognition. 33

Rosa L. Figueroa, Qing Zeng-Treitler, Sasikiran Kandula, and Long H. Ngo. Predicting sample size required for classification performance. *BMC Medical Informatics and Decision Making*, 12(1):8, Feb 2012. ISSN 1472-6947. doi: 10.1186/1472-6947-12-8. URL https://doi.org/10.1186/1472-6947-12-8. 29

Shuang Hao, Alex Kantchelian, Brad Miller, Vern Paxson, and Nick Feamster. Predator: Proactive recognition and elimination of domain abuse at time-of-registration. pages 1568–1579, 10 2016. doi: 10.1145/2976749.2978317. 18, 19, 24

Kate Highnam, Domenico Puzio, Song Luo, and Nicholas R. Jennings. Real-time detection of dictionary dga network traffic using deep learning. *SN Compututer Science*, 2:110, 2021. doi: 10.1007/s42979-021-00507-w. URL https://doi.org/10.1007/s42979-021-00507-w. 17

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf. 13

Sheharbano Khattak, Naurin Rasheed Ramay, Kamran Riaz Khan, Affan A. Syed, and Syed Ali Khayam. A taxonomy of botnet behavior, detection, and defense. *IEEE Communications Surveys Tutorials*, 16(2):898–924, 2014. doi: 10.1109/SURV.2013.091213.00134. 1, 4

Patrik Kocinec. Classification and blocking of dga domain names. Master's thesis, Masarykova univerzita, 2022. URL https://is.muni.cz/th/fnnl4/. 42

Joewie J. Koh and Barton Rhodes. Inline detection of domain generation algorithms with context-sensitive word embeddings. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2966–2971, 2018. doi: 10.1109/BigData.2018.8622066. 18

Charles Kozierok. *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*. No Starch Press, USA, 2005. ISBN 159327047X. 39

Ludmila Kuncheva, Marina Skurichina, and Robert Duin. An experimental study on diversity for bagging and boosting with linear classi ers. *Information Fusion*, 3:245–258, 12 2002. doi: 10.1016/S1566-2535(02)00093-3. 15

Jonghoon Kwon, Jehyun Lee, Heejo Lee, and Adrian Perrig. Psybog: A scalable botnet detection method for large-scale dns traffic. *Computer Networks*, 97:48–73, 3 2016. ISSN 13891286. doi: 10.1016/J.COMNET.2015.12.008. 22, 23

Niklas Kühl, Marc Goutier, Robin Hirt, and Gerhard Satzger. Machine learning in artificial intelligence: Towards a common understanding. In *Hawaii International Conference on*

System Sciences, 03 2019. URL https://arxiv.org/ftp/arxiv/papers/2004/2004.04686.pdf. 8

Niklas Kühl, Max Schemmer, Marc Goutier, and Gerhard Satzger. Artificial intelligence and machine learning. *Electronic Markets*, 32, 11 2022. doi: 10.1007/s12525-022-00598-0. 8

Wanting Li, Jian Jin, and Jong-Hyouk Lee. Analysis of botnet domain names for iot cybersecurity. *IEEE Access*, 7:94658–94665, 2019. doi: 10.1109/ACCESS.2019.2927355. 18

Jianbing Liang, Shuang Zhao, and Shuhui Chen. A protocol-independent botnet detection method using flow similarity. *Security and Communication Networks*, 2022:1–14, 7 2022. ISSN 1939-0122. doi: 10.1155/2022/3161143. URL https://www.hindawi.com/journals/scn/2022/3161143/. 22, 23

Hieu Mac, Duc Tran, Van Tong, Linh Giang Nguyen, and Hai Anh Tran. Dga botnet detection using supervised learning methods. In *Proceedings of the 8th International Symposium on Information and Communication Technology*, SoICT '17, page 211–218, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450353281. doi: 10.1145/3155133.3155166. URL https://doi.org/10.1145/3155133.3155166. 18

Batta Mahesh. Machine learning algorithms -a review. *International Journal of Science and Research (IJSR)*, 01 2019. ISSN 2319-7064. doi: 10.21275/ART20203995. 9, 11

Ahmed M. Manasrah, Thair Khdour, and Raeda Freehat. Dga-based botnets detection using dns traffic mining. *Journal of King Saud University - Computer and Information Sciences*, 34:2045–2061, 5 2022. ISSN 1319-1578. doi: 10.1016/J.JKSUCI.2022.03.001. 20

Jian Mao, Jiemin Zhang, Zhi Tang, and Zhiling Gu. Dns anti-attack machine learning model for dga domain name detection. *Physical Communication*, 40:101069, 03 2020. doi: 10.1016/j.phycom.2020.101069. 17

Cláudio Marques, Silvestre Malta, and João Paulo Magalhães. Dns dataset for malicious domains detection. *Data in Brief*, 38:107342, 2021. ISSN 2352-3409. doi: https://doi.org/10.1016/j.dib.2021.107342. URL https://www.sciencedirect.com/science/article/pii/S2352340921006260. 37

Juhong Namgung, Siwoon Son, and Yang-Sae Moon. Efficient deep learning models for dga domain detection. *Security and Communication Networks*, 2021:1–15, 01 2021. doi: 10.1155/2021/8887881. 17

Iosif-Viorel Onut and Ali A. Ghorbani. A feature classification scheme for network intrusion detection. *International Journal of Network Security*, 5, 01 2007. doi: 10.1109/CNSR.2006.53. 48

Alina Oprea, Zhou Li, Ting Fang Yen, Sang H. Chin, and Sumayah Alrwais. Detection of early-stage enterprise infection by mining large-scale log data. *Proceedings of the International Conference on Dependable Systems and Networks*, 2015-September:45–56, 9 2015. doi: 10.1109/DSN.2015.14. 20

Mayana Pereira, Shaun Coleman, Bin Yu, Martine DeCock, and Anderson Nascimento. Dictionary extraction and detection of algorithmically generated domain names in passive dns traffic. In Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis, editors, *Research in Attacks, Intrusions, and Defenses*, pages 295–314, Cham, 2018. Springer International Publishing. ISBN 978-3-030-00470-5. 18

Giorgio Piras, Maura Pintor, Luca Demetrio, and Battista Biggio. Explaining machine learning dga detectors from dns traffic data. 8 2022. doi: 10.48550/arXiv.2208.05285. URL https://deepai.org/publication/explaining-machine-learning-dga-detectors-from-dns-traffic-data. 22

Daniel Plohmann, Khaled Yakdan, Michael Klatt, Johannes Bader, and Elmar Gerhards-Padilla. A comprehensive measurement study of domain generating malware. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, page 263–278, USA, 2016. USENIX Association. ISBN 9781931971324. 6, 7

Tho Poon. Botnet detection using recurrent neural networks. Master's thesis, Open Universiteit, 2018. 23, 24

Phillip Porras. Inside risks - reflections on conficker. *Commun. ACM*, 52:23–24, 10 2009. ISSN 0001-0782. doi: 10.1145/1562764.1562777. URL https://doi.org/10.1145/1562764.1562777. 1

M. Praveena and V. Jaiganesh. A literature review on supervised machine learning algorithms and boosting process. *International Journal of Computer Applications*, 169:32–35, 2017. ISSN 0975-8887. URL https://www.academia.edu/35159133/A_Literature_Review_on_Supervised_Machine_Learning_Algorithms_and_Boosting_Process. 8, 10, 13

Muhammad Aidiel Rachman Putra, Tohari Ahmad, and Dandy Pramana Hostiadi. Analysis of botnet attack communication pattern behavior on computer networks. *International Journal of Intelligent Engineering and Systems*, 15, 8 2022. doi: 10.22266/IJIES2022.0831.48. 9, 21, 24, 25, 59, 68, 72

Fangli Ren, Zhengwei Jiang, Xuren Wang, and Jian Liu. A dga domain names detection modeling method based on integrating an attention mechanism and deep neural network. *Cybersecurity*, 3:4, 02 2020. doi: 10.1186/s42400-020-00046-6. 17

Raaghavi Sivaguru, Jonathan Peck, Femi Olumofin, Anderson Nascimento, and Martine De Cock. Inline detection of dga domains using side information. *IEEE Access*, 8:141910–141922, 3 2020. ISSN 21693536. doi: 10.1109/ACCESS.2020.3013494. URL https://arxiv.org/abs/2003.05703v1. 39

Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2009.03.002. URL https://www.sciencedirect.com/science/article/pii/S0306457309000259. 32

Ali Soleymani and Fatemeh Arabgol. A novel approach for detecting dga-based botnets in dns queries using machine learning techniques. *Journal of Computer Networks and Communications*, 2021, 2021. ISSN 2090715X. doi: 10.1155/2021/4767388. 19

Bhushan Talekar and Sachin Agrawal. A detailed review on decision tree and random forest. *Bioscience Biotechnology Research Communications*, 13:245–248, 12 2020. doi: 10.21786/bbrc/13.14/57. 10, 12

Duc Tran, Hieu Mac, Van Tong, Hai Anh Tran, and Linh Giang Nguyen. A lstm based framework for handling multiclass imbalance in dga botnet detection. *Neurocomputing*, 275:2401–2413, 2018. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2017.11.018. URL https://www.sciencedirect.com/science/article/pii/S0925231217317320. 37

Dinh Tu Truong and Guang Cheng. Detecting domain-flux botnet based on dns traffic features in managed network. *Security and Communication Networks*, 9:2338–2347, 9 2016. ISSN 19390122. doi: 10.1002/SEC.1495. 19

J.M. van Renswou. Detecting the presence of botnets by analysing tcp/ip network traffic. Master's thesis, Open Universiteit, 2021. 9, 22, 24, 25, 59, 67, 72

Jos van Roosmalen, Harald Vranken, and Marko van Eekelen. Applying deep learning on packet flows for botnet detection. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, page 1629–1636, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450351911. doi: 10.1145/3167132.3167306. URL https://doi.org/10.1145/3167132.3167306. 22, 23, 24

Jorge R. Vergara and Pablo A. Estévez. A review of feature selection methods based on mutual information. *Neural Computing and Applications*, 24:175–186, 9 2015. doi: 10.1007/s00521-013-1368-0. URL http://arxiv.org/abs/1509.07577http://dx.doi.org/10.1007/s00521-013-1368-0. 15

Thomas Vissers, Jan Spooren, Pieter Agten, Dirk Jumpertz, Peter Janssen, Marc Wesemael, Frank Piessens, Wouter Joosen, and Lieven Desmet. Exploring the ecosystem of malicious domain registrations in the .eu tld. pages 472–493, 10 2017. ISBN 978-3-319-66331-9. doi: 10.1007/978-3-319-66332-6_21. 19, 24

Harald Vranken and Hassan Alizadeh. Detection of dga-generated domain names with tf-idf. *Electronics*, 11(3), 2022. ISSN 2079-9292. doi: 10.3390/electronics11030414. URL https://www.mdpi.com/2079-9292/11/3/414. 4, 17, 18, 23, 29

Tzy Shiah Wang, Hui Tang Lin, Wei Tsung Cheng, and Chang Yu Chen. Dbod: Clustering and detecting dga-based botnets using dns traffic analysis. *Computers and Security*, 64:1–15, 1 2017. ISSN 01674048. doi: 10.1016/J.COSE.2016.10.001. URL https://www.researchgate.net/publication/308978280_DBod_Clustering_and_detecting_DGA-based_botnets_using_DNS_traffic_analysis. 21

Disha Wankhede. Artificial intelligence and its subsets: Machine learning and its algorithms, deep learning, and their future trends. 9:page no.i112–i117, 06 2022. doi: 10.6084/m9.jetir.JETIR2205914. 8

Michael Weber, Jun Wang, and Yuchen Zhou. Unsupervised clustering for identification of malicious domain campaigns. pages 33–39, 05 2018. doi: 10.1145/3203422.3203423. 19, 24

Jiaxuan Wu. Artificial neural network based dga botnet detection. *Journal of Physics: Conference Series*, 1578:012074, 07 2020. doi: 10.1088/1742-6596/1578/1/012074. 17

Congyuan Xu, Jizhong Shen, and Xin Du. Detection method of domain names generated by dgas based on semantic representation and deep neural network. *Computers & Security*, 85, 04 2019. doi: 10.1016/j.cose.2019.04.015. 17

Wei Xu, Kyle Sanders, and Yanxin Zhang. We know it before you do: Predicting malicious domains. In *Virus Bulletin Conference*, pages 73–77, 2014. URL https://www.virusbulletin.com/uploads/pdf/conference/vb2014/VB2014-XuZhangSanders.pdf. 25

Sandeep Yadav, Ashwath Kumar, Krishna Reddy, A L Narasimha Reddy, and Supranamaya Ranjan. Detecting algorithmically generated domain-flux attacks with dns traffic analysis. *IEEE/ACM Transactions on Networking*, 20:1663–1677, 2 2012. ISSN 1558-2566. doi: 10.1109/TNET.2012.2184552. 20

Ibrahim Yilmaz, Ambareen Siraj, and Denis Ulybyshev. Improving dga-based malicious domain classifiers for malware defense with adversarial machine learning. In *2020 IEEE 4th Conference on Information Communication Technology (CICT)*, pages 1–6, 2020. doi: 10.1109/CICT51604.2020.9311925. 17

Bin Yu, Jie Pan, Jiaming Hu, Anderson Nascimento, and Martine De Cock. Character level based detection of dga domain names. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018. doi: 10.1109/IJCNN.2018.8489147. 18

Mattia Zago, Manuel Pérez, and Gregorio Martinez Perez. Scalable detection of botnets based on dga: Efficient feature discovery process in machine learning techniques. *Soft Computing*, 24, 01 2019. doi: 10.1007/s00500-018-03703-8. 17, 29

Yihang Zhang et al. A ensemble learning method for domain generation algorithm detection. *Academic Journal of Computing & Information Science*, 3(4), 2020. doi: 10.25236/AJCIS.2020.030404. URL https://dx.doi.org/10.25236/AJCIS.2020.030404. 17

Yury Zhauniarovich, Issa Khalil, Ting Yu, and Marc Dacier. A survey on malicious domains detection through dns data analysis. *ACM Comput Surveys (CSUR)*, 51:67, 7 2018. ISSN 15577341. doi: 10.1145/3191329. 21, 22

Yong-Lin Zhou, Qing-Shan Li, Qidi Miao, and Kangbin Yim. Dga-based botnet detection using dns traffic. *Journal of Internet Services and Information Security (JISIS)*, 3:116–123, 2013. URL http://isyou.info/jisis/vol3/no34/jisis-2013-vol3-no34-11.pdf. 20

Xiangxin Zhu, Carl Vondrick, Charless C. Fowlkes, and Deva Ramanan. Do we need more training data? *International Journal of Computer Vision*, 119(1):76–92, mar 2015. doi: 10.1007/s11263-015-0812-2. URL https://doi.org/10.1007/s11263-015-0812-2. 29

APPENDIX A: DETAILED PROCESS FLOW

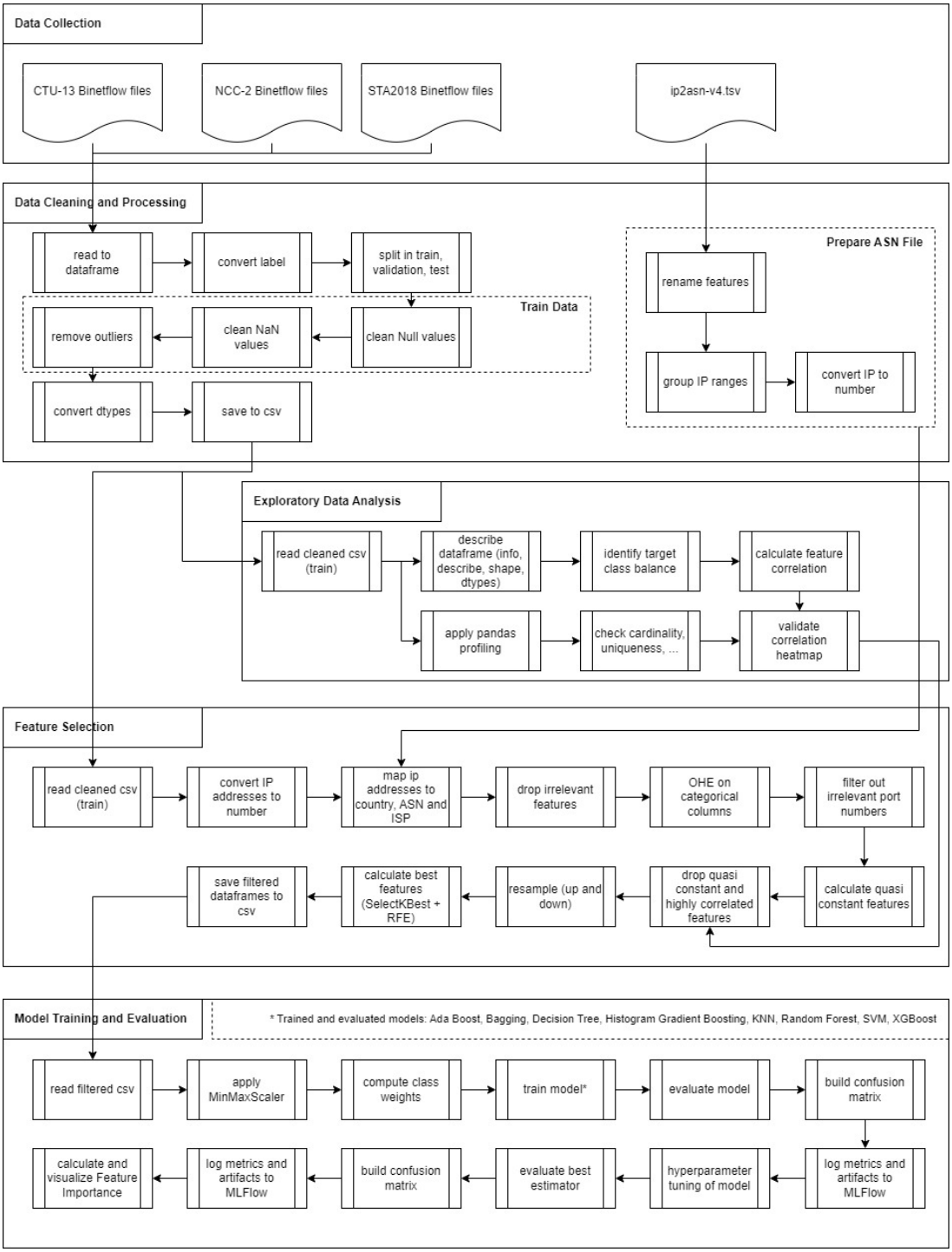


Figure 1: Detailed end-to-end ML process: From data cleaning to model evaluation