UNIVERSITY OF TWENTE.

TNO

Master Thesis

Detection of DoH tunnelling: comparing supervised with unsupervised learning

August 20, 2021

Author:
Lucas de Vries

Supervisors:
Dr. ir. H.B. Meeuwissen (TNO)
Ir. I. Chiscop (TNO)
Prof. dr. ir. R.M. van Rijswijk - Deij (University of Twente)
Dr. D. Bucur (University of Twente)

Abstract

DNS over HTTPS (DoH) makes browsing the internet more secure and increases the privacy of users. However, DoH makes it harder for network administrators to keep their networks secure. DoH can be misused for malicious purposes such as tunnelling and for resolving Command and Control (C2) addresses which is useful for botnets. The problem is that DoH traffic blends in with other HTTPS traffic, thus, it is hard to detect it. DoH can be used for benign and malicious purposes, so if DoH is detected, it is necessary to be classified as either one. In this thesis, a closer look is taken at the features that can be used to distinguish DoH network traffic from non-DoH traffic and to distinguishing benign DoH from malicious DoH traffic. The analysis of the features that are useful to detecting DoH traffic resulted in a list of interesting features. Mainly the duration of a network flow and the packet length related features were identified as features characterizing DoH traffic. To distinguish DoH used for tunnelling from benign DoH, also packet length related features are most powerful. Furthermore, both supervised and unsupervised learning methods are applied to two datasets containing both DoH and non-DoH network traffic. Whereas previous work relied on supervised learning which requires lots of labelled data, which in turn is hard to come by, this thesis applied unsupervised learning. Applying unsupervised learning methods resulted in an accuracy of roughly one to ten percent lower compared to supervised learning methods. These results prove the value of unsupervised learning for the detection of DoH traffic and can help system administrators lacking sufficient labelled data increasing the security of their networks.

Contents

List of Tables

List of Figures

1 Introduction

The Domain Name System (DNS) is used by the Internet to translate domain names to IP addresses. This is useful because domain names are easier to remember for humans compared to numeric IP addresses. Although DNS is well designed in terms of scalability, the design lacks security features that are required nowadays. DNS packets are sent over the wire unencrypted which makes the system vulnerable to eavesdroppers and a man-in-the-middle attack. In other words, confidentiality and integrity are not guaranteed.

DNSSEC is a solution that provides security by the use of digital signatures. This solves the integrity, data altered by a man-in-the-middle will be noted due to invalid signatures, but does not provide confidentiality since the queries are still sent in plain text [1]. In 2016, encrypted DNS was introduced: DNS over TLS (DoT) [29]. By using Transport Layer Security (TLS) all DNS traffic between the client and the DNS server is encrypted, which makes eavesdropping useless. This solves the confidentiality issue. Later, in 2018, also DNS over HTTPS (DoH) became a proposed standard by the IETF [28] improving privacy even further, since DNS traffic can blend in with other HTTPS traffic.

Although DoT and DoH can safely be used with only limited privacy and security issues, the DNS protocol can still be abused for malicious activities. In botnets, DNS can be used by the bots (compromised devices) to find the IP addresses of the controller of the botnet [18]. This is an essential step for the owner of the botnet since otherwise, it cannot communicate with its bots. Besides, DNS can be abused to exfiltrate data using tunnelling [21], in which an attacker hides additional data in DNS packets. Since encrypted DNS is backwards compatible with DNS, the same techniques abusing DNS can also be applied using DoT or DoH.

Because attackers have tried abusing DNS for decades, defence mechanisms have been developed. These defence mechanisms mainly rely on deep packet inspection in which, amongst others, the requested domain name is analysed. However, deep packet inspection is not effective on encrypted data. Thus, attackers can use DoT and DoH to evade the currently implemented security mechanisms detecting malicious use of DNS. Recent examples of malware using DoH are the PsiXBot and Godlua malware, in which the C2 server is resolved via a DoH request [63], [65]. Another example of abusing DoH is a sender of spam e-mails in which the malicious JavaScript via is loaded via DoH [2].

Where DoT is well detectable since it uses standardized port 853, DoH stays under the radar as it blends in with other HTTPS network traffic. Because DoT is easily detectable, a network administrator can decide to block DoT traffic to prevent malicious use of DoT.

However, the hidden DoH traffic forms a problem for network administrators trying to keep their network clean from malicious activities. Blocking the HTTPS port 443 results in blocking much more than only DoH traffic. Therefore, this research will focus on DoH. To prevent attackers from abusing DoH, firstly, the DoH needs to be detected and secondly, the DoH traffic needs to be classified as benign or malicious.

Recent studies showed that DoH network traffic is distinguishable from HTTPS web traffic. Vekshin et al. created a dataset by instructing virtual machines (VM) to visit websites [68], resulting in HTTPS traffic. The browsers in these VMs were configured to use DoH to resolve the web addresses, therefore, also DoH traffic was generated. By applying supervised learning techniques to the generated dataset, models could be created which were able to accurately detect the DoH network flows. In a similar study by MontazeriShatoori et al., a dataset was created using a similar approach [41], however, in addition to generating DoH traffic by browsers, this study also generated DoH traffic using DNS tunnelling tools. The use of DNS tunnelling tools in combination with a DoH proxy results in generated DoH tunnelling traffic, which is a form of malicious use of DoH. The study showed that malicious DoH network flows can accurately be distinguished from benign DoH flows.

1.1 Research Questions

Although multiple studies applied supervised learning methods to detect DoH, to the best of our knowledge, no study has applied unsupervised learning techniques for the detection of DoH traffic yet. In practice, labelled data is often unavailable, hence, unsupervised methods can be useful. Studies show that DoH is detectable, however, these studies use a large number of features. It is not clear what features are most useful and why they are useful. The datasets created in previous studies contain browser traffic, benign DoH traffic and malicious tunnelling traffic and are reused in this study. This leads to the following research questions.

1. Using which features can DoH traffic be distinguished from web traffic and using which features can benign DoH be distinguished from DoH tunnelling?
    - What are the underlying differences between DoH and web traffic with respect to those features?
    - What are the underlying differences between benign DoH and DoH tunnelling with respect to those features?

2. How well can DoH traffic be distinguished from non-DoH network traffic and benign DoH be distinguished from DoH tunnelling using unsupervised learning techniques?

    - What is the classification performance in terms of accuracy, recall, precision and F1-scores?
    - How much classification performance is lost by applying unsupervised learning compared to supervised learning?

The first contribution we present in this work is an analysis of existing datasets created for DoH detection. This research relies on the datasets created in previous work by Vekshin et al. [67] and MontazeriShatoori et al. [41]. In the analysis, the capturing method and data preprocessing are explained and the completeness and representativeness are discussed.

Secondly, we provide a feature analysis in which we examine features selected to detect DoH in previous work. The value distribution of each feature is looked at and the differences in the characteristics of DoH and non-DoH traffic are explained. Apart from that, the correlation between features is analysed, which is useful for the feature selection before applying machine learning algorithms.

The third and main contribution is the use of unsupervised learning methods for both the detection of DoH and the detection of malicious use of DoH. This is a novelty since previous work only applied supervised learning methods. A systematic approach is followed, which includes data pre-processing, thorough feature analysis and selection, hyperparameter tuning and finally, the training and testing of the unsupervised methods.

The rest of this thesis is structured as follows. In Section 2, background information that will help understand the rest of the thesis is described. In Section 3, an overview of other research related to DoH detection is given and the significance of this work is explained. The datasets created in previous studies for DoH detection are analysed and the completeness and representativeness are discussed in Section 4. In Section 5, the methods used to do the feature analysis and to apply both supervised and unsupervised machine learning techniques are described. In Section 6, the results of the feature analysis and the results of the applied machine learning are shown and discussed. In Section 7, answers to the research questions are provided and the conclusions are formulated. Finally, in Section 8 future research directions are discussed.

2 Background

This section provides background information to familiarize the reader with key concepts such as DNS, HTTPS and DoH. Furthermore, DNS-based techniques which are often employed by botnets to establish communication channels between the bot and the botmaster are explained. Since DoH is backwards compatible with DNS, these DNS-based techniques are relevant because they can be applied in a DoH environment.

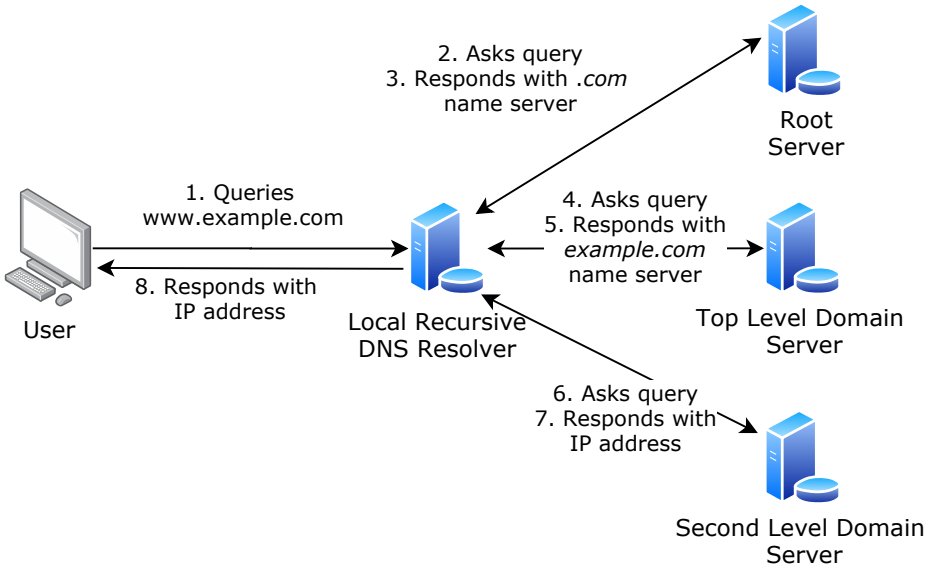

Figure 1: Recursive DNS Resolving

2.1 DNS

The Domain Name System (DNS) is a fundamental system for the Internet translating user-friendly domain names into IP addresses. The system is implemented as follows: when a user program wants to resolve a domain name, a query for that domain name is sent to the resolver. The resolver sends a query to the root name server and the root name server answers with the address of the top-level domain. The resolver recursively queries the address it received from the previous query until the IP address corresponding to the request is found. This process is described in RFC 1035 [45] and illustrated in Figure 1. Messages to DNS servers are sent to port 53, usually using UDP.

An example of a user that wants to visit a web page in a browser is shown in Figure 2. All traffic, except the traffic between the browser and the web server, is unencrypted.

Therefore, deep packet inspection is possible in DNS and the queried domain name is visible.

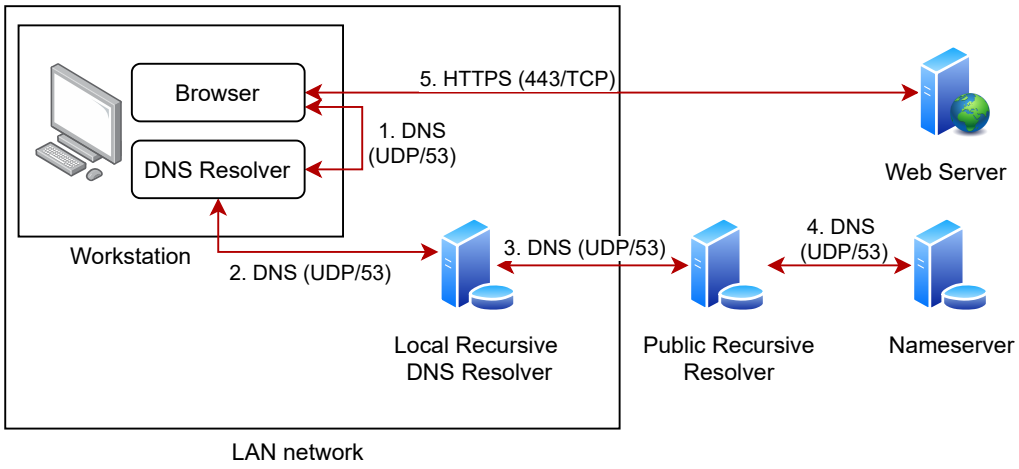

Figure 2: DNS resolving once a user visits a web page using a browser (inspired by [27])

2.2 HTTPS

Hypertext Transfer Protocol Secure (HTTPS) is a widely used secure communication protocol. As described in RFC2818 [52], HTTPS is the secured version of the HTTP protocol. The difference is that HTTPS traffic goes over the TLS layer instead of directly using the TCP layer. Due to the use of TLS (RFC8446 TLS 1.3[29]), the traffic is encrypted. TLS provides confidentiality, integrity and authentication. An HTTPS session starts with a TLS handshake in which the cipher suite will be selected and the symmetric session keys will be generated and shared.

2.3 DNS over HTTPS

In Sections 2.1 and 2.2, DNS and HTTPS are briefly explained. Naturally, DNS over HTTPS (DoH) uses both. The DNS packet is transformed into an HTTP packet. After applying encryption, an HTTPS packet is formed and send over the wire. Due to the encryption, the DNS content is no longer visible. The requested domain name, or even the fact that it is a DNS related packet remains unknown for eavesdroppers.

RFC8484 [28] describes how DNS over HTTPS should be implemented. The DoH Server is available on a specified URI. A DoH client can send an HTTP request to the specified URI, and this can be either an HTTP GET or an HTTP POST request. The DoH

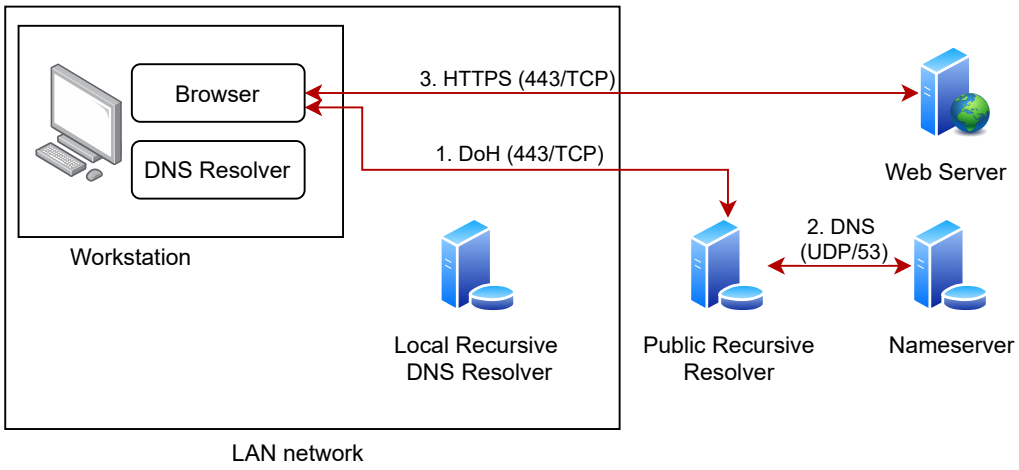

Figure 3: The DoH process once a users visits a website in a DoH enabled browser (inspired by [27])

server replies to the request by an HTTP response with the query and the result in the body. The minimum recommended HTTP version is HTTP/2. Figure 3 shows an example of the DoH process once a user visits a web page in a DoH enabled browser.

The RFC discusses security considerations for the application of DoH. One of the considerations is about traffic analysis. Since this consideration potentially changes the characteristics of DoH traffic, it is important to delve into it. The RFC mentions that an attacker might be able to gain information about the DNS queries by traffic analysis. As a solution, HTTP/2 compression or padding can be applied. However, if a DoH client uses HTTP/2 padding, the DoH server does not necessarily apply HTTP/2 padding as well. Therefore, a better solution is the use of DNS padding. RFC 8467 lists possible padding policies for encrypted DNS traffic and discusses the implications of each padding option [38]. Although the RFC does not prescribe a certain padding length, DoH servers can choose a certain length and aim to create DoH responses of constant length. DoH traffic with constant length might help to classify traffic as DoH traffic, but it will further obfuscate the content of the traffic.

2.4 DNS-based techniques used by botnets

A botnet is a group of infected devices (called zombies or bots) controlled by the so-called botmaster. The botmaster controls the bots via command and control (C&C or C2) software. Botnets can be used for malicious activities like performing DDoS attacks or sending spam. In the study by Al-Mashhadi et al. [5] concerning DNS-based botnet

detection, two techniques that botnets often use to make it difficult to shut down a botnet are mentioned. Since the techniques are based on DNS, the techniques can also be used in a DoH environment.

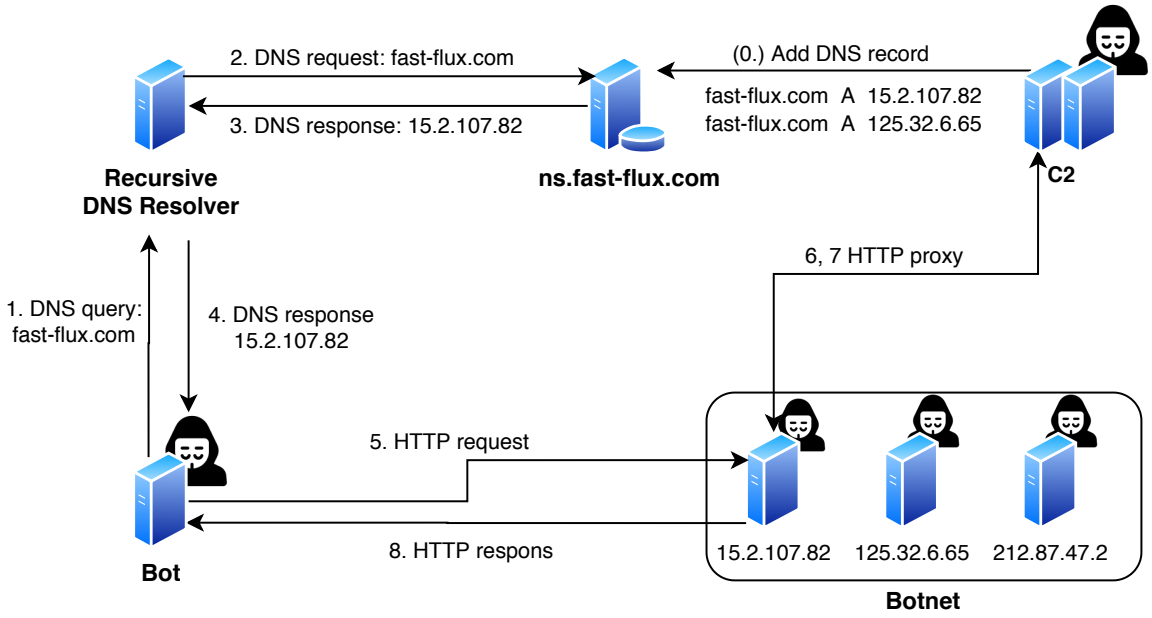

Figure 4: A new bot communicates with the botmaster via a Fast-Flux Service Network (FFSN) (figure inspired by [12])

A botmaster has to use defensive techniques because if a botmaster only uses one IP address or one domain name for its C2 server, it has a single point of failure; once the server related to the IP or the domain name is shut down, it loses control over the botnet. Therefore, botnets often use fast domain fluxing or Domain Generating Algorithms (DGA). Fast-Flux Service Network (FFSN) is a technique to hide the true location of the botnet server [64]. The idea is that a domain is resolved to hundreds or even thousands of IP addresses with a short Time-To-Live (TTL). Each IP address corresponds to a so-called flux agent. Once a connection to a flux agent is initiated, the flux agent redirects the request to the C2 server and it replies with the response of the C2 server. This way, it is hard to shut down the C2 server since it is hidden behind the flux agents. The concept is shown in Figure 4.

Another technique used in botnets to make them more resilient against shutdown attempts involves DGA. The concept is shown in Figure 5. First, the botmaster creates a DGA with a seed. The seed is a set of parameters used as input for the DGA. The

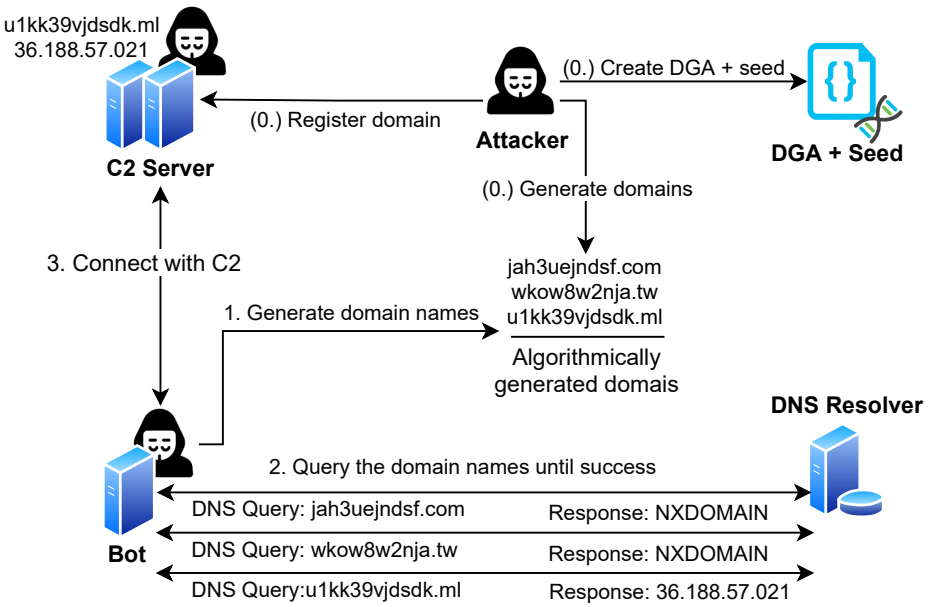

Figure 5: Illustration of the purpose of DGAs. Figure inspired by [11].

botmaster generates a number of domain names and registers only one or a few of them to use them as C2 server. Next, a bot generates the same domain names based on the deterministic algorithm [70]. The bot queries a DNS resolver until a domain name registered by the botmaster is found. Once the C2 domain name is found, the bot connects to it. In practice, it is often infeasible to use a blocklist approach to block all domain names generated by a DGA. Hence, the botnet is more resilient against shutdown attempts.

3 Related Work

This chapter discusses the literature relevant for DoH detection or the detection of malicious use of DoH. The related work is divided into four parts. It starts with the broad scope of detecting malicious DNS traffic in general (1) and the classification of encrypted data (2) and gradually narrows down to the most directly related work to this research with the detection of DoH (3) and finally the detection of malicious use of DoH (4). The first part of the literature review concerns the detection of malicious use of DNS. Under the assumption that malicious DoH usage is similar to the usage of malicious DNS, research to the detection of malicious DNS usage can give insights and ideas on how malicious usages of DoH can be detected. The second part is about the classification of encrypted data in general. Since DoH is a form of encrypted data, general data classification techniques can be applied to the DoH detection problem. Because these techniques are not used in this work, the related work about general encrypted data classification is placed in Appendix 8. Thirdly, research into the detection of DoH will be looked at. Finally, research into the detection of malicious DoH usage will be scrutinized. This chapter concludes by arguing the relevance of this work.

3.1 Malicious traffic detection using DNS

Two ways of malicious DNS uses will be covered, namely, DNS used for C2 in botnets and DNS used for data exfiltration. DNS can also be abused as an amplifier in DDoS attacks, however, that is strictly related to UDP traffic and not relevant for DoH.

3.1.1 Botnet Command and Control Activity

One approach to identifying botnet activity is by monitoring network traffic and detecting machines simultaneously sending DNS queries [9]. As explained in Section 2.4, a technique used in botnets is generating domains by a Domain Generating Algorithm (DGA). The bot queries a lot of domains and the botmaster only has to register a few of them to set up Command and Control (C2). The use of DGA results in a notably large amount of DNS queries. Most of these queries result in an NXDOMAIN (Non-existing domain) response. This phenomenon is detectable by analysing NetFlows [25], as the ratio of DNS queries and connections made is higher than usual.

Another approach to detecting botnet related DNS traffic, also described by Bumanglag and Kettani [9], is by actively measuring the additional DNS record information. Apart

from the A or AAAA record types, legitimate domains often have an MX record to specify the address of the mail server. The lack of other records than an A or AAAA record type indicates the queried domain is not a usual domain. A device querying a lot of unusual domains indicates that the device can be part of a botnet.

Other techniques to detect botnet activity are based on the detection of domain names generated by a DGA. Plohmann et al. [50] did a comprehensive study on DGAs, published in 2016. By reverse-engineering 43 DGAs in total, it became clear that DGAs mainly use arithmetical functions to generate the domains. However, some are based on hash functions or word lists and one specific DGA uses permutations. Furthermore, the seeds used as input to the DGA are analysed and insights about the string length of the Automatically-Generated Domains (AGDs) and the used Top Level Domains (TLD) were obtained. By reimplementing the DGAs, the corresponding AGDs were reproduced. The researchers combined these domains and the DomainTools WHOIS dataset [20], containing WHOIS records. The analysis showed that only 0.62% of the AGDs were actually registered, of which approximately 30% was already registered long before the botnet appeared, 20% was likely registered by law enforcement agencies and 50% presumably by the botmaster. Domains are often registered a few days before they become valid by the botmaster, however, sometimes the law enforcement agencies register a domain earlier and a few botmasters register a domain a (few) year(s) in advance. Although this research gave a lot of new information about DGAs, which can be useful in the battle against botnets, it is hard to apply the knowledge on DoH traffic since DoH traffic is encrypted and the DNS query is hidden.

Zago et al. [70] compare machine learning (ML) based approaches detecting generated domain names. The advantages and disadvantages of various machine learning techniques (like deep learning, decision tree, etc.) are discussed. Additionally, the authors analyse the importance of each selected feature. The most important features include string length, the ratio of numerical characters and pronounceability score. Apart from the string length feature, the features cannot be used on encrypted DNS traffic.

All in all, most of the current DNS based botnet detection techniques are dependent on deep packet inspection. Since DoH traffic is encrypted, deep packet inspection becomes useless. Therefore, most of the current approaches cannot directly be used on DoH traffic. One method, however, is flow-based and therefore, this approach can be useful.

3.1.2 Data exfiltration

DNS is a fundamental component for the Internet and therefore, DNS traffic cannot simply be blocked by a firewall. Even when DNS traffic is blocked, there will always be an internal DNS resolver and as long as that resolver can resolve external names, exfiltration via DNS is possible. Data can secretly be exfiltrated to a domain owned by an attacker using DNS. It works as follows: the attacker sends a query containing the data to the DNS resolver. For example, `secretdata.attacker.com` is queried, where `secretdata` is the data that the attacker wants to exfiltrate and `attack.com` is a domain name controlled by the attacker. Eventually, this query will show up at the attacker-controlled nameserver (see Figure 1) and the attacker successfully exfiltrated the data. Using the same technique, DNS can feature as a tunnel for other protocols like FTP, HTTP or VPN.

Ellens et al. detected DNS tunnelling by analysing network flows [21]. They created a dataset with normal (captured) DNS data and generated data simulating three types of DNS tunnels: file transfer, interactive session (shell command) and web browsing. Network flows were extracted and after analysing the differences in the characteristics of benign and malicious traffic, various anomaly detectors could be designed. The number of bytes per flow and the number of flows per time were found to be important features. The results show that the most accurate detector was able to detect all malicious flows with a relatively small number of false positives. Although the flows can be different in a DoH setting compared to DNS flows due to TLS handshake and other overhead, this method is not depending on deep packet inspection, so it can be a useful method in a DoH setting.

Ishikura et al. used DNS cache properties to detect a DNS tunnel [31]. The idea is that the domain names used in queries in a DNS tunnel needs to be unique, otherwise they do not reach the name server of the attacker. The two features derived from cache properties are: cache hit ratio (CHR) and access hit ratio (AHR). The CHR is computed by dividing the number of queries that can be resolved using the cache set of the DNS resolver by the total number of queries. DNS tunnels are likely to decrease this ratio because they query a lot of unique domains. Since a DNS resolver only caches domain names that are frequently queried, a legitimate query to an uncommon domain name may also decrease the CHR. To overcome this problem, the concept of access entries is introduced. A list of access entries contains all queried FQDNs. The AHR is computed by dividing the queries for domains in the access entry list by the total number of queries. The authors showed through numerical experiments that a DNS tunnel causes a drop in both CHR and AHR.

The drop is only significant when the tunnel sends a sufficient amount of queries per time interval. This approach requires knowledge about CHR and AHR, which is not available in a scenario where DoH is used.

In the past decade, the research in the DNS data exfiltration domain mainly focused on high-throughput DNS tunnelling detection while an example showed that a low-throughput tunnel was used to exfiltrate data regarding 56M credit cards [4]. Nadler et al. provided a solution to detect both high-throughput and low-throughput DNS tunnelling [42]. The detection method is based on an anomaly detection model, namely Isolation Forest. Isolation Forest is an unsupervised learning method and the model is only trained on the legitimate data [37]. Before training, first, the features need to be selected and extracted. The following features are selected: (1) character entropy of the FQDN, (2) ratio of non-IP queries (e.g. request of TXT instead of AAAA (IPv6)), (3) unique query ratio, (4) unique query volume, (5) average query length and (6) longest meaningful word, divided by domain length. Important to mention is that during the data gathering, the queries are grouped per primary domain, which is a combination of the top-level domain and second-level domain. Data is gathered at the DNS resolver every n minutes, where n is configurable with an example value of 15. The solution was tested using a large dataset from a DNS resolver with over a hundred thousand end-users. The malicious traffic of two low-throughput DNS tunnelling malware and two high-throughput tunnelling tools were added to the dataset. By comparing with other research, it became clear that prior proposed solutions were able to detect the high-throughput tunnels, while only the proposed solution of this research was able to detect the low-throughput tunnels. The used features are dependent on, or require information about, the DNS query. In a DoH scenario, this query is encrypted, thus, these features cannot be used.

In conclusion, the flow-based detection method by Ellens et al. [21] can be interesting since it does not rely on deep packet inspection. The methods based on monitoring the cache hit ratio requires information about cache hits, which is normally unavailable, both when using DNS and DoH. Finally, the machine learning approach requires features obtained by deep packet inspection. Due to the encryption in DoH, these features cannot be used.

Table 1 gives an overview of the malicious DNS detection methods discussed in this thesis. As shown in the table, the DGA detection method by Bumanglag et al. [9] can be used to detect malicious DoH once DoH is distinguished from other types of traffic. Besides, the DNS tunnelling detection method by Ellens et al. [21] can be used to detect DoH tunnels. The other methods are relying on the content of a DNS packet which is

Study	Detecting	Required data	Usable for DoH?	Technique
Bumanglag et al. [9]	DGA	NetFlows	✓, once DoH is detected	Monitor the ratio of DNS queries and new connections initiated
Bumanglag et al. [9]	DGA	Domain name	✗	Check the domain for additional DNS records (such as MX-record)
Plohmann et al. [50]	AGD	Domain name	✗	This study analyses characteristics of generated domains
Zago et al. [70]	AGD	Domain name	✗	Use advanced machine learning methods to detect AGDs
Ellens et al. [21]	Tunnelling	NetFlows	✓	Anomaly detection based on flow statistics
Ishikura et al. [31]	Tunnelling	DNS cache hits/misses	✗	Monitor the ratio of cache hits/misses
Nadler et al. [42]	Tunnelling	DNS query	✗	Isolation Forest classifier based on query related features

Table 1: Overview of malicious DNS detection methods

unavailable when DoH is used, due to the encryption.

3.2 DoH detection

Apart from studies about encrypted data classification, there is literature specifically addressing DoH traffic detection.

As explained in Section 2.3, DoH traffic uses port 443, the usual port for HTTPS traffic. The encrypted DoH traffic is blended together with other HTTPS traffic. Vekshin et al. used machine learning to detect DoH traffic [68]. First, they created a dataset containing both DoH traffic and HTTPS traffic. The data was collected by controlled virtual machines interacting with DoH enabled browsers (Mozilla Firefox and Google Chrome) and by a Raspberry Pi, which was installed as a DNS resolver in two university offices and used Cloudflare DoH to resolve the DNS queries. The data collection process ran over a period of three months. Based on this data, a list of 18 features was composed. The most important features are: flow duration and the average interval between packets of a flow. Five machine learning techniques were applied to the data and they all achieved high accuracy. The Ada-boosted Decision Tree technique achieved the highest accuracy: 99.6%. This researched showed that DoH traffic from DoH resolvers configured in browsers or in networks can be detected. However, the method only works for connections with multiple DNS queries since just a single query is too similar to an API request/response

and therefore, not distinguishable.

MontazeriShatoori et al. also did a study on the detection of DoH. They followed an approach similar to the approach of Vekshin et al. [41]. First, a labelled dataset is generated and after the data preprocessing step, machine learning is applied. The dataset was created by simulating web browsers and capturing the HTTPS traffic. The browsers were controlled by a script via Selenium [60]. The browsers were configured to connect to one of the following DoH servers: AdGuard [3], Cloudflare [14], Google DNS [24] or Quad9 [51]. By visiting a series of the top 10k Alexa websites [6], both HTTPS web traffic and DoH traffic was generated. Three types of features were selected, namely, statistical features, clumps, and handshake fingerprints. The statistical features are features related to the NetFlows such as the duration of a flow or the packet length variance. Additionally, the concept of clumps is introduced by the authors. A clump is defined as a sequence of one or more packets with the same source and destination address in the same direction within a certain time frame. The idea of using clumps is that the application data of multiple packets can be combined to minimize the effect of TLS and IP segmentation. Of each clump, the following is stored: size, packet count, direction, duration and interarrival time. A sliding window is used to create segments of clumps of a certain size. The size can be configured as a hyper-parameter. Apart from the clumps, the unencrypted information from the TLS handshake like available cipher suites, etc., is used to create a fingerprint. The type of features is a list of 28 statistical properties about the packet size, packet time and the request/response time difference. Classification is done by statistical feature classifiers and a time-series classifier. The classifiers used for classification based on statistical features were Random Forest (RF), Decision Tree (C4.5), Support Vector Machine (SVM), and Naive Bayes (NB) machine learning algorithms and deep neural network (DNN) and convolutional neural network (CNN). The time-series classifier is a deep learning model, trained to classify the clumps created during preprocessing. The results showed that the statistical classifiers scored F1-scores between 0.833 and 0.993 (RF and C4.5). The time-series classifier scored, depending on the clump size hyper-parameter, F1-scores of 0.876 and 0.998. Although the results are promising, it is arguable if the classification is also this accurate if the dataset also contained other HTTPS traffic apart from web traffic. For example, traffic-related to instant messaging has characteristics more similar to DoH than usual web traffic. Both traffic types namely have flows with a long duration consisting of relatively small packets.

In conclusion, Vekshin et al. and MontazeriShatoori et al. showed that DoH traffic generated by a browser can be accurately detected mainly by the long flow duration.

MontazeriShatoori et al. also demonstrated that DoH traffic can be detected by time-series based classification. Both studies include machine learning methods, however, only supervised learning methods are applied.

3.3 Detection of malicious use of DoH

Under the assumption that DoH traffic can be detected, it can be classified into benign and malicious usage of DoH. Three studies have looked into this classification of DoH traffic. The first one tried to detect DGA related DoH traffic and the two others tried to detect DNS tunnelling.

Patsakis et al. [47] were able to identify traffic related to a DGA based on encrypted DNS traffic. The domain names generated by ten DGAs and Alexa's top 1000 domain names were listed, 11.000 domain names in total. A dataset is created by querying all of these domains using a DoH resolver. Based on the fact that DGAs repeatedly generate domain names and queries them until an existing domain is found, the traffic of the DGA domains should have other characteristics compared to the traffic related to Alexa's domains. These differences can be expressed in terms of time and packet size. By applying a time series filter, the trend of packet size over time could be computed. By applying statistics, these trends could be used as an indicator, indicating botnet activity in encrypted DNS traffic. It was shown that approximately 30 DNS queries provide enough information to determine whether the traffic relates to botnets or not. The research did not apply the method in a real-world scenario. In contrast to the research where the classifier only got samples of one class, to determine whether the input is benign or malicious, in a real-world scenario, the input will be a mix of malicious and benign. Thus, the research showed that the characteristics of benign and malicious encrypted DNS traffic regarding packet size are different. However, it is unclear how well the distinction can be made in a real-world scenario.

In the same work of MontazeriShatoori et al. discussed in Section 3.2, apart from the classification of DoH and non-DoH, they also classified the DoH in malicious DoH and benign DoH [41]. The dataset contained benign DoH generated by browsers configured to use DoH resolvers (as explained before). The malicious DoH traffic was generated by the following DNS tunnelling tools: Iodine, DNS2TCP, DNScat2. These tools normally generate DNS data, however, a DoH proxy was used. The traffic between the DoH proxy and the DoH server was captured. The exact same classification techniques as described in Section 3.2 were used. The results showed that the classification is accurate with F1-

scores between 0.832 and 0.999 for the statistical classifiers and between 0.782 and 0.999 for the time-series classifier.

From the same dataset, generated by MontazeriShatoori et al., Sunil Kumar et al. used the benign and malicious DoH network traffic to detect the malicious DoH network traffic [62]. By applying various common machine learning methods to the dataset, they found that the Random Forest and Gradient Boosting techniques work best for classifying DoH traffic. With both techniques, less than 20 samples out of 67k were misclassified. These results are slightly more accurate compared to MontazeriShatoori et al. Since these techniques are both ensemble learning methods, it is suggested that ensemble learning suits this problem best.

In short, Patsakis et al. showed that DGA generated DoH traffic can be distinguished from normal DoH traffic, under the assumption that normal and malicious DoH are separated from each other. MontazeriShatoori et al. and Sunil Kumar et al. looked into DoH tunnelling detection and showed that DoH tunnelling could be distinguished from browser generated DoH traffic. By applying supervised learning techniques DoH tunnelling could accurately be detected.

3.4 Relevance of this work

Related work showed that it is possible to detect DoH based on NetFlow related features, however, the studies used a large number of features, which makes it unclear what the most important features are. The first contribution of the work in this thesis is an analysis of the features used in previous work in order to explain what features are important and why these features distinguish DoH from non-DoH.

By looking at detection methods of malicious use of DNS, it became clear that a method detecting DNS tunnelling can also be applied to detect DoH tunnelling. This concept is used by two studies to the detection of (malicious) DoH. By applying supervised learning methods to a dataset with NetFlow statistics of both DoH and non-DoH network traffic, the DoH traffic could be detected. Similarly, two studies showed malicious use of DoH can also be detected using supervised learning methods using features related to NetFlow statistics. These studies, however, only apply supervised learning techniques. Supervised learning requires labelled data which in practice often is scarce. No study has used unsupervised learning methods to detect DoH or malicious use of DoH yet. The second contribution of this thesis is that it presents unsupervised learning methods detecting both DoH and malicious use of DoH in the form of tunnelling.

4 Analysis of datasets

Two publicly available datasets are used in this research, both datasets are selected because both contain labelled DoH traffic. The first one is published by Vekshin et al. in 2020 [67], it will be referred to as the CTU dataset since the researchers are from the Czech Technical University. The second dataset is published by MontazeriShatoori et al. in 2020 [41] and it will be referred to as CIC because it is a project of the Canadian Institute for Cybersecurity. Respectively, the datasets are related to the papers [68] and [41].

In this chapter, the methods used to create the datasets will be described and information about the content of the datasets will be given both the CTU and the CIC dataset. Afterwards, the completeness and representativeness of both datasets will be discussed. Details about the selected features based on these datasets will not be discussed in this chapter, the feature analysis is part of Section 5.1.

4.1 CTU dataset

Vekshin et al. published a paper in which they demonstrated that DoH network traffic is distinguishable from HTTPS web traffic [68]. In the process, the researchers created a dataset containing both DoH network traffic and non-DoH network traffic on which the designed classification methods could be tested. In this section, the methods used to create the dataset will be described and details about what kind of network traffic the dataset contains will be listed.

4.1.1 Capturing method

The dataset can be divided into two parts, based on the capturing methods. The first part is network traffic generated by virtual machines visiting web pages and the second part is the result of capturing network traffic from university offices. The capturing methods are illustrated in Figure 6.

The left side of Figure 6 shows the data generated using Virtual Machines (VM's). The concept behind the setup is to automatically visit web pages, which result in both HTTPS web traffic and DoH traffic. HTTPS traffic is generated because the web pages are fetched over HTTPS and the DoH traffic is generated since the browsers are configured to use DoH instead of DNS to translate the domain name in an IP address. The concept is realized by using VM's on which Mozilla Firefox and Google Chrome are installed. The browsers are configured to use Cloudflare as DoH resolver and used to visit

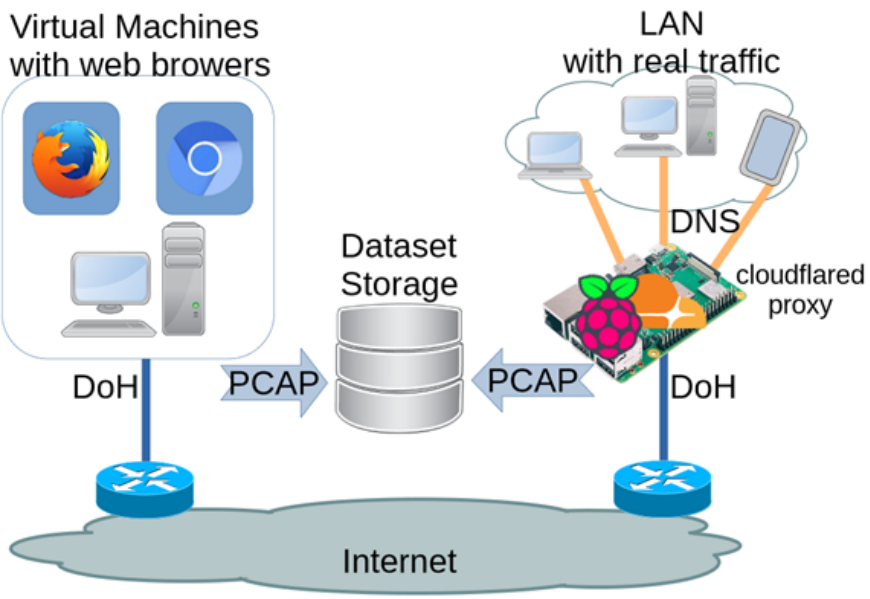

Figure 6: Testbed used to generate and capture the network traffic for the CTU dataset. Figure source: [68].

Alexa's top websites list [6]. The browsers are controlled by the Selenium framework [60], which shows the user browsing according to a predefined script. The network traffic was captured on the host machine of the VM's, listening to the network interface of the VM's.

The right side of Figure 6 show the data captured from university offices. For two university offices, the IP address of a Raspberry Pi was configured as a local DNS resolver. Auto-configured devices connected to the office network use this DNS resolver on default. On the Rapsberry Pi, the DoH proxy `cloudflared` [15] was installed. Consequently, all DNS requests sent to the Raspberry Pi result in DoH requests to Cloudflare. Both DNS and DoH traffic are captured on the Rapsberry Pi. The capture lasted almost three months and covered up to 20 devices, including computers, laptops and smartphones, which were all operational during the capture.

Apart from the connected devices, scripts were used to generate DNS requests. The idea is that the scripts simulate a busy network. The paper does not elaborate on the details such as the number of DNS requests per second. Due to the high number of DNS requests, the DoH queries end up in a sequence instead of in single queries, resulting in different connection patterns. The DNS requests generated by the scripts are queries for the domain names Alexa's list of top websites.

To complete the dataset, extra HTTPS traffic was added, namely traffic produced by

Source	Non-DoH Flows	DoH Flows
Firefox	380940	705
Google Chrome	71686	730
Cloudflared	452626	34195

Table 2: The CTU dataset contains 940K network flows of which 36K are DoH flows.

Instant Messaging clients (IM). The authors of the paper believe that the characteristics of IM traffic are likely similar to DoH traffic due to a similar request-response scheme and small packets.

4.1.2 Data processing

For further analysis, it is useful to extract features from the captured pcap files. First, IP flows were extracted using a flow exporter called `flow_meter`. This exporter is part of the NEMEA system [10]. Secondly, packet-level statistics were extracted using the PStatplugin and an additional python script. The ground truth label could be inferred by knowing how the proxy is set up and by knowing the IP addresses of the devices and the DoH servers.

4.1.3 Content

The data generated by VMs (left side of Figure 6) contains primarily TLS traffic. The TLS traffic is a result of (1) the visited web pages, which are fetched over TLS, and (2) the DoH traffic, since the browsers are configured to resolve the domain names using DoH. The non-TLS traffic in the dataset includes UDP traffic to Google servers and various DNS request. Only the TLS traffic is used.

Almost 5,000 web pages were visited using Firefox, resulting in 381,000 flows. With Google Chrome almost 1,000 pages were visited, resulting in 72,000 flows. In Tabel 2 the division of DoH/non-DoH flows is shown, it becomes clear that most of the flows generated by the VMs are non-DoH flows (99,7%).

All traffic captured in the university offices (right side of Figure 6) can be categorized in DNS or DoH traffic. This is a result of the capturing method. The Raspberry Pi, acting as DNS resolver, configured to use the Cloudflare DoH server, captured both DNS and DoH traffic. After three months of capturing, 453K non-DoH flows and 34K DoH flows were captured, see Table 2.

4.2 CIC dataset

Another study addressing DoH detection and proposing a new dataset for this purpose was published by MontazeriShatoori et al. Apart from detecting DoH, MontazeriShatoori et al. also focussed on classifying DoH traffic as either benign or malicious. As shown in Figure 7, the dataset can be divided into two layers. In the first layer, the DoH traffic can be distinguished from the web traffic and in the second layer, the benign DoH traffic can be distinguished from the malicious DoH traffic. This section describes the methods used to create the CIC dataset and the content of the dataset.

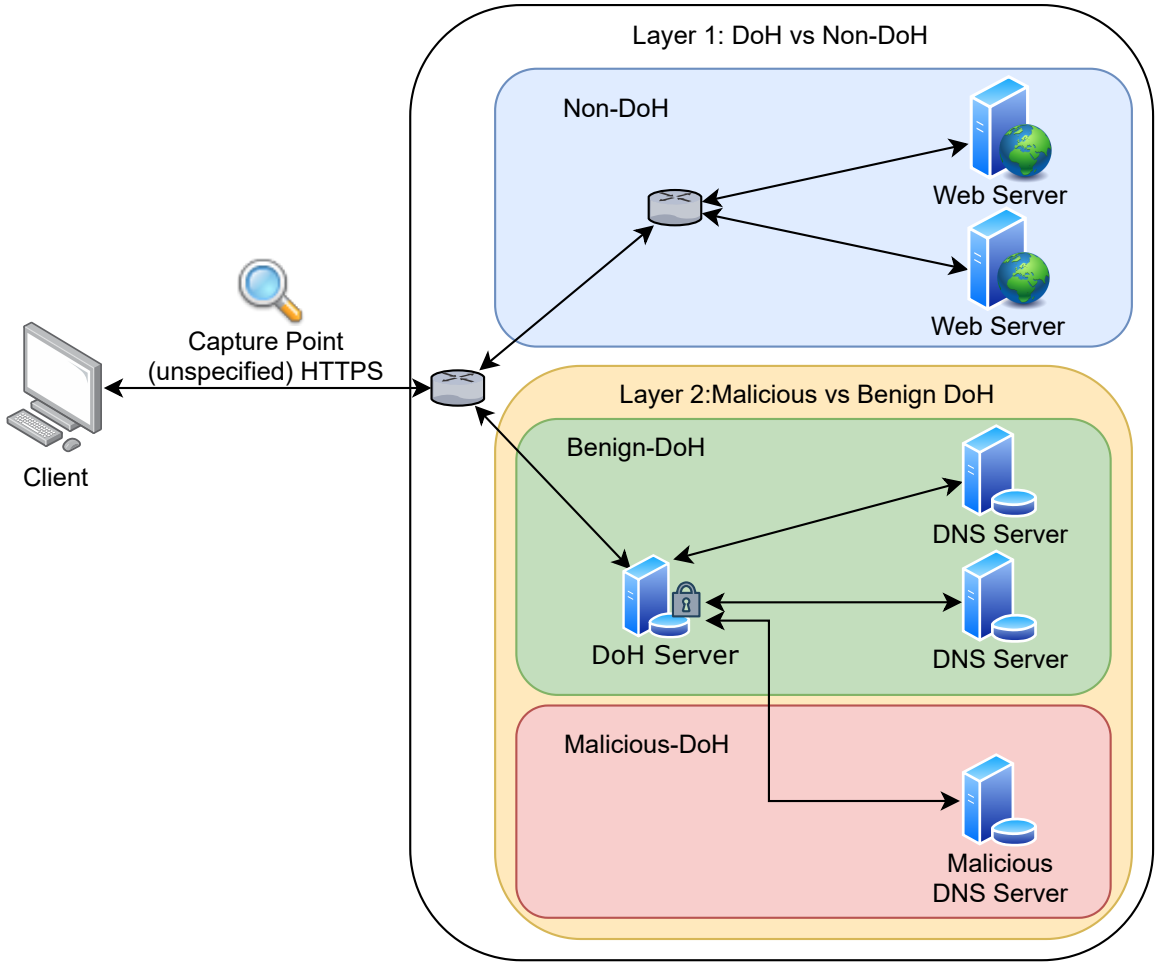

Figure 7: Testbed used to generate the network traffic for the CIC dataset. Figure inspired by [41].

4.2.1 Capturing method

The CIC dataset is generated quite similarly to the CTU dataset. VMs in which Google Chrome and Firefox are installed are used. The browsers are configured to use a certain DoH server to resolve the IP addresses of websites. Via the Selenium scripts, the browsers are controlled and instructed to visit the top 10k Alexa's websites. By capturing the HTTPS traffic, both non-DoH web-related traffic and benign DoH traffic could be captured. These captures correspond to the blue and green boxes in Figure 7.

For the generation of the malicious DoH traffic, in short, DNS tunnelling tools are used in combination with a DoH proxy. DNS tunnelling tools can be used since DoH is backwards compatible with DNS, i.e., a DoH request can be constructed based on a DNS packet.

To be precise, to set up the tunnelling tools, a tunnelling server and a client were configured. For the server, a domain name was registered and the authoritative nameserver was configured such that the DNS tunnelling tools receive the DNS requests. The client uses a DoH proxy which encapsulates every outgoing DNS request. The encapsulated request is sent to the DoH server and the response from the DoH server is relayed as a DNS response. The data is captured between the DoH proxy and the DoH server using tcpdump.

The described process is implemented in a python tool in which the following can be configured: DoH server, tunnelling tool, tunnelling configuration: settings such as the delay between sending requests and the DNS record types being used, transmission rate and the tunnel duration.

4.2.2 Data processing

To extract the features from the pcap captures, the researchers built another tool, called DoHMeter [40]. The python tool loops through all network traffic of a given pcap file and extracts statistical and time-series features of a reconstructed network flow. A network flow is reconstructed using the IPv4 address, the used port, time to live value and the TCP flags. Flows are cut after a flow duration of 120 seconds and after every 10,000 packets, taking all flows into account.

After extracting the feature using DoHMeter, a few rows contained Not a Number (NaN) values. This is a result of flows containing not enough packets to compute certain statistics. The rows containing NaN values are discarded.

Type	Flows
Non-DoH	889809
Benign DoH	19746
Malicious DoH	249553

Table 3: Specifying the flows per type (corresponding to Figure 7) shows that the benign DoH class is relatively small compared to the others.

4.2.3 Content

The network traffic related to the first layer, in which VM's were used to visit web pages and the browsers were configured to use DoH, resulted in more than 900K network flows. Most of these flows are Non-DoH flows (890K) and the rest of the flow (almost 20K) are DoH flows, see Table 3. Table 4 shows the distribution per browser and DoH server. The browsers Google Chrome and Mozilla Firefox are used and AdGuard, Cloudflare, Google DNS and Quad9 are the used DoH servers.

The second layer related to network traffic, containing DoH traffic as a result of DNS tunnelling tools, contains almost 250K flows. Table 4 shows the distribution per tunnelling tool. Three tunnelling tools are used, namely, dns2tcp [17], DNSCat2 [30] and Iodine [22]. The same four DoH servers as in the first layer are used.

Since only the HTTPS traffic is captured, the dataset only contains HTTPS traffic.

4.3 Discussion and conclusions

This section discusses the findings of the analysis of the datasets by looking at the methods used to create the datasets, the volume and the content of the datasets. In conclusion, the completeness and representativeness of the datasets will be evaluated.

The datasets are largely artificially created which makes it hard to tell how well the datasets compare to real data. Recent studies show similar approaches to generate network traffic are used. Nogues et al. [44] provide a framework that generates labelled traffic that can be used for anomaly detection which is based on virtual machines. Apart from virtual machines, Clausen et al. [13] introduce a network generating framework that uses Docker [19] containerization. The low-level differences between traffic directly generated by a physical machine and traffic generated via a VM, in terms of speed and number of system calls are similar according to Crussell et al. [16]. However, that does still not imply that the nature of generated traffic is comparable to real traffic.

Arguably, the difference between a user browsing websites and generating DoH and

Browser/Tool	DoH Server	Packets	Flows	Type
Google Chrome	AdGuard	5609k	105141	HTTPS (Non-DoH and Benign DoH)
	Cloudflare	6117k	132552	
	Google DNS	5878k	108680	
	Quad9	10737k	199090	
Mozilla Firefox	AdGuard	4943k	50485	
	Cloudflare	4299k	90260	
	Google DNS	6413k	138422	
	Quad9	4956k	92670	
dns2tcp	AdGuard	1281k	5459	Malicious DoH
	Cloudflare	3694k	6045	
	Google DNS	28711k	17423	
	Quad9	8750k	138588	
DNSCat2	AdGuard	1301k	5369	
	Cloudflare	12346k	9230	
	Google DNS	48069k	11915	
	Quad9	19309k	9108	
Iodine	AdGuard	3938k	11336	
	Cloudflare	5932k	14110	
	Google DNS	73459k	12192	
	Quad9	22668k	8975	

Table 4: The content of the CIC dataset in the number of packets or flows

HTTPS traffic in the process or a VM doing the same is not that large. Although the duration of a page visit and the time between visiting new pages is different, the data itself is fairly the same.

Both datasets use Alexa's top 10k websites for browsing the Internet. Although Alexa's top 10k websites may not be representative of the entire Internet, it gives a good impression of the normal browsing activity. The CTU dataset has an additional real capture of university offices which also covers web pages outside Alexa's list. Except for the captures of the university offices, the datasets are fully artificially created. It is likely there exist other types of HTTPS traffic than web traffic more similar to DoH which is not present in the dataset.

Both datasets are, with close to a million network flows each, large in volume. The benign DoH class is relatively small, in total roughly 2 percent, but that still represents approximately 50k flows.

HTTPS traffic not related to web traffic produced by browsers is not present in the datasets. HTTPS traffic generated by applications such as Slack can be added. Although the researchers of the CTU dataset tried to compensate for this shortcoming by manually adding traffic produced by Instant messaging clients, it is unclear how large the volume of this traffic is.

The malicious use of DoH in the CIC dataset adds value to the dataset. It covers DoH tunnelling. However, there are more ways to abuse DoH, for instance in a botnet, DoH can be used to resolve the address of the C2 server (explained in Section 2.4). Other forms of malicious use of DoH are not covered in the dataset.

In conclusion, there is data that can make the dataset more complete. To be precise, HTTPS traffic from other applications than browsers and other types of malicious DoH usage can be added. It is unclear how well the data of the (largely) artificial data relates to real data. However, the methods used are not uncommon and the approach is probably as good as it gets if capturing real data is not an option.

5 Methods

This section describes a series of methods that will be employed to answer the two research questions states in Section 1.

The first research question is about what features can be used to detect DoH network traffic. Therefore, the features selected in previous work are analysed. Section 5.1 describes the methodology for this analysis. The analysis will provide insights into the characteristics of DoH and malicious DoH traffic. First, the features that are analysed are listed. Secondly, the approach for the analysis is explained. The main step in the analysis is plotting the value distribution for each feature, to visualize the differences in characteristics of non-DoH, DoH and malicious DoH traffic. Apart from creating value distribution plots, the analysis also includes correlations matrices to assess whether strong relationships exist amongst the features.

The second research question is about applying unsupervised learning methods to detect DoH network traffic. Section 5.2 describes the methods used to apply both supervised and unsupervised learning techniques. Both techniques are applied to be able to compare the results. In the section, each step that is taken to get an adequate machine learning model is described. These steps include: preprocessing the data, selecting the machine learning model, tuning the hyperparameters and measuring the performance.

5.1 Features

One of the goals of the research is to learn how (malicious) DoH network traffic can be detected. An analysis of certain features, extracted from the network traffic, can give insights in the differences of the characteristics of DoH and non-DoH, and the differences of the characteristics of benign and malicious DoH. This analysis also forms the basis for selecting the features used for machine learning techniques. Therefore, this analysis is an essential part of the thesis. This section describes the approach of the feature analysis. First, the features that are analysed are listed and explained in subsection 5.1.1 and second, the approach for the analysis is described in subsection 5.1.2.

5.1.1 Analysed features

Since HTTPS traffic is encrypted, features relying on Deep Packet Inspection (DPI), cannot be used for the detection of DoH. Network flow-related features are used instead. Previous work has proven the value of network flow-related features [68], [41]. In the

previous work the features listed in Table 5 and 6 are used. In this thesis, the same features are used since in previous work, the authors obtained relatively good model performance with them and these features are easily available in the labelled datasets.

The features can be divided into different categories. For each category, it will be explained what is meant by the category and what features it contains. *Flow Statistics* is one of the categories containing features such as the duration of the flow and the number of packets sent or received in that flow. The category *Flow Bytes* contains features describing the number of total bytes sent and/or received. Furthermore, there is a *Packet Length* category containing statistical features about the packet lengths such as mean value or standard deviation. The same holds for the *Packet Time* and *Inter-Packet Delay* categories. Note that *Packet Time* is the absolute packet time since the first packet of the flow whereas *Inter-Packet Delay* notes the time between sending a packet and receiving the response packet, or vice versa. In addition, the category *Bursts* contains features about bursts and pauses. A packet burst is detected as the inter-packet time is shorter than a predefined burst threshold. Since the inter-packet delay is dependent on many factors, the threshold is relative for each connection and is set to the 33,3% percentile of the inter-packet times per flow. Similarly, the pause threshold is set to the 66,6% percentile. The last category is *Other*, containing the autocorrelation feature and three symmetry features. The symmetry features note the symmetry between incoming and outgoing packet lengths.

5.1.2 Approach

The goal of the feature analysis is to better understand the data. Essentially, the differences between DoH and non-DoH traffic are interesting since these can be meaningful for understanding classification by machine learning. The same holds for the differences between benign and malicious usage of DoH.

The feature analysis consists of two parts. Firstly, a Kernel Density Estimation (KDE) plot is created and analysed for each of the features. Secondly, a correlation matrix is created for each dataset.

KDE plot The value distribution for each feature is plotted using a Kernel Density Estimation plot. A KDE plot, introduced by Silverman [61], is similar to a histogram, however, the KDE plot shows the estimation of the probability density function of a variable instead of discrete bins. A KDE plot shows the density at the y-axis whereas a histogram shows the frequency/count. Histograms could also be used, but this also

Category	Feature
Flow Statistics	Duration
	Number of outgoing packets
	Number of incoming packets
	A ratio of the number of Incoming and outgoing packets
Flow Bytes	Total number of outgoing bytes
	Total number of incoming bytes
	A ratio of the number of Incoming and outgoing bytes
Packet Length	A variance of Incoming Packet Sizes
	A variance of Outgoing Packet Sizes
	Average of Incoming Packet sizes
	Average of Outgoing Packet sizes
	The median value of Incoming Packet sizes
	The median value of outgoing Packet sizes
Inter-Packet Delay	Minimal Inter-Packet Delay
	Maximal Inter-Packet Delay
	Average Inter-Packet Delay
Bursts	The ratio of bursts and pauses
	Number of bursts
	Number of pauses
Other	Autocorrelation
	Transmission symmetry in the 1st third of connection
	Transmission symmetry in the 2nd third of connection
	Transmission symmetry in the last third of connection

Table 5: Features used to detect DoH by Vekshin et al. [68]

Category	Feature
Flow Statistics	Duration
Flow Bytes	Number of flow bytes sent
	Number of flow bytes received
	Rate of flow bytes sent
	Rate of flow bytes received
Packet Length	Mean Packet Length
	Median Packet Length
	Mode Packet Length
	Variance of Packet Length
	Standard Deviation of Packet Length
	Coefficient of Variation of Packet Length
	Skew from median Packet Length
	Skew from mode Packet Length
Packet Time	Mean Packet Time
	Median Packet Time
	Mode Packet Time
	Variance of Packet Time
	Standard Deviation of Packet Time
	Coefficient of Variation of Packet Time
	Skew from median Packet Time
	Skew from mode Packet Time
Inter-Packet delay	Mean Request/response time difference
	Median Request/response time difference
	Mode Request/response time difference
	Variance of Request/response time difference
	Standard Deviation of Request/response time difference
	Coefficient of Variation of Request/response time difference
	Skew from median Request/response time difference
	Skew from mode Request/response time difference

Table 6: Features used to detect (malicious) DoH by MontazeriShatoori et al. [41]

involves optimizing the number of bins that best visualizes the data distribution and due to the large number of features for two datasets manual adjustments are time-consuming. Another advantage of the KDE plot is getting a line instead of bars, which makes it easier to compare multiple classes in one graph. A disadvantage of using KDE plots instead of histograms is that if in a histogram a high peak is surrounded by a low valley, that is smoothened out in a KDE plot. This occasionally results in a graph showing values that are not in line with reality, for example, an inter-packet delay of fewer than 0 seconds.

Correlation Matrix A correlation matrix shows which features are correlated with other features. In this thesis, Pearson's correlation is used [32]. For many machine learning techniques, selected features must not be strongly correlated [36]. Additionally, the correlation matrix can give new insights once a feature is surprisingly correlated with another.

5.2 Machine learning

In this research machine learning is used to classify the network traffic. Machine learning algorithms build a model based on training data, the trained model can be used to make predictions or decisions [8]. Using machine learning is a natural choice since machine learning techniques are powerful in finding patterns that can be used for classification.

In machine learning can be divided into three categories [8]. First, there is *supervised learning* in which the input for the algorithm contains the data samples together with the desired output (label). The algorithm trains a model in which the input is mapped to the output. Secondly, with *unsupervised learning*, the input for the algorithm only contains the data sample without the desired output. The algorithm can group similar data samples, called clustering. The third category is known as *reinforcement learning*, but this category is not relevant for this thesis.

In this work, both supervised learning and unsupervised learning method are applied. One of the objectives of this research is to compare supervised and unsupervised learning techniques, to learn how much classification performance is lost once unsupervised learning is applied instead of supervised learning.

In this section, the steps taken to apply machine learning are described. The following steps will be addressed, as illustrated in Figure 8. The process starts with the data from the datasets and ends with measuring an adequately trained model. The first step is selecting the features used for classification. Secondly, the data is preprocessed to remove

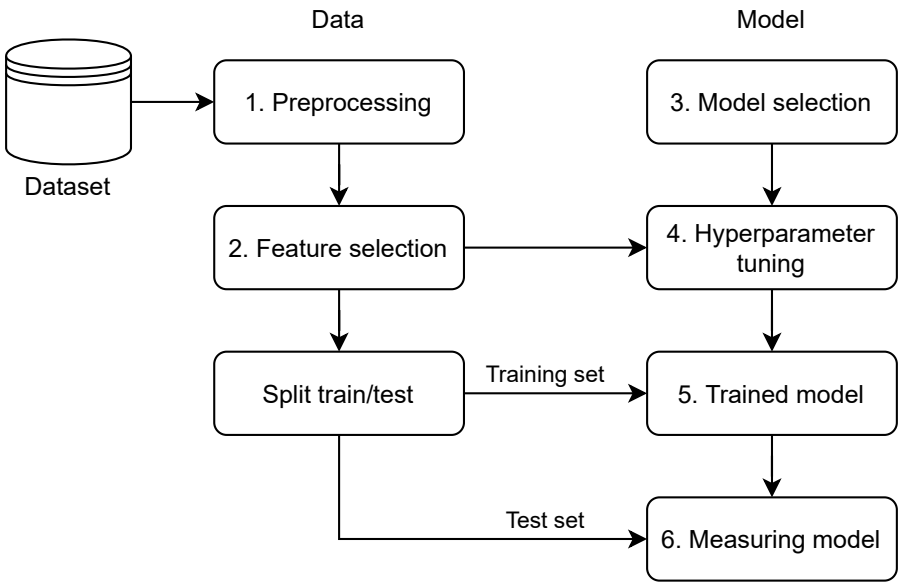

Figure 8: The process of applying machine learning

noise from the data. Thirdly, the machine learning model is selected. Fourth, to increase the classification performance of the model, the parameters are tuned. The fifth step is training the model and lastly, the classification performance of the trained model is measured.

5.2.1 Preprocessing

Most of the preprocessing work was already done by the creators of the datasets. One of the datasets contained a few rows with empty values which were discarded. The original datasets were ordered by label, to prevent the influence of the order on the results, the datasets were shuffled. In addition, the data was scaled using a Standard Scalar [59]. Scaling is important because many machine learning models require it, especially models that use the euclidean distance. Scaling prevents outliers to have a large impact on the model. The Standard Scalar scales the values per feature towards a normal distribution by subtracting the mean value of each sample and dividing the result by the standard deviation.

The last step, only applied before training a model, is splitting the dataset into a training set and a test set. The training set contains 80% of the shuffled dataset and the remaining 20% forms the test set.

5.2.2 Feature selection

This research is partially about which features are useful for detecting and classifying DoH and partially about applying Machine Learning. The feature selection step is relevant for both research questions.

The goal is to select the smallest subset of features with which still an accurate classification can be made. A small subset is preferred to avoid selecting redundant features. To achieve this goal, first, the right number of features is determined and afterwards, the right features were selected.

The right number of features is determined by applying Recursive Feature Elimination (RFE) [26]. In the process of RFE, first, all features are selected. Based on a ranking of the feature importance, the least important feature is removed from the set of features. By iteratively eliminating the features and measuring the performance of the model, the optimal number of features can be determined.

After determining the right number of features, the best subset of features needs to be selected. For selecting the best features, forward Sequential Feature Selection (SFS) is used [23]. Forward SFS start with an empty set and finds the feature that maximizes the accuracy score when the model is only trained with that feature. Iteratively, more features are added until the right number of features is reached.

Both run with 5-fold cross-validation, to minimize the variation in the results. Using k-fold cross-validation, the dataset is split into k parts and in k runs, each time another part is used as the test set and the other parts are used as the training set.

The reason to apply cross-validation is that although the data samples are shuffled, a certain subset of the dataset may contain a smaller or larger number of data samples that are easier to classify than average. k-fold cross-validation minimizes the effect of selecting a certain training set and test set since the effect is spread over k folds.

In RFE, in each round, the least important feature is eliminated. The machine learning classification model that is used for RFE should be able to order the features on the importance. Models that are based on decision trees are able to show the importance per feature. At each node in the tree, it can be computed how much a feature decreases the impurity [35]. Since some nodes are more often reached, the decrease of impurity is weighted by the probability of reaching that node. The importance per feature can then be determined by taking the total weighted decrease of impurity. Random Forest [53] is a model, as the name suggests, based on decision trees and thus suitable to use for RFE. Although there are more suitable models, random forest is used for RFE because

the random forest classifier scores well for this dataset in other studies [68], [41], [62].

The reason behind applying RFE first and SFS second is that this approach is computationally feasible. Since SFS iteratively greedily adds the best feature to the set of selected features, for selecting $i \in [1, 2, ..., n]$ features, for a dataset with n features, $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$ combinations are tried. That is computationally expensive.

By first applying RFE, the number of features that will be selected by SFS can be kept small. RFE itself is computationally cheap since each round requires training just one model (k models for k-fold cross-validation) and the number of rounds is only limited to the number of features.

5.2.3 Model selection

More often than not, labelled data is scarce. Therefore, it is useful to look further than only supervised learning methods since supervised methods rely on labelled data. Unsupervised learning methods, however, do not need labels for training. For classification, a limited number of labelled data samples are required.

As stated in Research Question 2, in this thesis, supervised methods detecting DoH will be compared with unsupervised learning methods. Therefore, both supervised models and unsupervised models were selected.

The models were selected manually. The following models were selected: k-nearest neighbours (kNN) [54] and k-means [55] clustering. As the supervised learning method, kNN is selected. It is a simple and powerful classification model.

Classification using kNN works as follows [54]. When a new, unknown data sample needs to be classified, the already known labels of the k nearest neighbours will be inspected and the new data sample will be classified according to the plurality vote of the neighbours' label. This requires a properly labelled dataset, which is available.

K-means is selected as the unsupervised model because it is a simple clustering model. The idea of k-means is that the data samples are divided over k clusters with as centre of a cluster, called centroid, the mean value of all data samples in that cluster [55]. Finding suitable centroids happens in three steps. The first step is choosing initial centroids, one can use k random data samples as initial centroids. Second, the data samples will be assigned to the nearest centroids. Third, the position of the centroids will be adjusted by taking the mean value of all data samples assigned to the centroid. The second and third steps are repeated until no more significant changes are made.

5.2.4 Hyperparameter tuning

A machine learning model learns from the training data, however, a model can have hyperparameters that are not learned by training. For example, the k-value for kNN is a hyperparameter that needs to be configured before the model can be trained.

For finding the optimal hyperparameters, resulting in the best classification, an exhaustive search is used. The python library scikit-learn [48] implements a `GridsearchCV` [56] function that is used for the exhaustive search. In the search, different models are trained covering all (manually) preconfigured parameter values. Each model is tested after training and in the end, the optimal hyperparameter configuration is shown.

For the kNN model, the number of neighbours is tuned. For the number of neighbours, the values 3, 4 and 5 were configured. The search was done with 5-fold cross-validation to be less susceptible to outliers.

For the k-means model, the number of clusters is tuned. However, this is not a straightforward process. A common approach is the elbow method [43]. In this method, the cost function (or error function), which is computed by the sum of squared distances of each data sample to their centroid, is plotted against the number of clusters. By increasing the number of clusters, naturally, the distance between the centroids and the data samples decreases. Usually, when increasing the number of clusters, at first, the value of the cost function decreases fast while later on the cost only decreases slowly. With the elbow method, the point where the fast decline of costs turns into a slow decline is the point of interest and the corresponding number of clusters is used.

The k-means model gives as output a number of clusters, thus, to get a classification, an extra step is required. This step is explained in Section 5.2.5. Since with this step, a classification can be obtained by the k-means model, the cost function is not used, the accuracy is used instead. By plotting the accuracy against the number of clusters, the appropriate number of clusters can be selected using the elbow method.

5.2.5 Measuring

Once the model is trained, it is interesting how well it performs. By measuring several metrics, the classification performance can be compared with previous work. In this thesis, the scores such as accuracy, precision, recall and F1-scores are noted since these are directly comparable with other studies. Additionally, confusion matrices and ROC curves are reported. These give insights into the strong and weak points of the classifiers, it shows which classes are often misclassified.

The following metrics were measured: accuracy, precision, recall and F1-score. The metrics are computed as:

$$Precision(P) = \frac{True\ Positive}{True\ Positive + False\ Positive}$$
$$Recall(R) = \frac{True\ Positive}{True\ Positive + False\ Negative}$$
$$F1\text{-}Score = 2 \cdot \frac{P \cdot R}{P + R}$$

where *True Positive* means a DoH sample is classified as a DoH and *False Positive* means a non-DoH sample being classified as DoH in case of classifying DoH versus non-DoH. In the case of classifying benign DoH versus malicious DoH, *True Positive* means malicious DoH classified as being malicious and *False Positive* means benign DoH classified as being malicious. The scores are measured by taking the average of a 5-fold cross-validation.

Additionally, confusion matrices [57] are generated. A confusion matrix shows the classification results in a table layout. The rows represent the actual class while the columns represent the predicted class (vice versa is also possible). The table can give an overview of classes that are often misclassified. 5-fold cross-validation is used and the average values of 5 folds are presented.

Last, ROC curves [58] were plotted to get more insights into how the true positive rate relates to the false positive rate. For the detection of DoH, it is important to have a low false-positive rate because, in a real-world scenario, that leads to blocking too much network traffic. The ROC curves are also plotted with the average of 5-fold cross-validation.

k-means For the k-means model, it is not straightforward to compute the standard machine learning metrics. Predicting based on the k-means model results in a number of clusters containing certain data samples. Based on these clusters, a classification is made by looking at the majority of the true label. In case the majority of the true labels is DoH, the whole cluster is counted as if it were classified as DoH (and vice versa for the non-DoH data). Based on this classification, the scores can be computed.

The same technique is used to plot the ROC curve for the k-means. However, instead of simply looking to the majority class based on the true label, a threshold is added. The threshold t gives a weight to each class. If a cluster, for example, contains 100 data samples of which 40 of class 0, normally, with $t = 0.5$, the cluster would be classified as being class 1. However, with $t = 0.7$, the 40 samples in class 0 have a weight of $0.7 \cdot 40 = 28$ whereas

the 60 samples in class 1 have a weight of $(1 - 0.7) \cdot 60 = 18$ meaning the cluster would be classified as being class 0. The class for each cluster is determined using the formula in equation 1, where $|\text{class}_x|$ notes the number of elements in the cluster having class_x as the true label. For each t value from 0 to 1 in steps of 0.001, the true positive rate and the false positive rate is computed. All the rates together form the ROC curve.

$$class = \begin{cases} 0 & t \cdot |\text{class}_0| > (1 - t) \cdot |\text{class}_1| \\ 1 & otherwise \end{cases} \tag{1}$$

5.3 Error Analysis

After training the machine learning models, an error analysis is performed. In the error analysis, the classification results are scrutinized to find out why certain data samples are misclassified. Finding patterns in the misclassified data samples gives a deeper understanding of which kind of samples are often misclassified.

For the supervised learning model, per feature, a histogram is plotted, visualizing the amount of true positive, false positive, true negative and false negative samples. This gives insights into what areas are easy or hard to classify.

In a histogram, only a single feature can be shown, thus, to get a more detailed perspective, also scatter plots are created. The scatter plots visualize one feature on the x-axis and another feature on the y-axis.

For the unsupervised learning model, an intermediate step was taken since a data sample is not directly classified but only clustered by the k-means model. Based on the clusters, a classification is made, as explained in Section 5.2.5. Therefore, first, the clusters are analysed and secondly, the content of clusters is inspected. The analysis of the clusters consists of looking at the size per cluster and examining the amount of misclassified samples in a cluster.

Next, the cluster containing the most misclassified samples are analysed in-depth. Both a cluster classified as DoH and a non-DoH cluster are selected. The approach of the in-depth analysis is similar as the analysis of the supervised learning model. First histograms are generated and to get a more detailed view, scatter plots are used.

The analysis is only focused on the dataset having the worst classification scores. Analysing all datasets, CTU, CIC Layer 1 and CIC Layer 2 for both supervised and unsupervised learning is time-consuming. The dataset with the worst classification scores, being CIC Layer 1, has the most room for improvement and is, therefore, the most interesting.

6 Results

As there are two research questions, the results chapter consists of two parts. First, the results related to the first research question, about which features can be used for the detection of (malicious) DoH, will be shown and discussed. These results cover the feature analysis and largely include the value distributions per feature. Apart from that, the correlation matrices will be shown and discussed.

The second research question involves detecting DoH and malicious use of DoH using unsupervised learning methods. The second part of the results section covers the comparison of supervised and unsupervised learning. First, the results of the intermediate steps of getting an adequate machine learning model will be shown. This includes selecting features using an automated systematic approach, tuning the hyperparameters and showing that the models are appropriately trained, the learning curves will be shown. Next, the models are evaluated and the metrics of the performance of supervised and unsupervised learning models will be compared. These metrics include accuracy, recall, precision, F1-score, confusion matrices and ROC curves.

6.1 Feature analysis

To learn the differences in characteristics of DoH and non-DoH, a feature analysis is performed. In this section, the results of this analysis will be shown and discussed. First, the plots of the value distribution of the most interesting features will be shown and discussed and secondly, the correlation matrices will be analysed.

6.1.1 Value distribution

Per feature for both datasets, the value distribution is plotted using a KDE plot (as explained in Section 5.1.2). The most interesting ones will be discussed. The features are ordered by the categories as shown in Table 5 and Table 6. For each feature, the value distribution of the CTU dataset can be compared with the value distribution of the CIC dataset for DoH against non-DoH, and it can be compared with the CIC dataset for benign against malicious DoH. However, some features are only present in one dataset and consequently, there will be no comparison for these.

Flow Statistics The duration of a network flow is a feature clearly distinguishing DoH from non-DoH for the CTU dataset. As shown in Figure 9, DoH flows have a longer

duration. Web browsers keep the HTTP/2 connection with the DoH resolver open for possible new requests resulting in flows with a long duration. Contrary, the non-DoH web traffic network flows have a short duration since the whole web page is fetched in only a few seconds. The same holds for the CIC dataset, DoH flows, in particular malicious DoH, have a longer duration than non-DoH flows.

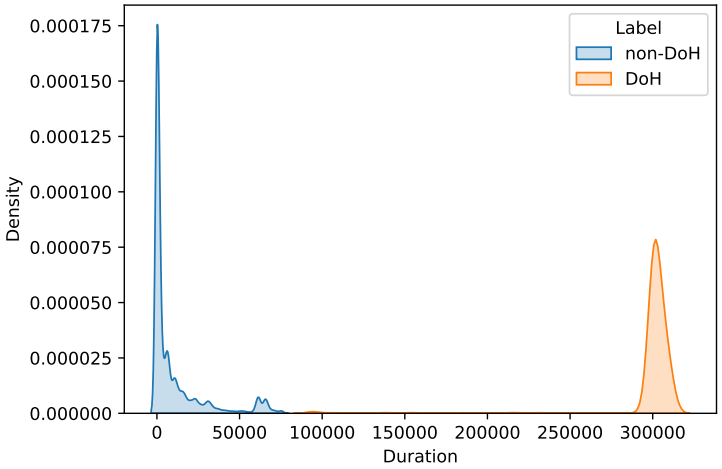

Figure 9: Value distribution of the duration feature of the CTU dataset

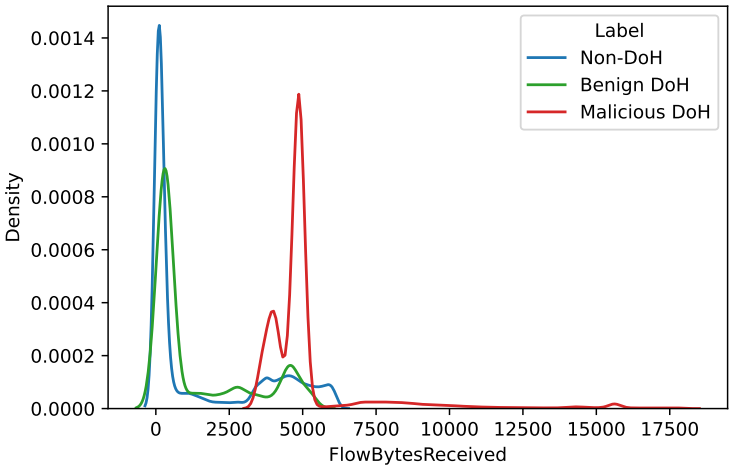

Figure 10: The value distribution of the flow bytes received feature of the CIC dataset

Another flow statistics related feature, only present in the CTU dataset, is the ratio of the number of incoming and outgoing packets. The figure showing the value distribution can be found in Appendix B. The ratio is close to 1 for DoH traffic, which is in line with the protocol, since a DoH request packet is answered by one, or in exceptional cases, two response packets. For non-DoH traffic, the ratio is more spread with values between 0.6 and 1.8.

The last two flow statistics, only present in the CTU dataset are the number of incoming and outgoing packets. The figures for the value distributions can be found in Appendix B. For both, the DoH flows contain more packets, for most of the flows between 20 and 60. Most of the non-DoH flows contain only between 0 and 25 incoming and outgoing packets. Thus, the amount of packets sent in a DoH is higher for most of the flows.

Flow Bytes For the CTU dataset, the bytes sent and bytes received features show a lot of overlap between the DoH and the non-DoH class (see Appendix B). The exact values for the DoH class are dependent on the number of DoH requests sent in a flow and for

the non-DoH class depending on the size of the web pages that are fetched. Both vary in the same range. However, the CIC dataset shows that the number of bytes sent or received for malicious DoH is clearly higher compared to non-DoH and benign-DoH. This is shown for the flow bytes received in Figure 40. It is a logical consequence that if tunnelling tools are used, meant to transfer additional data, the number of bytes sent and received increases for malicious DoH.

Contrary to the number of bytes sent and received, the ratio of incoming and outgoing bytes do not show much overlap between DoH and non-DoH (see Appendix B). The ratio is concentrated close to 1 for DoH flows. This is a result of DoH requests and responses having similar lengths. The ratio for non-DoH traffic is varying between 1 and 5 for most of the flows.

The value distributions for sending and receiving rates are similar for non-DoH, benign DoH and malicious DoH (see Appendix B). The rates are apparently not depending on the kind of network traffic.

Packet Length Similarly to the flow bytes sent and received, the mean packet length also shows that tunnelling involves larger packets. Figure 11 shows, despite the overlap between the classes, that malicious DoH flows clearly have greater mean packet lengths than benign DoH flows. Their overlap with non-DoH flows is only limited. As Figure 11 corresponds with the CIC dataset we can compare this to the CTU dataset. In the CTU features, the mean packet length is specified for incoming and outgoing packets. Similar to Figure 11, the incoming benign DoH packets have a mean length between 80 and 180 bytes whereas the mean packet length for non-DoH varies between 50 and 600 bytes (see Appendix B). It is interesting to notice the mean outgoing packet lengths are smaller for most DoH flows than non-DoH flows. This likely relates to TCP ACK packets.

Another packet length related feature is the variance of packet lengths. The variance of incoming packets is shown in Figure 12. In general, DoH flows have more consistent packet lengths, resulting in a smaller variance compared to non-DoH. The same holds for outgoing packets. The CIC dataset shows the same trends (see Appendix B), however the overlap between the classes is higher. An interesting difference between benign and malicious DoH is that the variance of malicious DoH is higher but not as high as some non-DoH flows. Generally, the variance of packet lengths for web traffic is website specific and can both be low or high, while the variance of benign DoH is relatively small as all packets are quite constant in length. The variance for malicious DoH is always relatively high due to alternating small and larger packets.

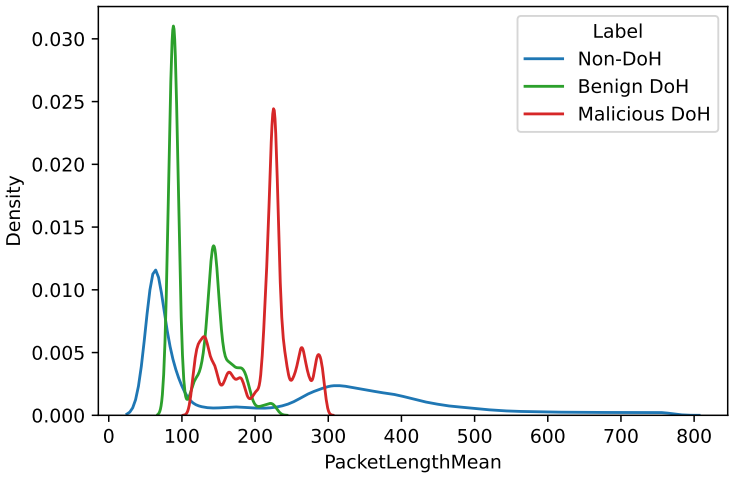

Figure 11: The value distribution of the mean packet length (bytes) of the CIC dataset

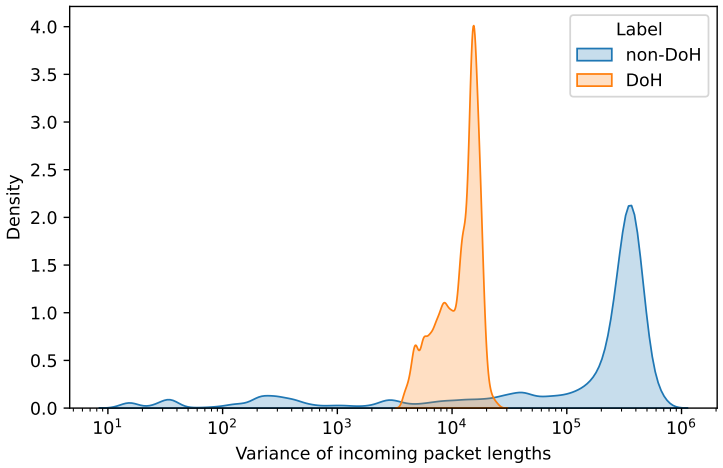

Figure 12: The value distribution of the variance of incoming packet lengths (CTU dataset)

The CIC dataset contains a packet length mode feature, which almost perfectly distinguishes malicious DoH from the other two classes. The most frequent packet length corresponds with the TCP ACK message. Although these messages should be equal in length for all classes, the malicious DoH ACK messages are 2 byes longer. Deep packet inspection showed that the Ethernet frame of malicious DoH has two additional bytes, which is a result of the capturing method [1]. Using this feature for classification results in getting an artificial advantage.

The last packet length related feature is the packet length median (see Appendix B). Surprisingly, the non-DoH flows are concentrated at the lowest median value, malicious DoH comes second and benign DoH third. This is not in line with what one can expect based on the packet length mean shown in Figure 11. Although benign DoH has a lower mean packet length, the median value is higher compared to malicious DoH. Malicious DoH flows contain larger packets on average, however, the flows also contain a lot of smaller packets which results in a lower median value.

Packet Time The packet time denotes the absolute packet time since the start of a flow. The packet time features are only present in the CIC dataset. The features relate to statistics such as packet time mean, median and mode. However, all value distributions are similar to the value distribution of the duration feature. The correlation between the

[1] `https://www.wireshark.org/lists/ethereal-users/200412/msg00314.html` wireshark - Ethereal-users: Re: [Ethereal-users] What is "Linux cooked capture" and why does it add 2 bytes?

duration and the packet time is not strange, since a flow with a higher duration, will also have higher packet time values. Therefore, it is redundant to show and discuss the figures.

Inter-Packet delay The inter-packet delay denotes the time between an outgoing packet is sent and a response packet is received. For the CTU dataset, the three inter-packet delay related features are the minimum, the average and the maximum inter-packet delay. All of them are similar to the duration feature. That makes sense since the maximum inter-packet delay of a flow with a higher duration is likely to be higher as well (and vice versa). That makes it redundant to show and discuss figures for these features. The correlation between duration and inter-packet delay related features is less strong for the CIC dataset, however, all plots show substantial overlap between the non-DoH, benign-DoH and malicious DoH classes. Since the inter-packet delay mean, median and mode values are smaller than 2 hundredths of a second, it seems like the inter-packet delay is correlated with the round-trip time. That would also explain the substantial overlap. The substantial overlap reduces the relevancy and therefore, the figures will not be shown and discussed in detail.

Other The CTU dataset has various features that do not fit in the categories above. The first one is the number of bursts during a flow. A burst is defined as: inter-packet time shorter than a predefined burst threshold, which is configured as the 33.3% percentile from the inter-packet times of a flow. Most of the non-DoH flows contain less than 20 bursts while most of the DoH flows contain at least 20 bursts. The number of bursts for DoH flows is dependent on the number of DoH requests that are made during the connection.

To measure the periodicity of the traffic, the autocorrelation feature is used. However, the value distribution of the autocorrelation feature of DoH traffic substantially overlaps with the non-DoH traffic.

The last three features used in the CTU dataset are the symmetry of the amount of incoming and outgoing data for the first, second and third part of a flow. The idea is that DNS requests/response packets have similar lengths whilst web traffic is more unbalanced. However, the value distributions show serious overlap.

Conclusions By analysing each of the features for both datasets, it becomes clear which features are relevant for the detection of DoH and the detection of malicious DoH and

Category	Feature(s)	Overlap DoH / non-DoH *	Overlap benign / malicious DoH *, **	Useful?
Flow Statistics	Duration	0	0	Yes, browsers keep connection with DoH server open whilst a web page is fetched in a few seconds
	Number of packets in/out	0	-	Yes, DoH flows consist out of more packets than non-DoH flows
	Ration packets in/out	1	-	Yes, DoH always has a ratio close to one due to the DoH protocol
Flow Bytes	Number of bytes sent/received	2	0	Useful for detecting malicious DoH since these flows contain more bytes
	Ratio bytes sent/received	1	-	Yes, DoH always has a ratio close to one
	Rate sending/receiving	2	2	No, too much overlap
Packet Length	Variance packet length in/out	1	1	Yes, benign DoH has a low variance and malicious DoH has a large variance
	Mean packet length in/out	1	1	Yes, benign DoH has a low mean packet length and malicious DoH an average mean value
	Median packet length in/out	0	1	Yes, clear seperation from small to large: non-DoH, malicious and benign-DoH
	Mode packet length	2	0	No, the values for malicious DoH do not overlap due to the capturing method
Packet Time	Mean, median, mode, etc.	1	1	Rather use the correlated Duration feature.
Inter-Packet delay	Min, average, max delay	0	-	Rather use the correlated Duration feature.
	Mean, median, mode, etc.	2	2	No, too much overap since it relates to RTT which is traffic independent
Other	Bursts	0	-	Yes, DoH flows consists of more bursts than non-DoH
	Autocorrelation	2	-	No, too much overlap
	Symetry	2	-	No, too much overlap

* scores can be 0,1 or 2. Meaning 0: <25% overlap, 1: >25% and <75% overlap, 2: >75% overlap

** can be empty because some features are not present in the dataset containing malicious DoH traffic

Table 7: Overview of the features and how useful each feature is for detecting (malicious) DoH

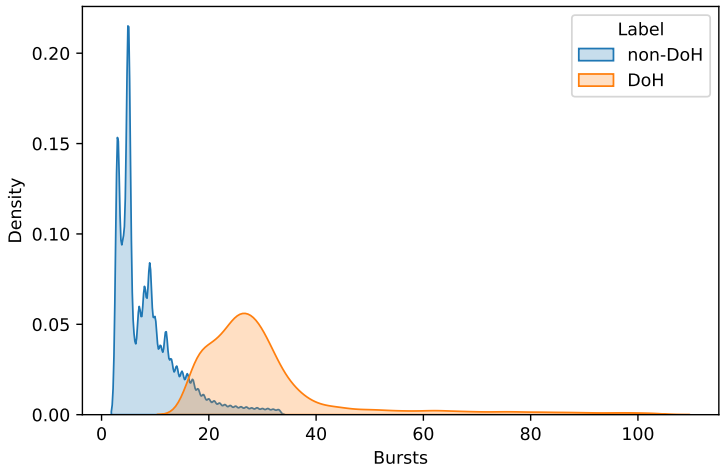

Figure 13: DoH flows contain more packet bursts than non-DoH flows

which features are less relevant. As shown in Table 7, the overlap between DoH/non-DoH and benign/malicious DoH in the value distribution varies per feature. Less overlapping features are naturally more useful.

The features most relevant for the detection of DoH are the duration of a flow, ratios between incoming and outgoing traffic and packet length related features. The duration of a flow is relevant since, as described in the RFC, an HTTP/2 connection can be kept open for multiple DoH requests, resulting in flows with a long duration. The ratios between incoming and outgoing traffic are interesting because both the number of packets and the length of the packets are remarkably similar for incoming and outgoing DoH traffic. Finally, the packet length related features are interesting because DoH flows consist of many small packets with a low variance in packet lengths.

For the detection of malicious use of DoH, the following features are most relevant: duration of a flow, number of bytes sent or received and packet length related features. The duration of a flow is interesting since DoH tunnels use connections with a DoH resolver that are kept open for a while. Due to the additional data that is sent over a DoH tunnel, the number of bytes sent and received is notably higher compared to non-DoH and benign DoH flows. Last, the packet related features are relevant since the length of each packet that is sent over a DoH tunnel is larger due to the additional data that is transmitted.

The packet time and inter-packet delay related features are less relevant. The packet time-related features do correlate with the duration feature. Similarly, several inter-packet

delay related features correlate with the duration feature as well. Others relate to the round trip time which is traffic independent and therefore not useful for the detection of (malicious) DoH.

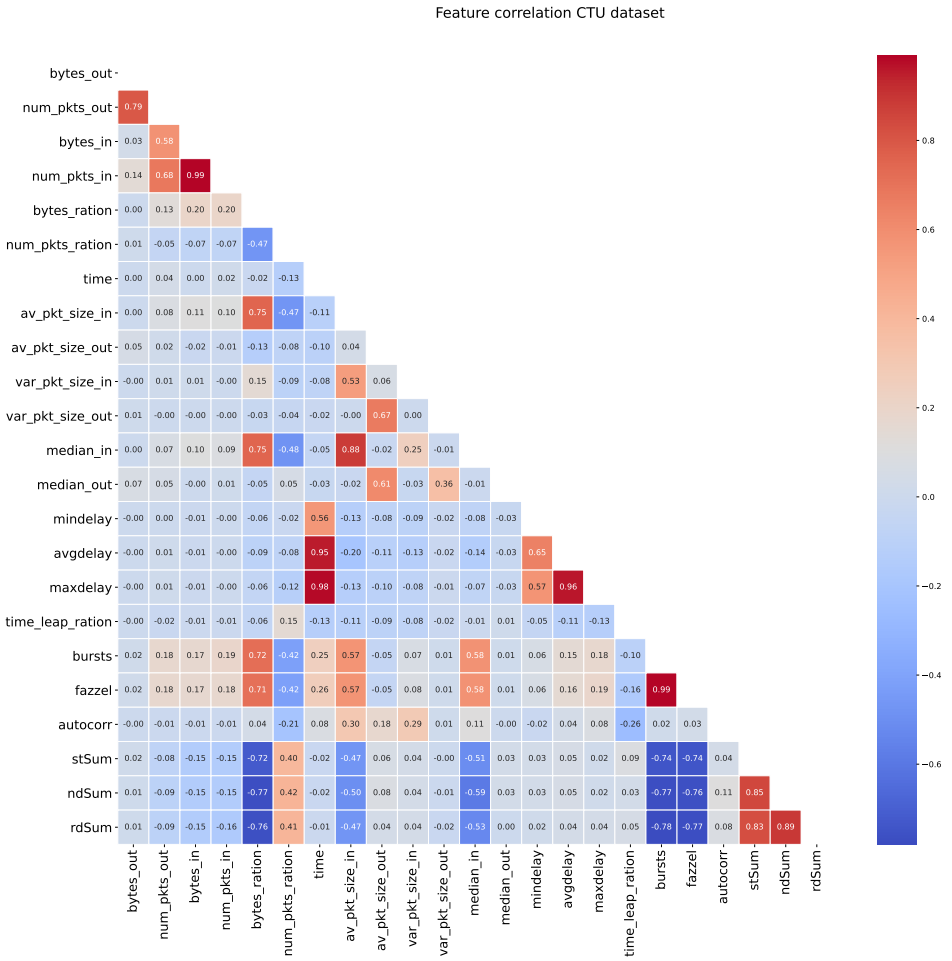

Figure 14: The correlation matrix of the CTU dataset

6.1.2 Correlation matrix

The correlation matrix is useful to identify correlations between certain features. Once machine learning is applied, it is often preferable to reduce the number of strongly correlated features [36]. For both datasets, the correlation matrices are created. Figure 14 shows the correlation matrix for the CTU dataset, Figure 15 show the correlation matrix for Layer 1 of the CIC dataset (non-DoH vs DoH) and Figure 16 shows the correlation

matrix of Layer 2 of the CIC dataset (benign DoH vs malicious DoH). For each matrix, the most interesting points will be discussed.

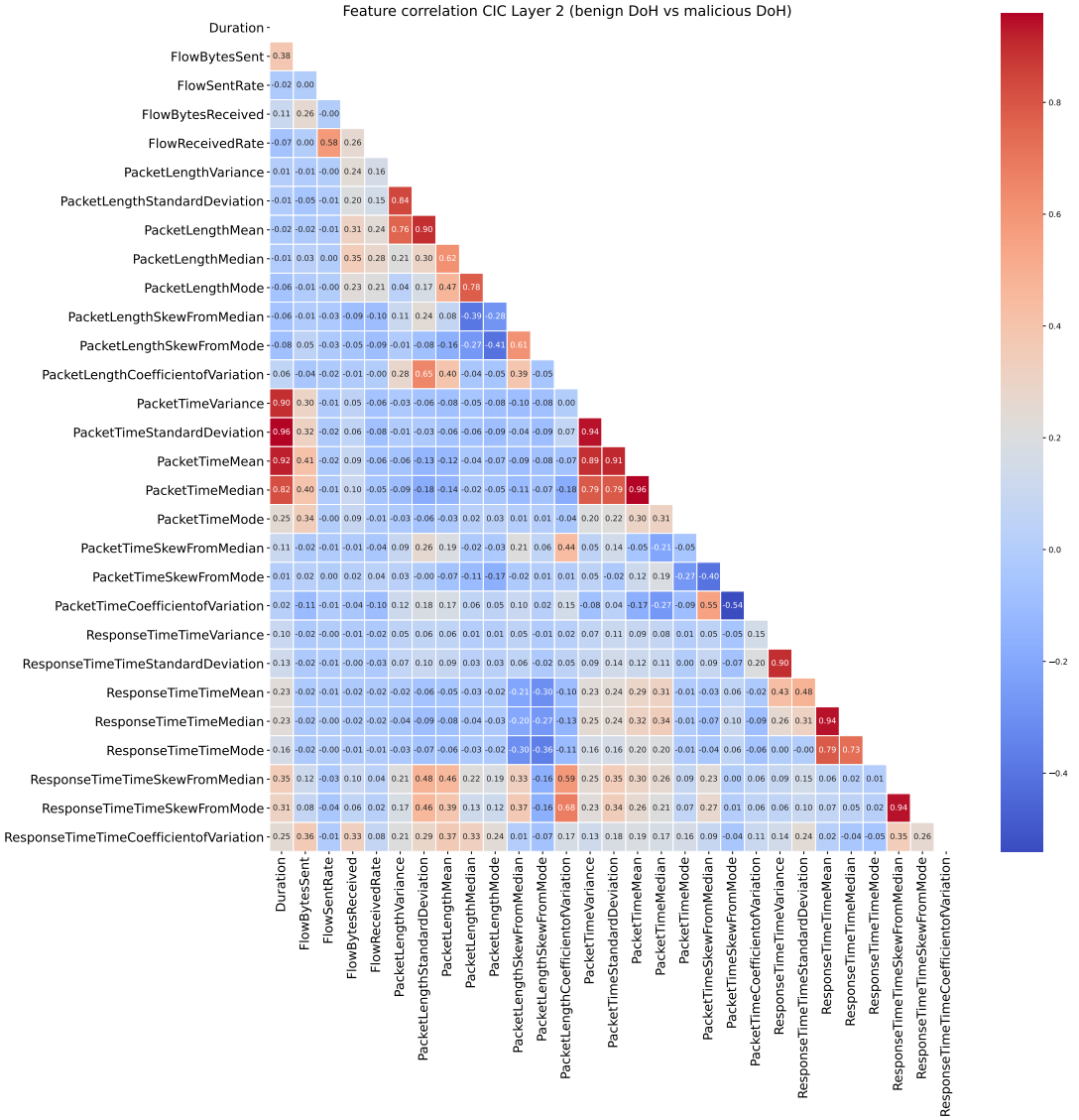

Figure 15: The correlation matrix of Layer 1 (non-DoH vs DoH) of the CIC dataset

The CTU correlation matrix shows blocks of correlated features. First, the incoming flow bytes and the number of incoming packets are naturally correlated. Similarly, for the outgoing bytes and packets. The number of incoming packets is also correlated with the number of outgoing packets. That is not strange since the network traffic follows

request/response patterns. Second, the flow duration (time) is correlated with the minimum, average and maximum inter-packet delay. Naturally, a short flow can only have a short minimum inter-packet delay and likewise, a longer flow has a higher maximum inter-packet delay. The third and last block identified involves the average packet length of incoming packets and the variance of incoming packets. These are correlated, as expected. The same holds for the incoming packets. The median packet length is also correlated with the average packet length and so are the median and average incoming packet lengths with the bytes ratio. Larger incoming packets results in a higher bytes ratio.

Layer 1 of the CIC dataset has one block of correlated features and certain patterns of correlated features (see Figure 15). The block of correlated features involves the duration feature which is correlated with Packet Time related features. It is plausible that the value of packet time related features increase as the flow has a longer duration. Apart from that, patterns of correlated statistical properties can be identified. Most natural is the correlation between the standard deviation and the variance. For each category of features, these two properties are correlated. The same holds, to a lesser extent, for the mean and the standard deviation, which is most noticeable for the packet length. The mean shows correlations with the median as well.

For many features, the correlation matrix for Layer 1 of the CIC dataset is similar to the matrix for Layer 2. However, there is one noticeable difference. The correlation between flow bytes sent and received is much stronger in Layer 2 (0.81 instead of 0.26). Thus, the difference in the number of flow bytes sent and received is smaller for benign or malicious DoH. That makes sense since for non-DoH, a few small HTTP requests are replied to in the form of a relatively large web page. For DoH, the packet length of a DoH query and a DoH response does not differ that much.

6.2 Machine learning

6.2.1 Feature selection

To select the most relevant features, first Recursive Feature Elimination (RFE) is applied, to find the right number of features. Afterwards, Sequential Feature Selection (SFS) is used to find the most relevant features. As explained in Section 5.2.2, RFE starts with a set containing all features and in each round, the least important one is removed from the subset. SFS is rather the opposite, it starts with an empty set and in each round, the feature adding the most value (i.e., resulting in the highest accuracy) is added to the set.

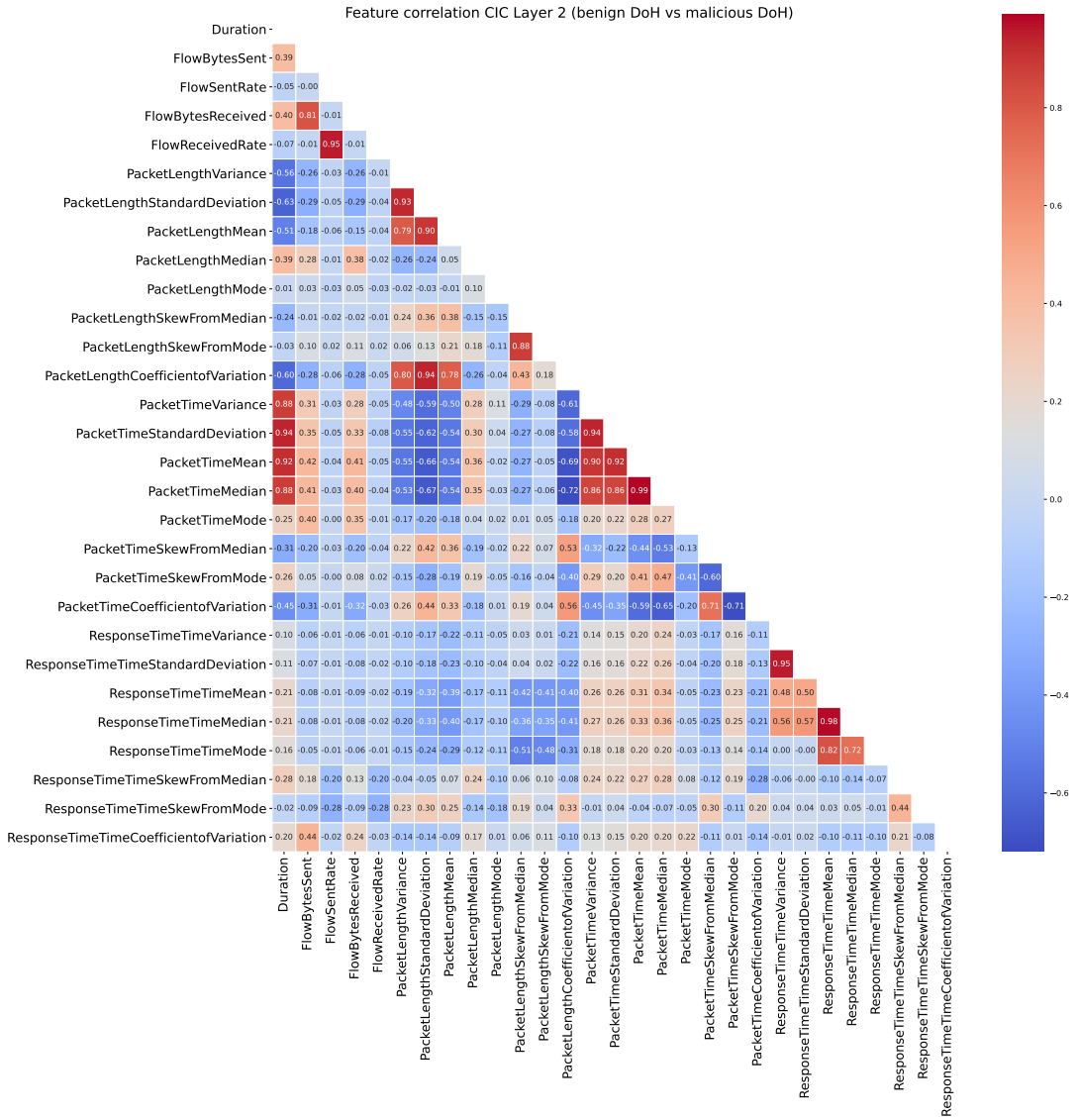

Figure 16: The correlation matrix of Layer 2 (benign DoH vs malicious DoH) of the CIC dataset

The results of each step will be described for the CTU dataset, the first layer of the CIC dataset and for the second layer of the CIC dataset.

RFE Figure 17 shows the model accuracy scores after each round of feature elimination for the CTU dataset. All scores are extremely high, using only a single feature results in an accuracy of over 99 percent. With six or more features selected, the scores are relatively

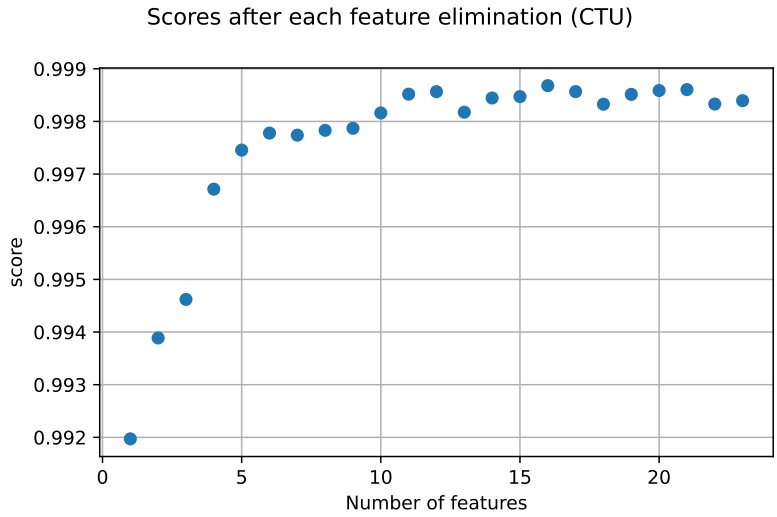

Figure 17: Only the six most important features increase the accuracy score of the machine learning model

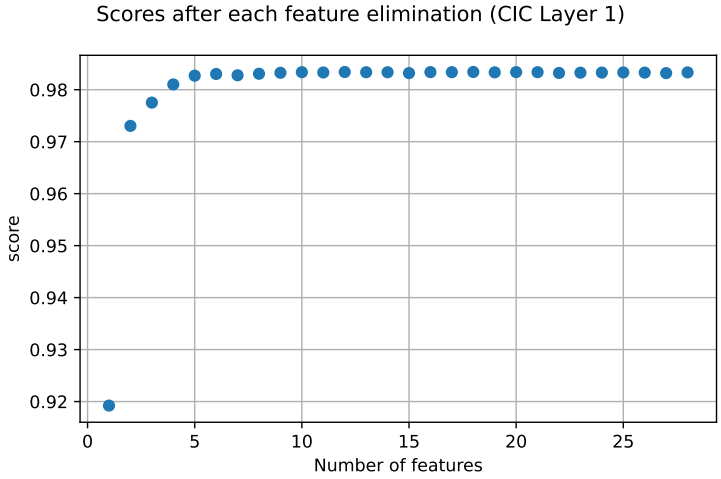

Figure 18: Only the five most important features increase the accuracy score of the machine learning model

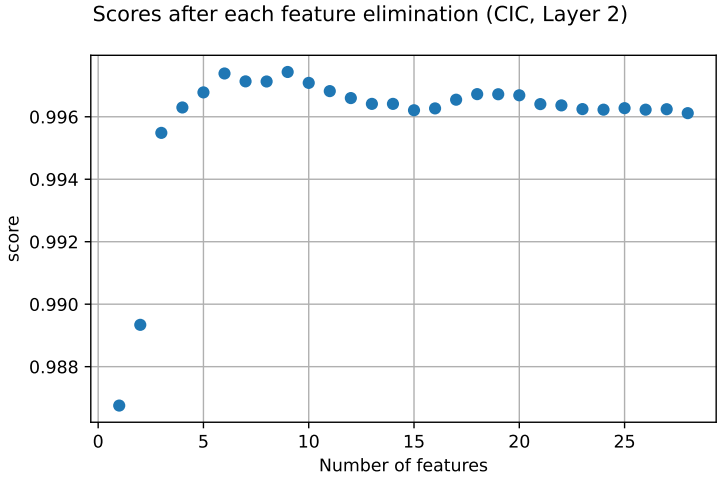

Figure 19: Only the six most important features increase the accuracy score of the machine learning model

stable. Arguably, one feature would already be enough. Adding more than one feature to the selection only improves the accuracy a little.

For the CIC dataset, Layer 1, the RFE plot is shown in Figure 18. For this dataset, the scores after eliminating features are stable with five or more features. Therefore, for this dataset, five features are selected.

The RFE plot for CIC Layer 2 has an absolute top at exactly six features, thus, six

features are selected for this dataset.

Sequential Feature Selection With the right number of features obtained by the RFE method, the SFS method is used to determine which specific features maximize the accuracy score. The six features selected by SFS for the CTU dataset are: num_pkts_ration, time, var_pkt_size_out, median_in, median_out and bursts. The selected features are in line with what one could expect based on the observations made during the manual feature inspection in Section 6.1.1 (summarized in Table 7).

Selecting the *time* (duration of a flow) feature is intuitive since DoH flows have a clearly longer duration than usual web traffic. The *num_pkt_ratio* feature is not remarkable either, because due to the DoH protocol, the packet ratio is always close to 1 whereas the ratio of web traffic varies. Selecting the *bursts* feature is not a surprise, since DoH flows contain multiple bursts when new DoH requests are sent. Selecting the last three features, *var_pkt_size_out*, *median_in* and *median_out* is also not strange. DoH flows mainly consists of three types of packets, namely, DoH requests, DoH responses and TCP packets. The packets are small and within the category, packets have similar lengths. This results in a low variance and typical median values.

Although the features that are selected are not surprising, it is somewhat counterintuitive that a number of features are not selected. The number of packets sent and received, the number of bytes sent and received, the mean packet size of incoming and outgoing traffic are all features that are suitable candidates according to the manual feature inspection. However, a high accuracy could already be achieved using only six features.

Similar to the CTU dataset, SFS is used to determine the features that maximize the accuracy score for the CIC dataset as well. Table 8 shows the features for both Layer 1 and Layer 2. The selected features are different from what one would expect based on the manual inspection. First, the selected features for Layer 1 will be discussed and afterwards the features for Layer 2.

From the selected features for Layer 1, the PacketLengthMean and PacketLengthMedian are also recommended after manual inspection. However, during manual inspection, it turned out that the flow duration is a feature clearly distinguishing DoH from non-DoH and instead of selecting that feature, the ResponseTimeTimeMedian feature is selected. Besides, the features FlowBytesSent and FlowBytesReceived are selected while the value distributions for these features show more than 75% overlap between the DoH and non-DoH class.

For Layer 2, again, the selection of some features correspond to the expectations based

Feature	CIC Layer 1	CIC Layer 2
Duration	-	-
FlowBytesSent	✓	-
FlowSentRate	-	✓
FlowBytesReceived	✓	-
FlowReceivedRate	-	✓
PacketLengthVariance	-	-
PacketLengthStandardDeviation	-	-
PacketLengthMean	✓	-
PacketLengthMedian	✓	✓
PacketLengthSkewFromMedian	-	-
PacketLengthSkewFromMode	-	-
PacketLengthCoefficientofVariation	-	-
PacketTimeVariance	-	-
PacketTimeStandardDeviation	-	-
PacketTimeMean	-	-
PacketTimeMedian	-	-
PacketTimeMode	-	-
PacketTimeSkewFromMedian	-	-
PacketTimeSkewFromMode	-	-
PacketTimeCoefficientofVariation	-	-
ResponseTimeTimeVariance	-	✓
ResponseTimeTimeStandardDeviation	-	-
ResponseTimeTimeMean	-	-
ResponseTimeTimeMedian	✓	✓
ResponseTimeTimeMode	-	✓
ResponseTimeTimeSkewFromMedian	-	-
ResponseTimeTimeSkewFromMode	-	-
ResponseTimeTimeCoefficientofVariation	-	-

Table 8: Features selected by the Sequential Feature Selector for the CIC dataset

on the manual feature inspection and the selection of other features are not. Selecting the PacketLengthMedian feature was to be expected since the value distribution of this feature clearly separates malicious and benign DoH. However, the features FlowSentRate and FlowReceiveRate are selected, although the value distributions of these features show much overlap between malicious and benign DoH. The last two selected features, ResponseTimeTimeMedian and ResponseTimeTimeMode are also not expected due to overlap between benign and malicious DoH in the value distributions.

Based on SFS, unexpected features were selected for the CIC dataset. Features that

Dataset	Neighbours	Distance
CTU	4	Euclidean
CIC Layer 1	3	Manhattan
CIC Layer 2	3	Manhattan

Table 9: Hyperparameters per dataset for the k-nearest neighbours model

show much overlap in their value distribution between either DoH/non-DoH or benign/malicious DoH. However, it is worthwhile mentioning that a much overlapping feature does not necessarily mean that the feature contains less information. It is possible that the non-overlapping part contains just the information that was not yet present in the other selected features. Also, note that for the CIC Layer 1 dataset, after selecting two features, the third feature barely improves the accuracy anymore (See Figure 18). Thus, the differences in the value that certain features add is very small. In other words, it is likely that selecting other features would not affect the accuracy significantly.

6.2.2 Hyperparameter tuning

Hyperparameters are tuned to improve the classification performance of the models. For both datasets, CTU and CIC Layer 1 and Layer 2, the models are tuned. Additionally, for both supervised learning and unsupervised learning, respectively the k-nearest neighbours model and the k-means model are tuned.

Supervised learning For the k-nearest neighbours, the following hyperparameters were tuned: the number of neighbours and the power parameter for the Minkowski distance. As explained in Section 5.2.4, a grid search has been performed. Table 9 shows the output of the grid search.

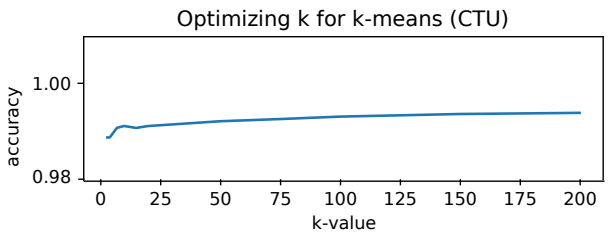

Figure 20: Accuracy per k-value

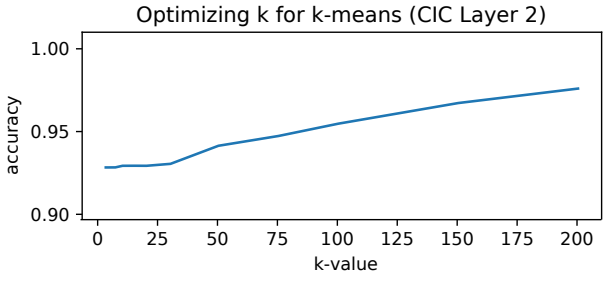

Figure 21: Accuracy per k-value

Unsupervised learning For the k-means model, the k-value is tuned. Figures are generated in which the k-value is plotted against the model accuracy. Figure 20 shows the accuracy per k-value for the CTU dataset. The accuracy is notably high for all k-values. Since the accuracy does not significantly increase by increasing the k-value, for the rest of this thesis, the smallest measured k-value of 3 is selected.

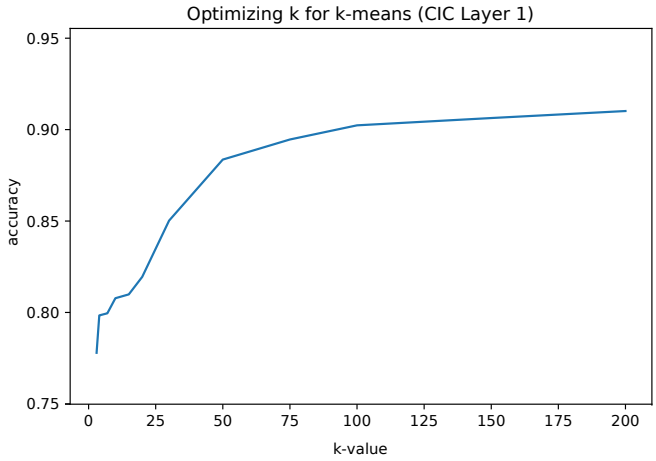

Figure 22: Accuracy per k-value

The accuracy per k-value for Layer 1 of the CIC dataset is plotted in Figure 22. The figure shows an upwards trend that eventually stagnates. Most classification performance, measured in accuracy, is gained by increasing the number of clusters from 3 to 50. By increasing the number of clusters even more, the accuracy still increases, although it is only a shallow curve. For the rest of the thesis, the k-value of 50 is selected.

For Layer 2 of the CIC dataset, the plot is shown in Figure 21. The accuracy slowly increases as the number of clusters increases. For the rest of this thesis, the smallest measured value 3 is used.

Finding the optimal k-value depends on the specific situation in which k-means is applied. Increasing the number of clusters does also increase the accuracy. However, the computation time and the required amount of labelled data increases as well, since for each cluster data is required to determine to classify the cluster as DoH/non-DoH or benign/malicious DoH. For two datasets a k-value of 3 is chosen, which means that only a small amount of labelled data samples for 3 clusters is required. For the CIC Layer 1 dataset, the k-value of 50 is chosen, which means that sufficient labelled data is required for all 50 clusters to use k-means for classification.

6.2.3 Training

To ensure there is no overfitting or underfitting and there is a sufficient amount of training data, the learning curves of the models are plotted.

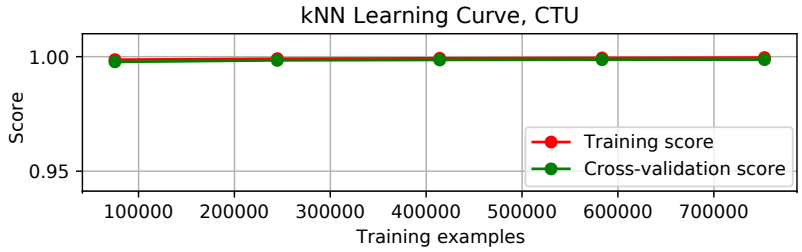

Figure 23: Learning curve of the k-nearest neighbours for the CTU dataset

Supervised learning Figure 23, shows the learning curve of the k-nearest neighbours model for the CTU dataset. Since the learning curves are rather similar, the learning curves for the CIC dataset can only be found in Appendix C.

All three figures show both the training and the test score are high. The fact that the cross-validation score becomes constant means there is no underfitting problem, there is enough training data. To be precise, Figures 23 shows that the model achieves its maximum accuracy when using only 100,000 training samples. The other 600,000 training samples are unnecessary for training the model. There is no large gap between the training and test lines, which shows that the model is not suffering from overfitting either [49].

Unsupervised learning Similarly, the learning curves for the k-means algorithm do not show signs of overfitting or underfitting. The curves can be found in Appendix C. For all three datasets, the smallest measured amount of training samples was already sufficient to fully train the model.

6.2.4 Evaluation

For the evaluation of the trained models, the confusion matrices, ROC curves and classification scores will be shown and discussed. The classification scores will be compared with the related work.

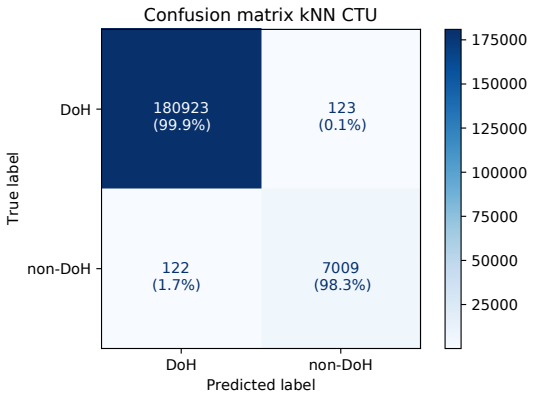

Figure 24: Confusion matrix for kNN CTU

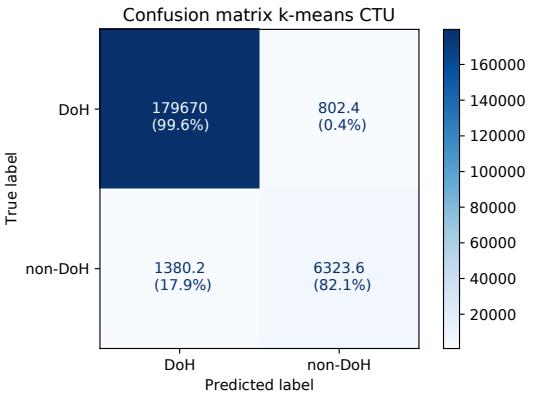

Figure 25: Confusion matrix for k-means CTU

Confusion matrix The confusion matrices for the CTU dataset are shown in Figure 24 for the kNN model and in Figure 25 for k-means model. The number of false positives (non-DoH classified as DoH) and false negatives (DoH classified as non-DoH) for the supervised kNN model is very small, with 0.01%. For the unsupervised k-means model, the number of false positives and false negatives is clearly larger, although both are still below 1%.

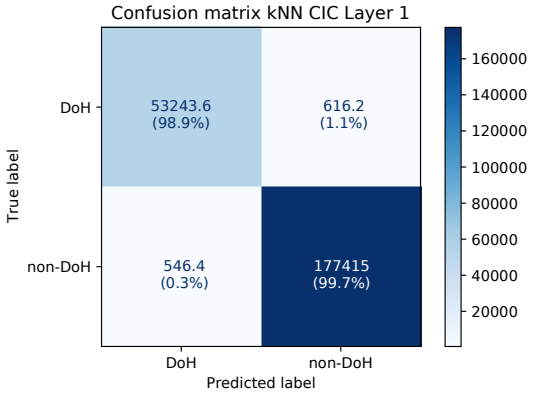

Figure 26: Confusion matrix for kNN CIC Layer 1

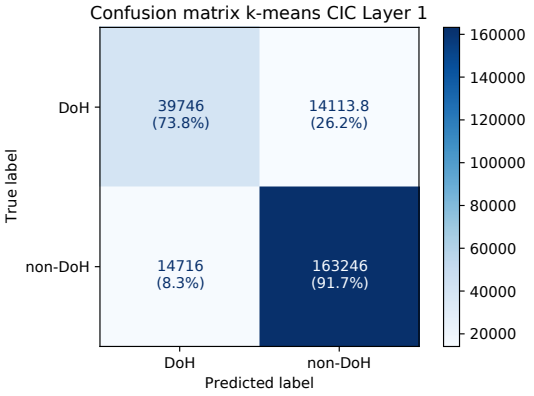

Figure 27: Confusion matrix for k-means CIC Layer 1

The confusion matrices for the CIC Layer 1 dataset are shown in Figures 26 and 25 for respectively the kNN model and the k-means model. The kNN model scores less than 1% false-positives and false-negatives. The k-means models scores less in all quadrants; both true DoH and true non-DoH are more often misclassified, however, DoH is still accurately detected.

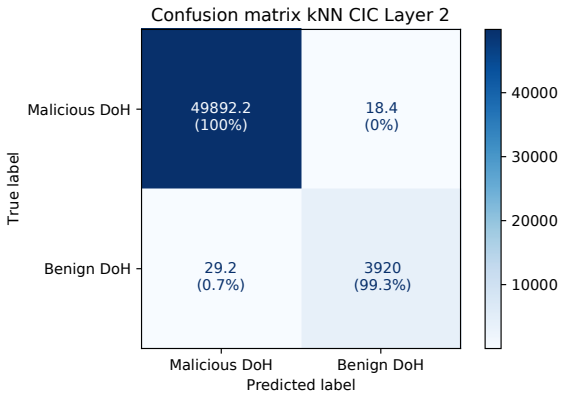

Figure 28: Confusion matrix for kNN CIC Layer 2

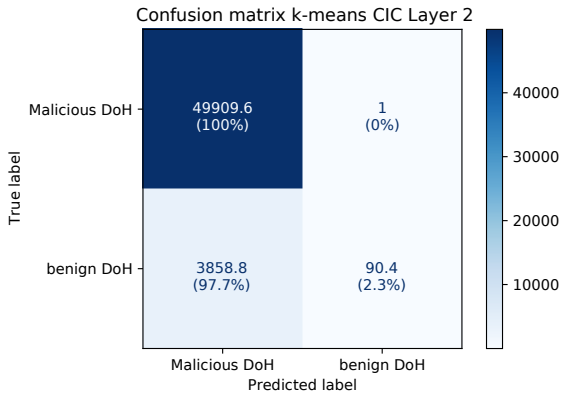

Figure 29: Confusion matrix for k-means CIC Layer 2

For the CIC Layer 2 dataset, Figure 28 shows the confusion matrix for the kNN model and Figure 29 shows the matrix for the k-means model. Where the kNN model scores well in all quadrants, the k-means model scores only well for the true malicious DoH samples. Almost all benign DoH samples are misclassified as being malicious.

ROC Figures 30 and 31 show the ROC curve for the CTU dataset for respectively the kNN and k-means model. As it turned out with the confusion matrices, the classification performance of supervised cannot be matched with unsupervised learning. However, Figure 30 shows that the k-means model is able to have a true positive rate of approximately 0.9 when having a very low false positive rate. In other words, approximately 90% of the DoH traffic is detected while the number of false positives is very small. Thus, when no labelled data is available, k-means can be applied to detect a vast majority of the DoH traffic. Note that a small number of labelled data samples is required to use k-means for classification.

Scores The scores are measured according to the methods described in Section 5.2.5. Using the scores, comparisons can be made with related work and the scores of kNN can be compared with the scores of the k-means model.

Table 10 gives an overview of all measured scores. For the CTU dataset, the k-means model achieves a F1-score of 0.85, which shows that the unsupervised learning method can be very useful for the detection of DoH. The k-means scores CIC Layer 2 are also very high. Therefore, the unsupervised k-means method can also be used for the classification of DoH traffic as either malicious or benign. The performance of the kNN method is a

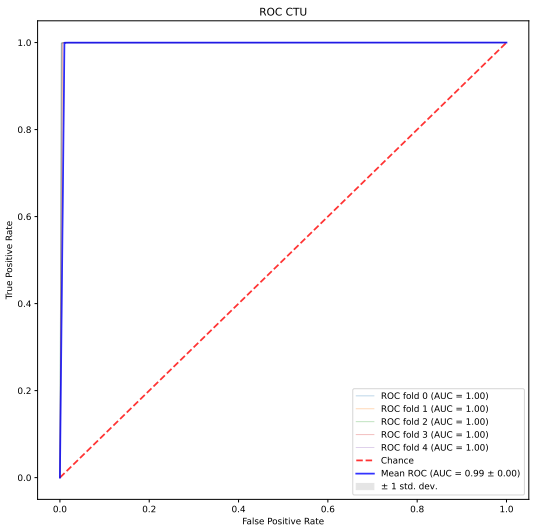

Figure 30: kNN ROC curve for the CTU dataset

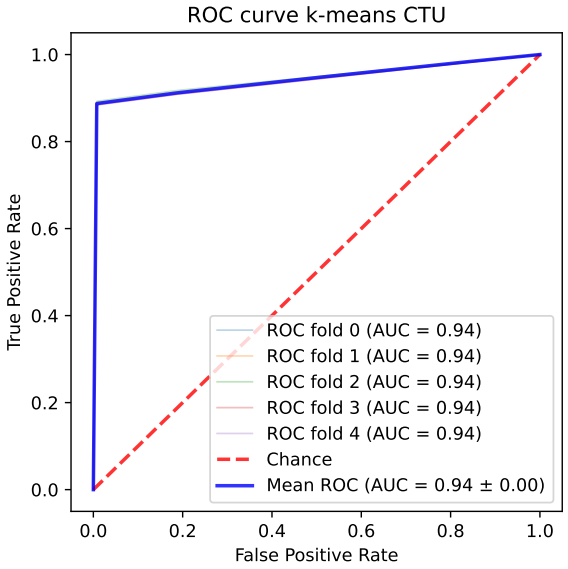

Figure 31: k-means ROC curve for the CTU dataset

little less for the CIC Layer 1 dataset, compared to the other datasets. The CIC Layer 1 dataset is apparently harder to classify. The k-means method clearly performs less on this dataset.

Table 11 shows the scores achieved in this thesis compared to previous work. Only the accuracy metric can be compared since only this metric is presented in previous work. The kNN model from this thesis has the highest accuracy, even higher than kNN applied by Vehskin et al. It is unknown what hyperparameters are used by Vehskin et al. but it is likely that in this thesis the hyperparameters are tuned more precisely. Note that in this thesis only 6 out of 23 features were selected. It is also possible that selecting more features results in more noise. The accuracy of the k-means model is surprisingly close to the other, supervised, models.

The comparison of the scores of the CIC dataset is shown in Tables 12 and 13. The performance of the kNN model is similar to the scores of previous work for both Layer 1 and Layer 2. It is interesting to observe that similar scores are achieved while this thesis only uses 5 instead of 29 features for Layer 1 and 6 instead of 29 for Layer 2. The k-means model performs impressively well on CIC Layer 2. With a recall of 1, all malicious DoH traffic has correctly been classified as malicious DoH. The precision is about 7 percent lower, indicating that k-means suffers from a larger number of false positives, i.e., benign

		kNN	k-means
CTU	Accuracy	1.000	0.988
	Precision	0.996	0.821
	Recall	0.992	0.887
	F1-score	0.994	0.853
CIC Layer 1	Accuracy	0.994	0.881
	Precision	0.992	0.745
	Recall	0.984	0.747
	F1-score	0.988	0.745
CIC Layer 2	Accuracy	0.999	0.928
	Precision	0.999	0.928
	Recall	1.000	1.000
	F1-score	1.000	0.963

Table 10: Scores of supervised learning (kNN) compared with unsupervised learning (k-means)

	Accuracy
kNN	*0.994*
DecisionTree	*0.994*
Random Forest	*0.995*
Naive Bayes	*0.968*
Ada-boosted DT	*0.996*
kNN	1.000
k-means	0.988

Table 11: Scores of CTU compared to the results obtained by Vekshin et al. (in italics) [67]

	Precision	**Recall**	**F1-score**
Random Forest	*0.993*	*0.993*	*0.993*
Decision Tree	*0.993*	*0.993*	*0.993*
SVM	*0.877*	*0.877*	*0.877*
Naive Bayes	*0.84*	*0.834*	*0.833*
DNN	*0.97*	*0.97*	*0.97*
2D CNN	*0.98*	*0.98*	*0.98*
kNN	0.992	0.984	0.988
k-means	0.745	0.747	0.745

Table 12: Scores of CIC Layer 1 compared to the results obtained by MontazeriShatoori et al. (in italics) [41]

	Precision	Recall	F1-score
Random Forest	*0.999*	*0.999*	*0.999*
DecisionTree	*0.999*	*0.999*	*0.999*
SVM	*0.89*	*0.885*	*0.884*
Naive Bayes	*0.836*	*0.833*	*0.832*
DNN	*0.98*	*0.98*	*0.98*
2D CNN	*0.99*	*0.99*	*0.99*
kNN	0.999	1.000	1.000
k-means	0.928	1.000	0.963

Table 13: Scores of CIC Layer 2 compared to the results obtained by MontazeriShatoori et al. (in italics) [41]

DoH is more often classified as being malicious. For CIC Layer 1, the scores of the k-means model are less compared to the other, supervised, models. Both the precision and recall are lower for the k-means model. That means, both DoH more often gets classified as being non-DoH and non-DoH more often gets classified as DoH.

6.2.5 Error Analysis

As explained in Section 5.3, an error analysis is performed to identify why certain data samples are misclassified while others are correctly classified.

Supervised learning Visualizing the value distributions of correctly classified and misclassified data samples shows what areas are generally well classified and what areas contain a lot of misclassified samples. A histogram of the classification distribution for the packet length mean feature by the kNN model is shown in Figure 32. Note that the visualized data is descaled, for training the model it was scaled as explained in Section 5.2.1, however, for visualization, the data is transformed back to the original scale. DoH traffic forms the positive class and non-DoH the negative.

Figure 32 shows that flows with a mean packet length of roughly higher than 500 bytes are almost always correctly classified as being non-DoH traffic. However, the area hard to classify is where both DoH and non-DoH flows intersect, flows with a packet length smaller than 500 bytes. When DoH and non-DoH flows have similar mean packet lengths, both DoH and non-DoH traffic gets misclassified, resulting in false negatives and false positives.

The figures of the histograms of the classification distribution for the remaining features are shown in Appendix 8. From the distribution for the median packet length

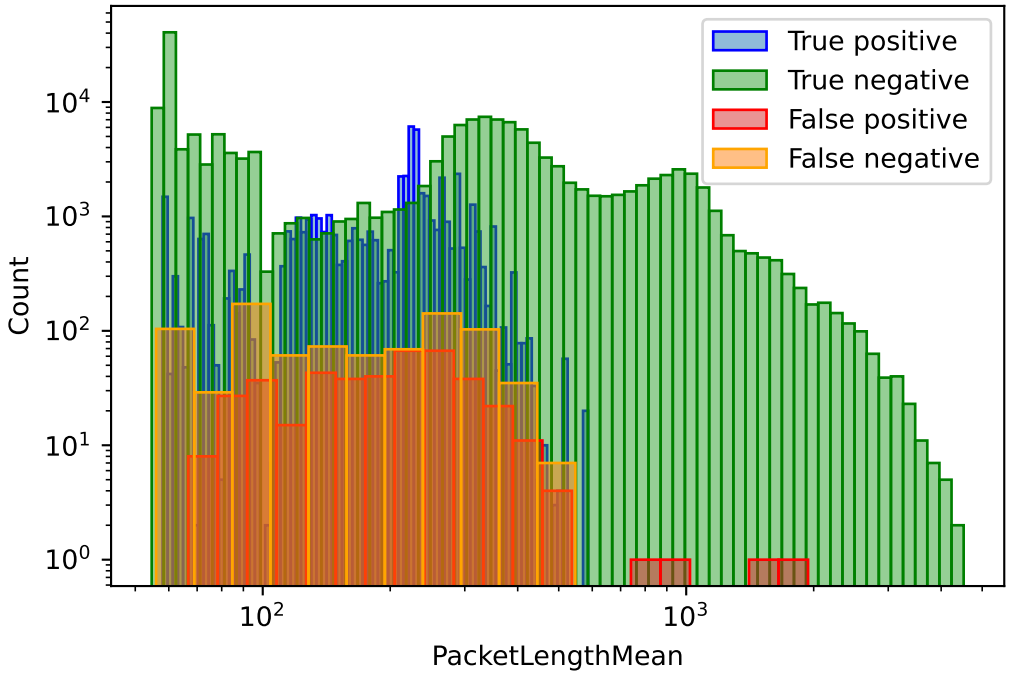

Figure 32: Histogram of the kNN classification distribution for the packet length mean feature

feature, it can be deduced that almost all errors regard flows with a packet length median smaller than 100 bytes. Flows with a median packet length larger than 200 bytes are nearly perfectly classified as non-DoH.

In general, for larger values, classification is relatively easy, since these relate to non-DoH flows. For small values, however, classification is hard since both classes, DoH and non-DoH, contain flows with smaller values.

The histograms only show the distribution per single feature. To get a closer look at what data gets misclassified, also scatter plots, such as shown in Figure 33, were generated. In these scatter plots, the x-axis denotes one feature and the y-axis shows another feature. The scatter plots show more detail since they show two dimensions instead of one. A disadvantage of scatter plots is that data points can lie on top of each other making the bottom one invisible. It is important to mention that the data points of the latest added category are shown in front of the others. Consequently, Figure 33 would have shown more red than blue samples if no countermeasures were taken. To reduce the impact of this phenomenon, the figure is created by performing 20 rounds of adding 500

randomly selected data samples per category. It is possible that identical data points are added multiple times. Due to the log scale, the data points are more spread and thus overlap to a lesser extent.

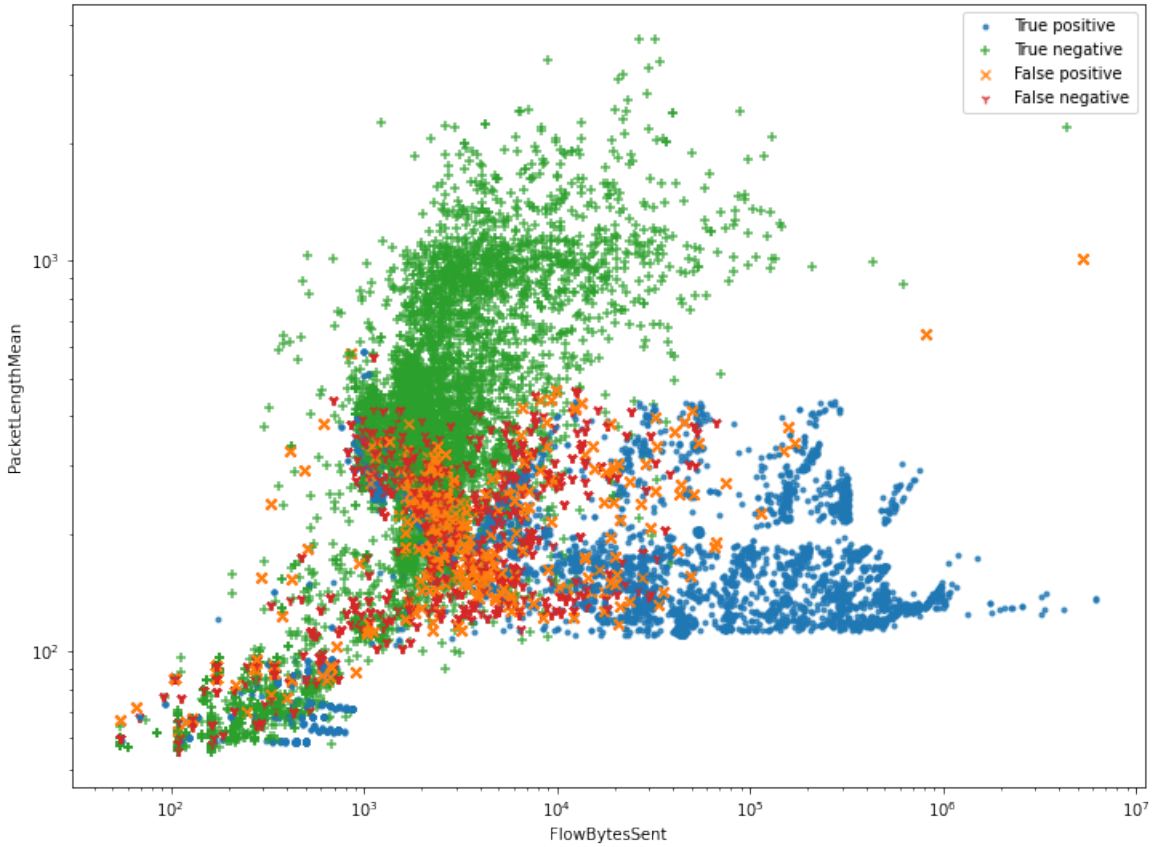

Figure 33: Scatter plot of the kNN classification for the flow bytes sent and mean packet length features

Figure 33 shows the scatter plot for the flow bytes sent and mean packet length features. When analysing the figure, a green block, denoting the correctly classified non-DoH traffic and a blue block, denoting the correctly classified DoH flows can be identified. However, we are mainly interested in the misclassified samples. From the figure, it becomes clear that the largest part of the green block is correctly classified; it only contains non-DoH flows classified as non-DoH. However, the orange points inside the blue block

indicate that there are also non-DoH data samples inside the DoH block. Since the non-DoH data points are a bit spread all over the place, there are both false positives and false negatives inside the blue block. This gives the impression that by tuning the k-value for the kNN model, the ratio between false positives and false negatives can be tuned. With a higher k-value, the non-DoH data points inside the DoH block are very likely to be classified as a false positive.

Figure 33 also shows that most misclassified samples are located at the intersection of the green and the blue block. Where the DoH and the non-DoH data is similar, classification becomes hard.

The scatter plots of the other combination of features are shown in Appendix 8. Analysis of these figures shows that often a green and blue block can be identified and the misclassified samples are mainly located at the intersection of these two blocks. This means that similar data causes misclassification. Additionally, patterns indicating correctly classified data are observed. These patterns include:

- Flows with a mean packet length larger than 500 bytes are correctly classified as non-DoH
- Both DoH and non-DoH flows with a median packet length larger than 100 are classified correctly
- Flows with more than 10,000 bytes sent are correctly classified as DoH

In conclusion, similar data is mainly the problem for misclassified samples. Furthermore, three patterns indicating correctly classified samples were identified.

Unsupervised learning The unsupervised learning method k-means clusters the data samples. The first step of the error analysis is examining in how many clusters there are a significant number of errors. Therefore, a plot is created in which the size per cluster is shown, see Figure 34. In dark blue, the number of true DoH samples is shown. The letter above each bar indicates whether the cluster is classified as DoH (D) or non-DoH (n). As only a few clusters contain more than 10,000 data samples, there are only a few large clusters and many small ones. The figure shows that the smaller ones are classified close to perfect. The large clusters, however, contain a lot of samples from both classes, resulting in a number of misclassified samples.

To determine why the larger clusters contain a large number of misclassified samples, the DoH and non-DoH clusters containing the largest number of misclassified samples

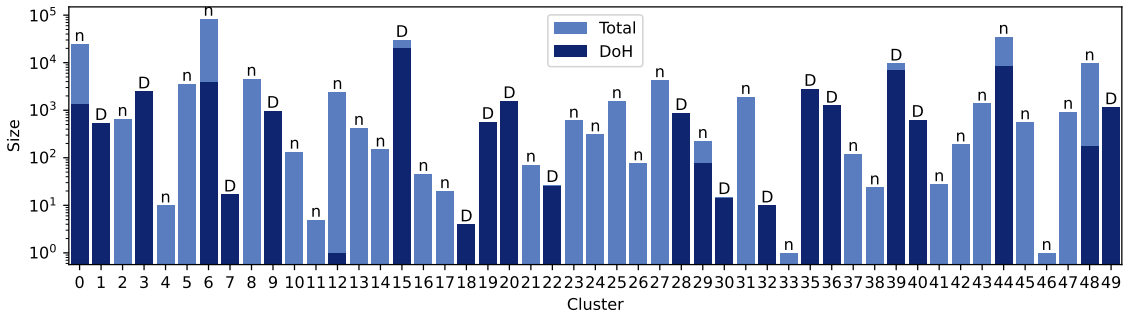

Figure 34: The size of the clusters formed by the k-means model. The letter above each bar indicates whether the cluster is classified as DoH (D) or non-DoH (n).

Cluster	All		15 (DoH)		6 (non-DoH)	
Feature	min	max	min	max	min	max
FlowBytesSent	54	8243	1455	2159	54	395
FlowBytesReceived	54	2482	4471	5187	54	72
PacketLengthMean	54	993	176	264	54	72
PacketLengthMedian	54	144	76	76	58	63
ResponseTimeTimeMedian	$2.0e-06$	0.05	0.01	0.02	$3.0e-06$	0.08

Table 14: Cluster ranges

are scrutinized. The DoH cluster containing the most misclassified samples is cluster 15. This cluster contains 36,022 samples in total, of which 23245 are DoH and the remaining 12777 are non-DoH. The other interesting cluster is cluster 6, with 37366 samples of which 6645 are DoH and the remaining 30721 are non-DoH.

As a clustering algorithm is used, each cluster contains data points with similar values. Analysing the value ranges per feature for each of the clusters together with the ranges for all data, gives a clue where the clusters fit in the bigger picture. Table 14 shows the value ranges for the selected clusters and for all data. To reduce the impact of outliers, outliers are removed based on the z-score [7]. Data points with a z-score greater than 3.5 are not taken into account when computing the ranges. Consequently, the DoH cluster contains FlowBytesReceived values higher than the maximum of all clusters because the values are considered as outliers.

The ranges can be analysed by comparing the values to the value distributions of all traffic (see Section 6.1.1). Analysis shows that cluster 15 regards common DoH traffic (see Figures 40 and 11). Similarly, the non-DoH cluster regards common non-DoH flows.

When zooming in on the DoH cluster, the value distribution per feature shows that

the non-DoH flows are evenly distributed over the range for most features, see Figure 35. The DoH flows are more concentrated around certain values. It is likely that the length of DoH flows, in bytes, can be expressed as: TLS handshake length + n-times DoH request/response length. If that is the case, classification algorithms can use that.

For the median response time feature, the DoH class is more concentrated to higher values, as visualized in Figure 36. Two peaks are visible. Increasing the number of clusters might result in a separate cluster for each of the peaks which might lead to high classification performance.

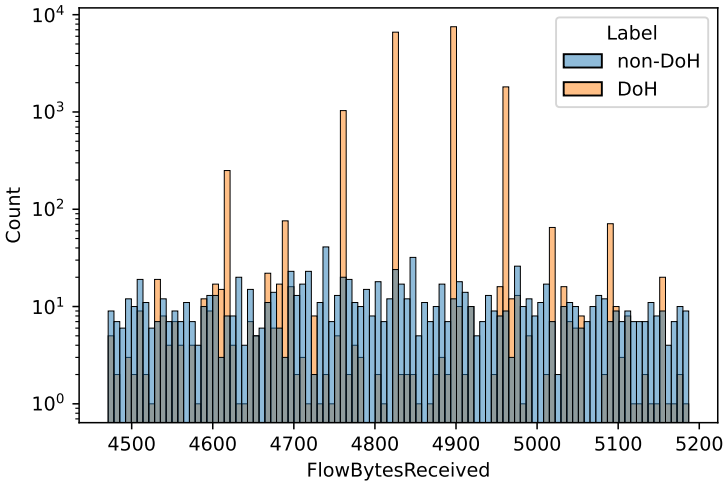

Figure 35: Histogram of value distribution of cluster 15 (DoH) for the flow bytes received feature

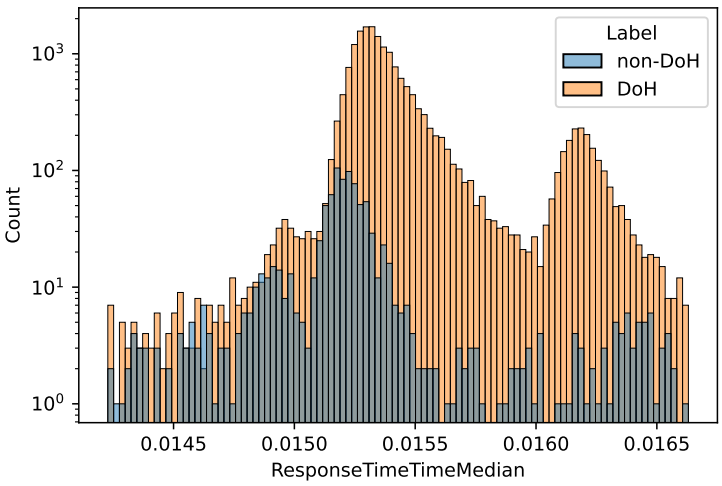

Figure 36: Histogram of value distribution of cluster 15 (DoH) for the median response time feature

The scatter plots of each combination of two features confirm that data points inside a cluster of both DoH and non-DoH are similar. Due to the similarities, it is hard to improve the performance of the model. Figure 37 shows the scatter plot for the flow bytes received and median response time features. A blue/orange stripe can be identified with a median response time between 0.01 and 0.03. However, there are a few groups of separated orange, DoH, samples. By increasing the number of clusters, these groups are likely to be placed in separate clusters, which improves the classification. On the other hand, does increasing the number of clusters also means that more labelled data is required to get a classification (more detailed explained in Section 6.2.2).

The analysis of cluster 6, the non-DoH cluster containing the most misclassified samples, includes analysing the value distributions per feature and analysing the scatter plots

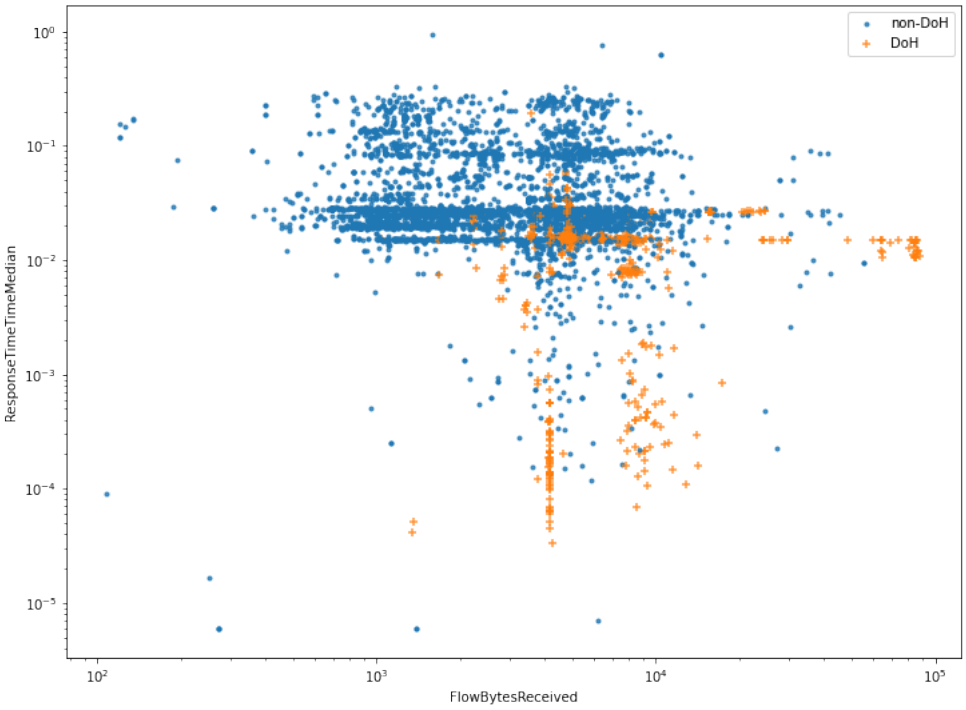

Figure 37: Scatter plot of the k-means classification of cluster 15 (DoH) for the flow bytes received and median response time features

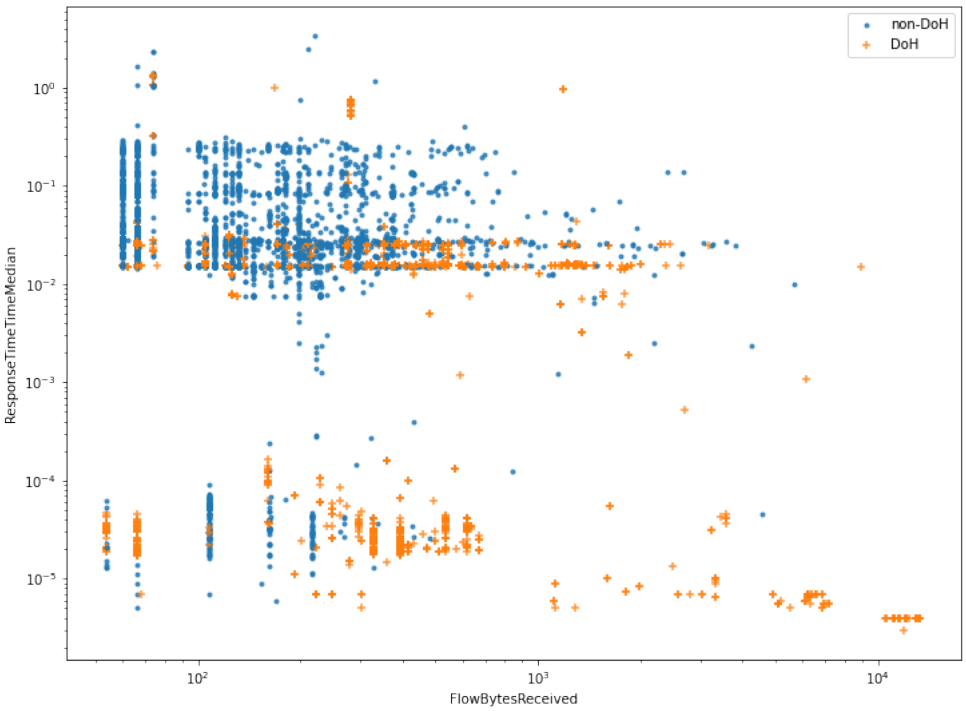

Figure 38: Scatter plot of the k-means classification of cluster 6 (non-DoH) for the flow bytes received and median response time features

of each combination of features.

Similarly to the value distribution of cluster 15, the DoH cluster, the values of non-DoH traffic is spread over a certain domain for most of the features, whereas the DoH traffic is more concentrated around certain values. Unfortunately, it is hard to use this information to improve a clustering algorithm.

The analysis of the scatter plots per feature combination confirms that the DoH and non-DoH traffic are similar for large parts of the non-DoH cluster. However, as visualized in Figure 38, there are some groups of DoH samples separated from the large group. This indicates that increasing the number of clusters is promising as the separated DoH groups will likely get their own cluster.

In short, most of the clusters are classified almost entirely correct. Both the largest clusters, however, contain a lot of misclassified samples. The largest DoH and non-DoH

clusters contain respectively common DoH and non-DoH flows. Mainly because the data is similar for both classes, the clusters contain numerous false positives or false negatives. However, visualization of the data points per combination of features show that misclassification of certain samples can likely be prevented by increasing the number of clusters.

7 Conclusions

In this thesis, we worked on the research to the detection of (malicious) DoH network traffic to improve the security against malicious use of DoH. To this end, the features that can be used for DoH detection are analysed and unsupervised learning is applied and compared to supervised learning.

In previous work, two datasets containing labelled DoH traffic were created and published (called CTU dataset and CIC dataset). This work utilizes the datasets. The dataset includes a number of features extracted from the data.

To conclude the thesis, we provide the answers to the research questions formulated in Section 1.1.

1. Using which features can DoH traffic be distinguished from web traffic and using which features can benign DoH be distinguished from DoH tunnelling?

To answer the first research question, the value distribution per feature was visualized and analysed. We first compare benign and malicious DoH with non-DoH web traffic and second, we compare malicious DoH with benign DoH. In the analysis of DoH versus non-DoH, nine features emphasizing the characteristics of DoH traffic were found. Namely, the flow duration, the number of incoming and outgoing packets per flow, the amount of packets bursts per flow, the ratio between incoming and outgoing bytes, the ratio of the amount of incoming and outgoing packets, the mean and median packet length and lastly, the packet length variance.

For the detection of malicious use of DoH in the form of tunnelling, five distinguishing features were identified. As a result of sending additional data when tunnelling, the values of the following five features are notably higher for malicious DoH traffic: the number of bytes sent, the number of bytes received, the packet length mean, median and variance.

2. How well can DoH traffic be distinguished from non-DoH network traffic and benign DoH be distinguished from DoH tunnelling using unsupervised learning techniques?

To answer the second research question, both supervised and unsupervised learning methods are applied. The results using supervised learning are a baseline for unsupervised learning. The approach of applying machine learning included systematic feature

selection using a combination of Recursive Feature Elimination (RFE) and Sequential Feature Selector (SFS). As the supervised learning method, the k-nearest neighbours model was used and as the supervised learning method, k-means was chosen.

Evaluation of the models showed that the supervised learning method achieved equal classification scores as presented in previous work. It is interesting to observe that, we only use six or fewer features whereas previous work used 23 or 29 features. Analysis of training showed that, depending on the dataset, 25,000; 100,000 and 300,000 training samples are sufficient to completely train the models.

The unsupervised learning methods are between 1 and 10 percent less accurate compared to the supervised learning method. The precision, indicating false positives, is between 10 and 20 percent lower for the unsupervised learning method, with a minimum precision of 0.75.

All in all, this thesis showed which features characterize DoH traffic and can be used for (malicious) DoH detection. Additionally, unsupervised learning is applied for the detection and classification of DoH traffic. The scores for supervised learning are naturally better than for unsupervised learning but the unsupervised learning method scores surprisingly well.

8 Future Work

This section describes four issues that could be examined in future work.

In this thesis, it is shown that a clustering method is capable of detecting DoH traffic. However, there are more unsupervised learning methods that can be used for the detection of (malicious) DoH. For instance, anomaly detection or an autoencoder [34] can be used. Possibly, other unsupervised learning methods are more accurate or more useful than clustering.

Future studies can also address harder DoH detection cases. In previous studies and in this thesis, the DoH traffic as a result of a DoH configured browser is used. These DoH flows have clear characteristics such as a long flow duration. However, a custom program can send only one DoH request, resulting in a short flow containing a single DoH request and response. It might be harder to classify such a flow as DoH.

Additionally, other types of malicious use of DoH can also be an interesting topic for future research. In particular, DGAs in botnets might abuse DoH. Patsakis et al. published a paper about the classification of encrypted DNS queries, related to botnets. Future work can aim to distinguish DGA related DoH traffic from other HTTPS traffic.

Finally, future research might apply the DoH detection methods in a real network. Currently, DoH traffic is only distinguished from browser traffic. However, there might be HTTPS traffic created by other applications than browsers, with characteristics more similar to DoH traffic. If this is the case, similar traffic is likely being classified as DoH traffic. The current detection methods will produce a lot of false positives in that case.

References

[1] Donald E. Eastlake 3rd. Domain Name System Security Extensions. RFC 2535, March 1999.

[2] Lawrence Abrams. New Spam Campaign Controlled by Attackers via DNS TXT Records, June 2019. https://www.bleepingcomputer.com/news/security/new-spam-campaign-controlled-by-attackers-via-dns-txt-records/.

[3] AdGuard. AdGuard DNS | AdGuard Knowledgebase. https://kb.adguard.com/en/dns. accessed: 23-06-2021.

[4] G DATA CyberDefense AG. New FrameworkPOS variant exfiltrates data via DNS requests | G DATA, 2014. https://www.gdatasoftware.com/blog/2014/10/23942-new-frameworkpos-variant-exfiltrates-data-via-dns-requests.

[5] Saif Al-Mashhadi, Mohammed Anbar, Shankar Karuppayah, and Ahmed K. Al-Ani. A Review of Botnet Detection Approaches Based on DNS Traffic Analysis. In Vincenzo Piuri, Valentina Emilia Balas, Samarjeet Borah, and Sharifah Sakinah Syed Ahmad, editors, *Intelligent and Interactive Computing*, pages 305–321, Singapore, 2019. Springer Singapore.

[6] Alexa. Alexa's top 500 sites on the web. https://www.alexa.com/topsites.

[7] Mary Banach. I have an outlier!, Sep 2011. data accessed: 26-07-2021.

[8] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007.

[9] K. Bumanglag and H. Kettani. On the Impact of DNS Over HTTPS Paradigm on Cyber Systems. In *2020 3rd International Conference on Information and Computer Technologies (ICICT)*, pages 494–499, 2020.

[10] T. Cejka, V. Bartos, M. Svepes, Z. Rosa, and H. Kubatova. NEMEA: A framework for network traffic analysis. In *2016 12th International Conference on Network and Service Management (CNSM)*, pages 195–201, 2016.

[11] Shateel A. Chowdhury. Domain generation algorithm – dga in malware - hackers terminal. https://hackersterminal.com/domain-generation-algorithm-dga-in-malware/, August 2019. accessed: 23-06-2021.

[12] Shateel A. Chowdhury. Malicious uses of Fast-Flux Service Networks (FFSN). https://hackersterminal.com/fast-flux-service-networks-ffsn-technique/, Dec 2019. accessed: 23-06-2021.

[13] Henry Clausen, Robert Flood, and David Aspinall. Traffic generation using containerization for machine learning, 2020.

[14] Cloudflare. DNS over HTTPS · 1.1.1.1 docs. https://developers.cloudflare.com/1.1.1.1/dns-over-https. accessed: 23-06-2021.

[15] Cloudflare. Running a DNS over HTTPS client. https://developers.cloudflare.com/1.1.1.1/dns-over-https/cloudflared-proxy. accessed: 23-06-2021.

[16] Jonathan Crussell, Thomas M Kroeger, Aaron Brown, and Cynthia Phillips. Virtually the same: Comparing physical and virtual testbeds. In *2019 International Conference on Computing, Networking and Communications (ICNC)*, pages 847–853, 2019.

[17] Olivier Dembour. dns2tcp | Penetration Testing Tools. https://tools.kali.org/maintaining-access/dns2tcp. accessed: 23-06-2021.

[18] C. J. Dietrich, C. Rossow, F. C. Freiling, H. Bos, M. v. Steen, and N. Pohlmann. On Botnets That Use DNS for Command and Control. In *2011 Seventh European Conference on Computer Network Defense*, pages 9–16, 2011.

[19] Docker. Empowering App Development for Developers | Docker. https://www.docker.com/.

[20] DomainTools. Whois Lookup, Domain Availability & IP Search - DomainTools. https://whois.domaintools.com/.

[21] Wendy Ellens, Piotr Żuraniewski, Anna Sperotto, Harm Schotanus, Michel Mandjes, and Erik Meeuwissen. Flow-Based Detection of DNS Tunnels. In Guillaume Doyen, Martin Waldburger, Pavel Čeleda, Anna Sperotto, and Burkhard Stiller, editors, *Emerging Management Mechanisms for the Future Internet*, pages 124–135, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[22] Erik Ekman, Bjorn Andersson and Anne Bezemer. yarrick_iodine Official git repo for iodine dns tunnel. https://github.com/yarrick/iodine. accessed: 23-06-2021.

[23] F.J. Ferri, P. Pudil, M. Hatef, and J. Kittler. Comparative study of techniques for large-scale feature selection. In Edzard S. GELSEMA and Laveen S. KANAL, editors, *Pattern Recognition in Practice IV*, volume 16 of *Machine Intelligence and Pattern Recognition*, pages 403–413. North-Holland, 1994.

[24] Google. Public DNS | Google Developers. `https://developers.google.com/speed/public-dns`. accessed: 23-06-2021.

[25] Martin Grill, Ivan Nikolaev, Veronica Valeros, and Martin Rehak. Detecting dga malware using netflow. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1304–1309, 2015.

[26] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning*, 46(1):389–422, Jan 2002.

[27] Drew Hjelm. A New Needle and Haystack: Detecting DNS over HTTPS Usage, Aug 2019.

[28] P. Hoffman and P. McManus. DNS Queries over HTTPS (DoH). RFC 8484, RFC Editor, October 2018.

[29] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. Specification for dns over transport layer security (tls). RFC 7858, RFC Editor, May 2016.

[30] iagox86. iagox86_dnscat2. `https://github.com/iagox86/dnscat2`. accessed: 23-06-2021.

[31] N. Ishikura, D. Kondo, I. Iordanov, V. Vassiliades, and H. Tode. Cache-Property-Aware Features for DNS Tunneling Detection. In *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 216–220, 2020.

[32] Wilhelm Kirch, editor. *Pearson's Correlation Coefficient*, pages 1090–1091. Springer Netherlands, Dordrecht, 2008.

[33] M. Korczyński and A. Duda. Markov chain fingerprinting to classify encrypted traffic. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 781–789, 2014.

[34] Quoc V. Le. Tutorial on Deep Learning Part 2: Autoencoders, Convolutional Neural Networksand Recurrent Neural Networks. https://cs.stanford.edu/~quocle/tutorial2.pdf, 2015. accessed: 03-07-2021.

[35] Ceshine Lee. Feature Importance Measures for Tree Models - Part I, Sep 2020. accessed: 30-06-2021.

[36] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature selection. *ACM Computing Surveys*, 50(6):1–45, Jan 2018.

[37] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, page 413–422, USA, 2008. IEEE Computer Society.

[38] A. Mayrhofer. Padding Policies for Extension Mechanisms for DNS (EDNS(0)). RFC 8467, RFC Editor, October 2018.

[39] Meng Shen, Mingwei Wei, Liehuang Zhu, Mingzhong Wang, and Fuliang Li. Certificate-aware encrypted traffic classification using Second-Order Markov Chain. In *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, pages 1–10, 2016.

[40] M. MontazeriShatoori, L. Davidson, G. Kaur, and A. Habibi Lashkari. DoHMeter, 2019. https://github.com/ahlashkari/DoHLyzer/tree/master/meter.

[41] M. MontazeriShatoori, L. Davidson, G. Kaur, and A. Habibi Lashkari. Detection of DoH Tunnels using Time-series Classification of Encrypted Traffic. In *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pages 63–70, 2020.

[42] Asaf Nadler, Avi Aminov, and Asaf Shabtai. Detection of malicious and low throughput data exfiltration over the DNS protocol. *Computers & Security*, 80:36 – 53, 2019.

[43] Andrew Ng. Lecture 13: Clustering. https://vkosuri.github.io/CourseraMachineLearning/home/week-8/lectures/pdf/Lecture13.pdf, 2012. accessed: 01-07-2021.

[44] Maël Nogues, David Brosset, Hanan Hindy, Xavier Bellekens, and Yvon Kermarrec. Labelled network capture generation for anomaly detection. In Abdelmalek Benzekri, Michel Barbeau, Guang Gong, Romain Laborde, and Joaquin Garcia-Alfaro,

editors, *Foundations and Practice of Security*, pages 98–113, Cham, 2020. Springer International Publishing.

[45] P. Mockapetris. Domain names - implementation and specification. RFC 1035, November 1987.

[46] W. Pan, G. Cheng, and Y. Tang. WENC: HTTPS Encrypted Traffic Classification Using Weighted Ensemble Learning and Markov Chain. In *2017 IEEE Trustcom/BigDataSE/ICESS*, pages 50–57, 2017.

[47] Constantinos Patsakis, Fran Casino, and Vasilios Katos. Encrypted and covert DNS queries for botnets: Challenges and countermeasures. *Computers & Security*, 88:101614, 2020.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[49] Claudia Perlich. *Learning Curves in Machine Learning*, pages 577–580. Springer US, Boston, MA, 2010.

[50] 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.

[51] Quad9. DoH with Quad9 DNS Servers | Quad9. `https://www.quad9.net/news/blog/doh-with-quad9-dns-servers/`. accessed: 23-06-2021.

[52] Eric Rescorla. HTTP Over TLS. RFC 2818, May 2000.

[53] scikit-learn developers. 1.11. Ensemble methods — scikit-learn 0.24.2 documentation. `https://scikit-learn.org/stable/modules/ensemble.html#forest`. accessed: 30-06-2021.

[54] scikit-learn developers. 1.6. Nearest Neighbors — scikit-learn 0.24.2 documentation. `https://scikit-learn.org/stable/modules/neighbors.html#classification`. accessed: 30-06-2021.

[55] scikit-learn developers. 2.3. Clustering — scikit-learn 0.24.2 documentation. https://scikit-learn.org/stable/modules/clustering.html#k-means. accessed: 30-06-2021.

[56] scikit-learn developers. 3.2. Tuning the hyper-parameters of an estimator — scikit-learn 0.24.2 documentation. https://scikit-learn.org/stable/modules/grid_search.html#grid-search. accessed: 30-06-2021.

[57] scikit-learn developers. 3.3. Metrics and scoring quantifying the quality of predictions — scikit-learn 0.24.2 documentation. https://scikit-learn.org/stable/modules/model_evaluation.html#confusion-matrix. accessed: 01-07-2021.

[58] scikit-learn developers. Receiver Operating Characteristic (ROC) — scikit-learn 0.24.2 documentation. https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html. accessed: 30-06-2021.

[59] scikit-learn developers. sklearn.preprocessing.StandardScaler — scikit-learn 0.24.2 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler. accessed: 29-06-2021.

[60] Selenium. SeleniumHQ Browser Automation. https://www.selenium.dev/.

[61] B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. 02 2018.

[62] Sunil Kumar Singh and Pradeep Kumar Roy. Detecting Malicious DNS over HTTPS Traffic Using Machine Learning. In *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, pages 1–6, 2020.

[63] The Proofpoint Threat Insight Team. PsiXBot Now Using Google DNS over HTTPS and Possible New Sexploitation Module: Proofpoint US. https://www.proofpoint.com/us/threat-insight/post/psixbot-now-using-google-dns-over-https-and-possible-new-sexploitation-module, September 2019. accessed: 03-07-2021.

[64] Dinh-Tu Truong, Dac-Tot Tran, and Bao Huynh. Detecting Malicious Fast-Flux Domains Using Feature-based Classification Techniques. *Journal of Internet Technology*, 21(4):1061–1072, 2020.

[65] Alex Turing and Genshen Ye. An Analysis of Godlua Backdoor - Netlab Blog. https://blog.netlab.360.com/an-analysis-of-godlua-backdoor-en/, July 2019. accessed: 03-07-2021.

[66] A.P. Varga and R.K. Moore. Hidden Markov model decomposition of speech and noise. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 845–848 vol.2, 1990.

[67] Dmitrii Vekshin, Karel Hynek, and Tomas Cejka. Dataset used for detecting DNS over HTTPS by Machine Learning, May 2020.

[68] Dmitrii Vekshin, Karel Hynek, and Tomas Cejka. DoH Insight: Detecting DNS over HTTPS by Machine Learning. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ARES '20, New York, NY, USA, 2020. Association for Computing Machinery.

[69] Zhongjiang Yao, Jingguo Ge, Yulei Wu, Xiaosheng Lin, Runkang He, and Yuxiang Ma. Encrypted traffic classification based on Gaussian mixture models and Hidden Markov Models. *Journal of Network and Computer Applications*, 166:102711, 2020.

[70] 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*, 01 2019.

Appendix A: Related work, Encrypted data classification

Since the thesis at hand focuses on classifying DoH traffic and DoH traffic is a form of encrypted data, the related work about encrypted data classification is relevant for this work.

In the research area about classifying applications based on the (encrypted) TLS network traffic, Korczýnski and Duda started using the unencrypted "Content Type" field of the TLS layer as a feature for classification [33]. The Content Type is a field in TLS indicating the content of a packet. During the initial TLS handshake it has values like Client Hello, Server Hello and after the handshake it has the value Application Data, indicating encrypted application data. Fingerprints of the Content Type values of different applications are created by the use of a Markov Chain. A Markov Chain is a chain of events, in which each event is only dependent on the state of the previous event. Each transition from one event to another is denoted with a probability. Naturally, a TLS session starts with a handshake, then several times application data is transferred and at the end, an alert message is sent. The study showed, that this process varies for different types of applications. The researchers gathered data captured on a university campus over several days. By comparing regularly used applications like PayPal, Twitter and seven others, the researchers were able to test their approach on real data. The classification was accurate (above 90%) for most applications and less accurate for others. The researchers also looked at the cause of the differences in the fingerprints. They found that the main causes are: incorrect implementations, misuse of TLS protocol, specific server implementation or the nature of the application. Furthermore, it was noted that fingerprints per application can change over time.

One issue with this method is that similar fingerprints of different applications leads to misclassification between the applications. Shen et al. [39] made the classification model more complex as a solution, instead of a first-order Markov Chain, a second-order Markov Chain was used . In a second-order Markov chain, a transition from one state to another is not only dependent on the current state, but also depends on the previous state. Consequently, the small differences in the fingerprints of two applications using the old model are amplified in the new model. This significantly increased the accuracy of the method. Additionally, the researchers added a new feature to the classification method, namely, the length of the TLS certificate. The length of a certificate is dependent on the certificate content and the certificate content appears to differ for different applications.

W.Pan et al. [46] did build further upon the method of Korczýnski and Duda and the

Study	(Added) Feature	Model
Korczynski and Duda [33]	Content Type	Markov Chain
Shen et al. [39]	Certificate length	Second-order Markov Chain
W.Pan et al. [46]	Joint feature	Added Hidden Markov Models
Yao et al. [69]	Inter Packet Time	Added Mixture of Gaussians

Table 15: The iterative improvements in the field of encrypted data classification

improvements of Shen et al. They tried to further improve the accuracy for classifying applications with a similar fingerprint and a certificate of similar length. This was done by using a joint feature, namely, Content Type and Packet Length, instead of only the Content Type. Additionally, Hidden Markov Models (HMM) [66] were added to the method. In an HMM, a Markov Process X is assumed to have observable states. Another process, Y is influenced by X and the goal is to learn about the hidden states of X by observing Y. In this case, the Markov Process with observable states are the fingerprints and the packet length of each transmitted data packet form the observable Markov Process. The final classification is done by a Weighted Ensemble Classifier (WENC). The incoming network flows are divided into a set of chunks. Each chunk index has its own classifier. Eventually, the classifiers get weighted by adjusting the weights until the total classification error is minimized

Most recently, Yao et al. [69] worked further on the WENC approach and added the Inter Packet Time (IPT) to the classification model. The IPT feature is added in the form of a mixture of Gaussians (MOG) model. A mixture model is a probabilistic model that can be used to cluster certain groups within a dataset. The MOG has two dimensions, that is, packet length and IPT. Where Korczýnski and Duda used a Markov Chain in which each event was denoted by the TLS content type, in this research each event is denoted by a MOG. Apart from this improvement, the research describes some other improvements, amongst others, how optimal model parameters can be selected efficiently.

These studies, summarized in Table 15, show that application classification, based on HTTPS network traffic can be done accurately. DoH can also be seen as an application that uses HTTPS, so it is likely that with the described methods DoH can be classified. It is worthwhile mentioning that none of the techniques described in Table 15 have been applied to DoH detection.

Appendix B: Results, Feature Analysis Figures

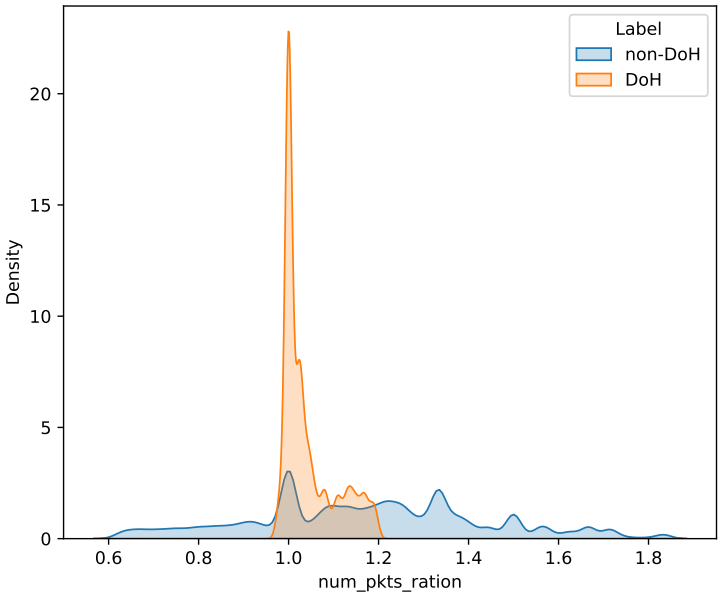

Figure 39: Value distribution of the ratio of the number of incoming and outgoing packets (CTU dataset)

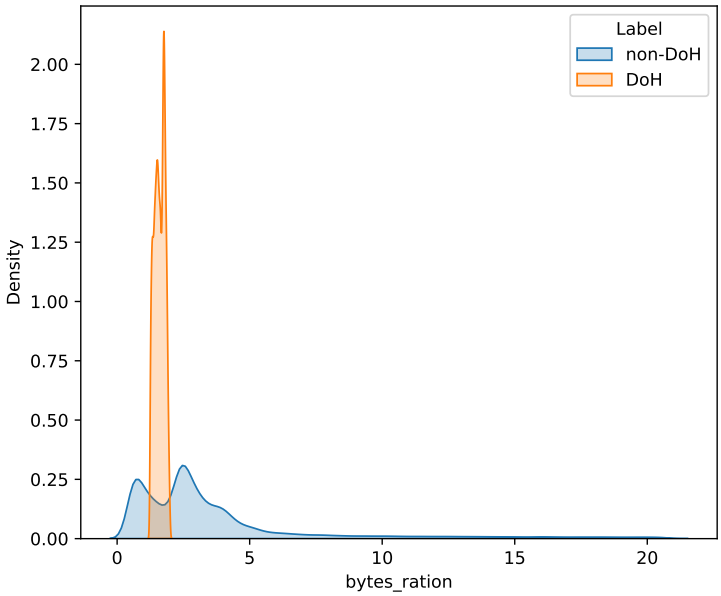

Figure 40: The value distribution of ratio of incoming and outgoing bytes (CTU dataset)

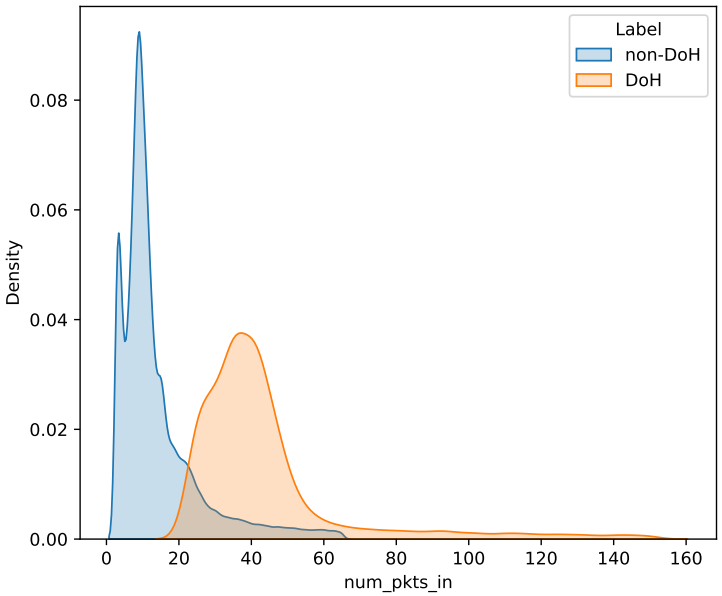

Figure 41: Value distribution of the number of incoming packets (CTU dataset)

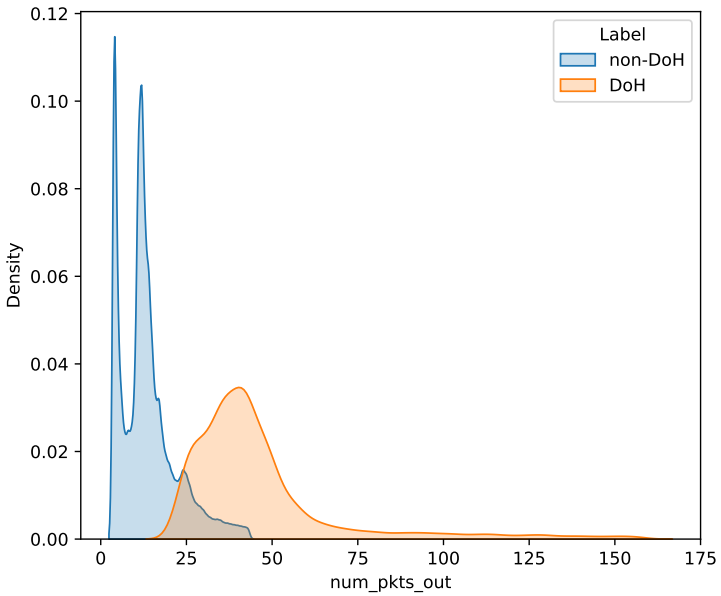

Figure 42: Value distribution of the number of outgoing packets (CTU dataset)

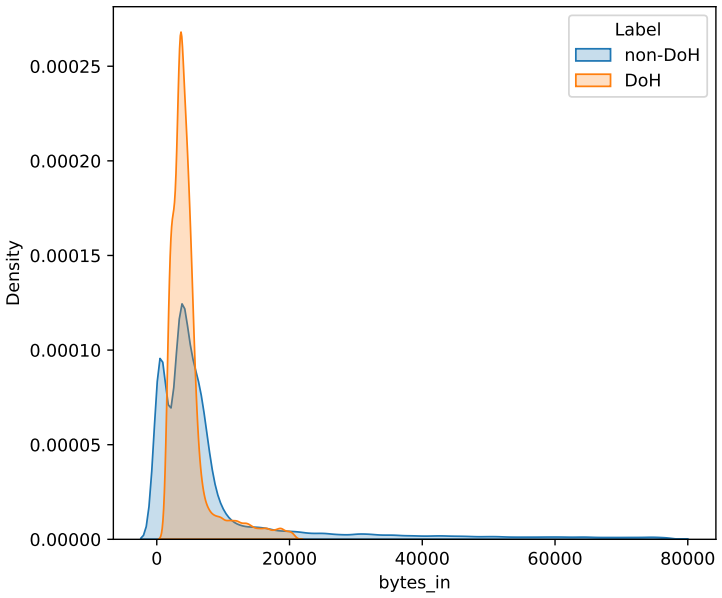

Figure 43: Value distribution of the number of bytes received (CTU dataset)

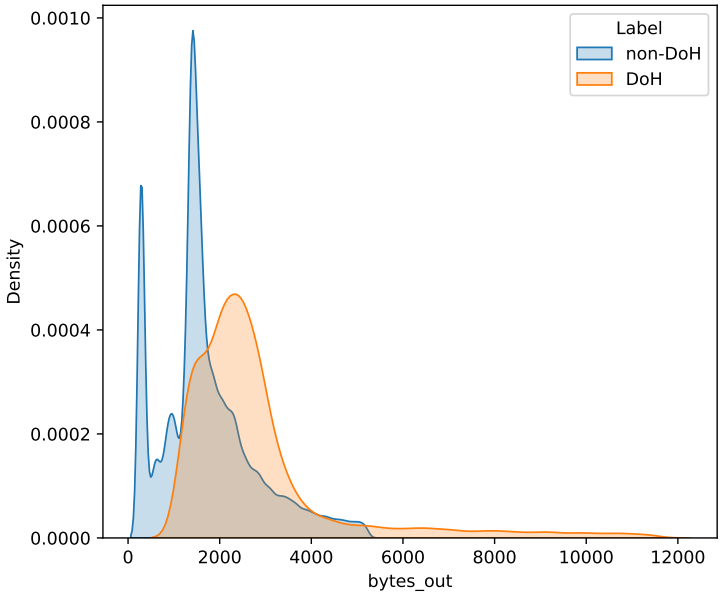

Figure 44: Value distribution of the number of bytes sent (CTU dataset)

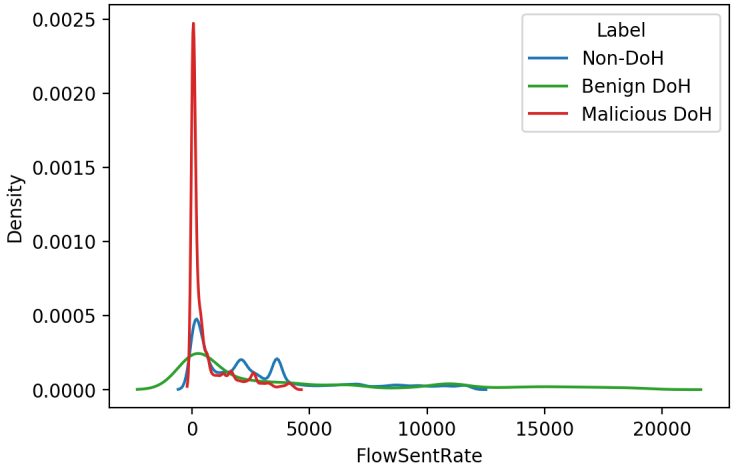

Figure 45: Value distribution of sending rate (CIC dataset)

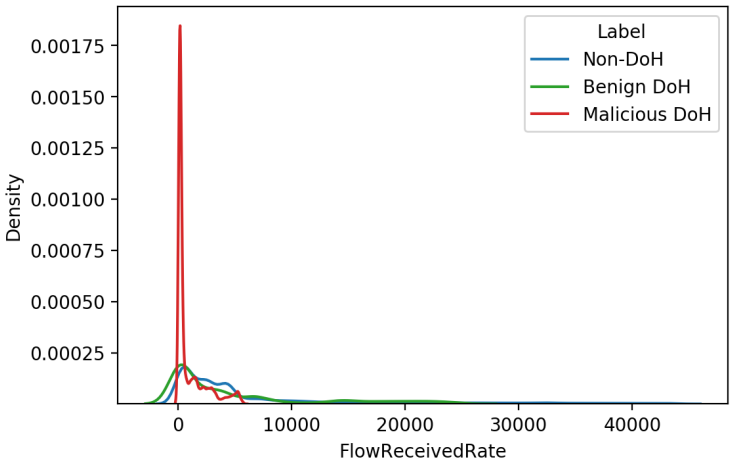

Figure 46: Value distribution of the receiving rate (CIC dataset)

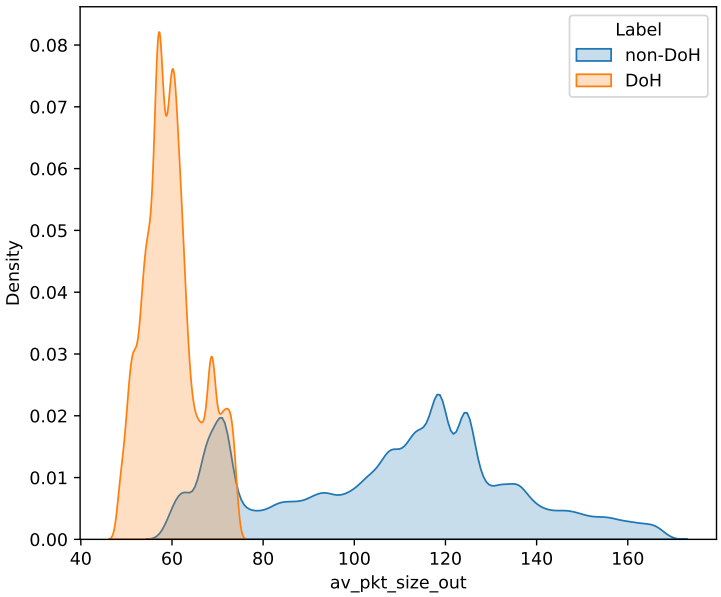

Figure 47: Value distribution of average outgoing packet length (CTU dataset)

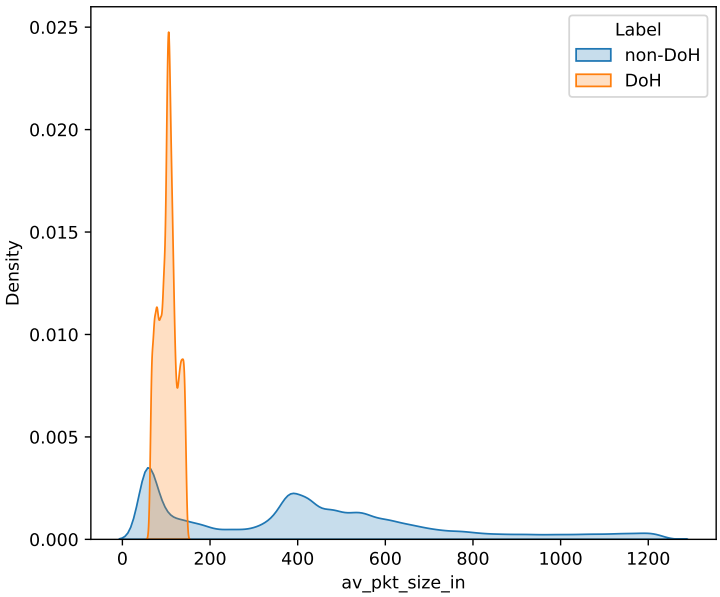

Figure 48: Value distribution of the average incoming packet length (CTU dataset)

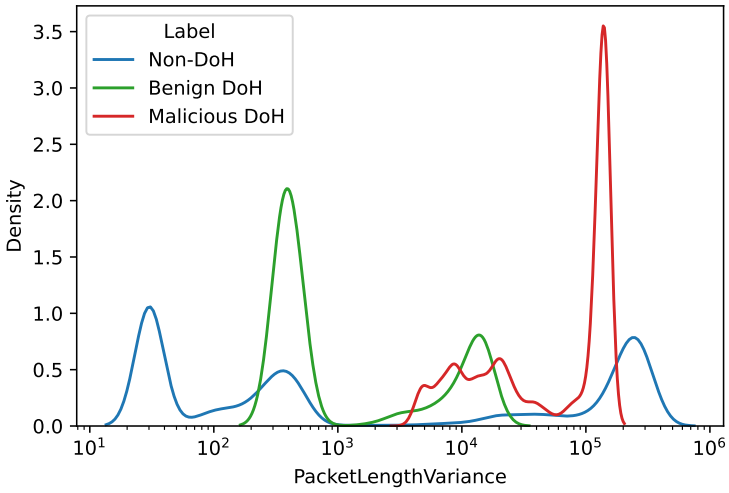

Figure 49: Value distribution of the variance of packet lengths (CIC dataset)

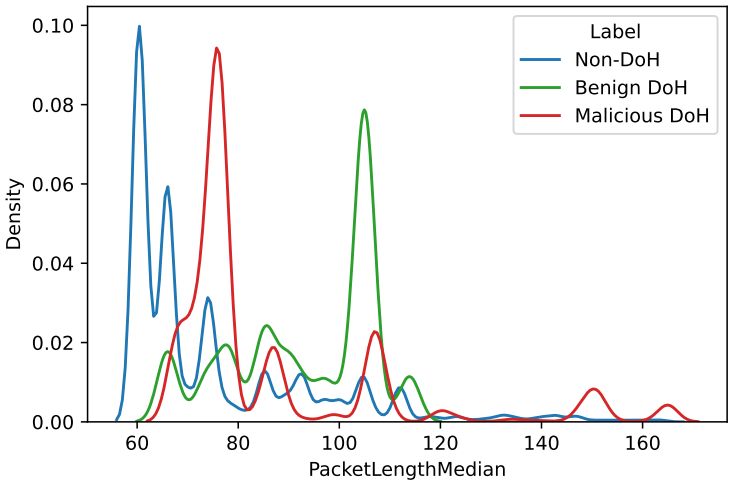

Figure 50: Value distribution of the median packet length (CIC dataset)

Appendix C: Results, Learning Curves

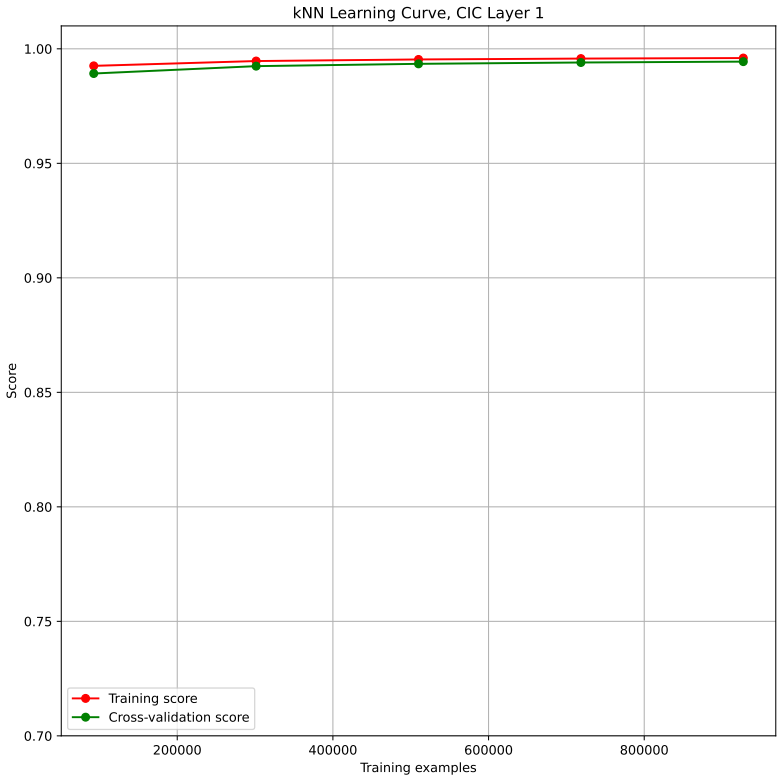

Figure 51: Learning curve of the k-nearest neighbours model for the CIC Layer 1 dataset

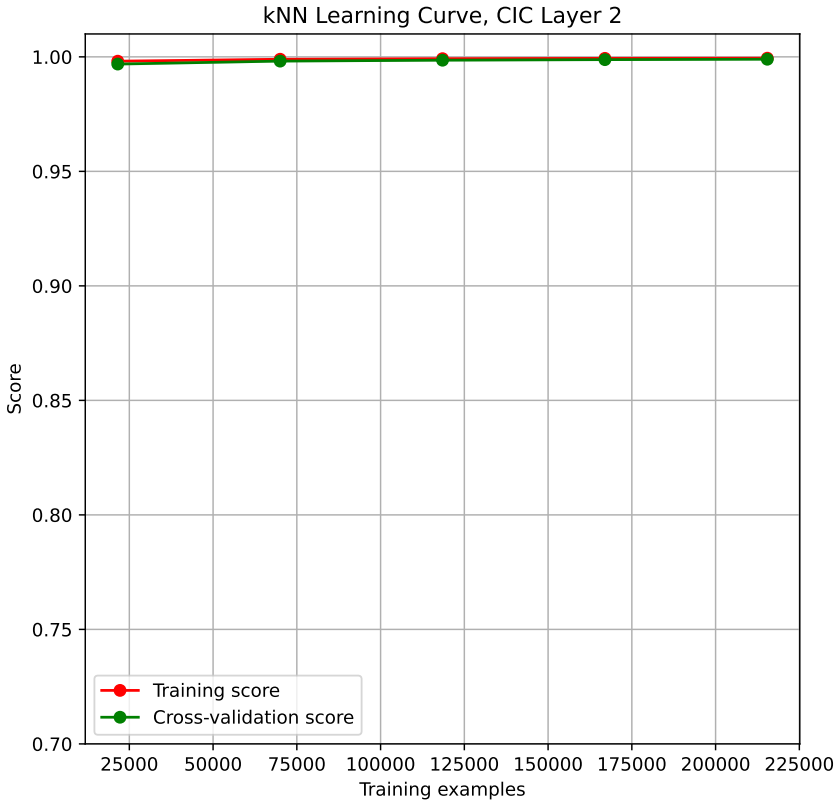

Figure 52: Learning curve of the k-nearest neighbours model for the CIC Layer 2 dataset

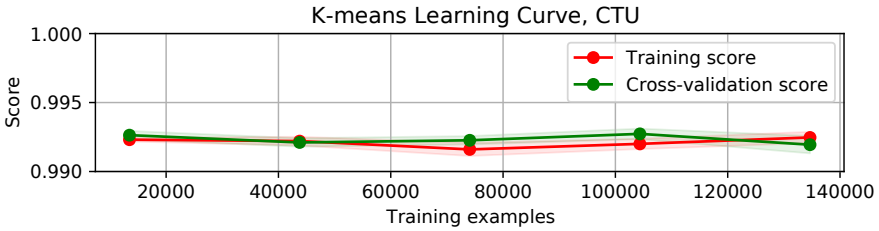

Figure 53: Learning curve of the k-means model for the CTU dataset

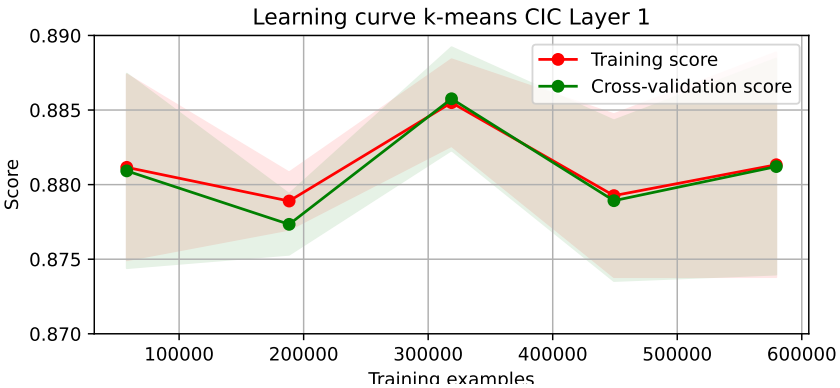

Figure 54: Learning curve of the k-means model for the CIC Layer 1 dataset

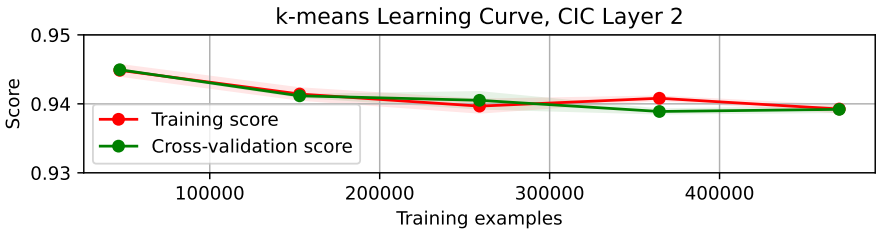

Figure 55: Learning curve of the k-means model for the CIC Layer 2 dataset

Appendix D: Results, Error Analysis

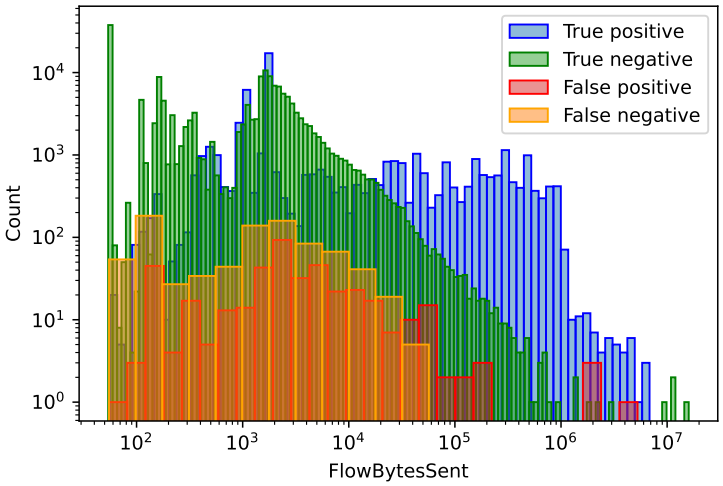

Figure 56: Histogram of the kNN classification distribution for the flow bytes sent feature

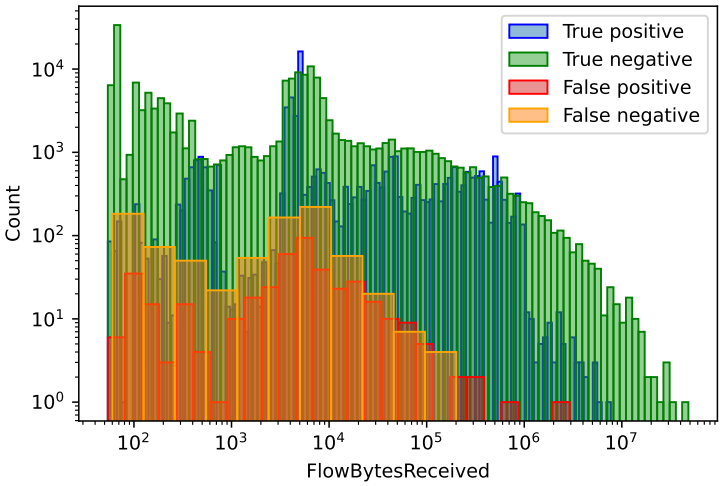

Figure 57: Histogram of the kNN classification distribution for the flow bytes received feature

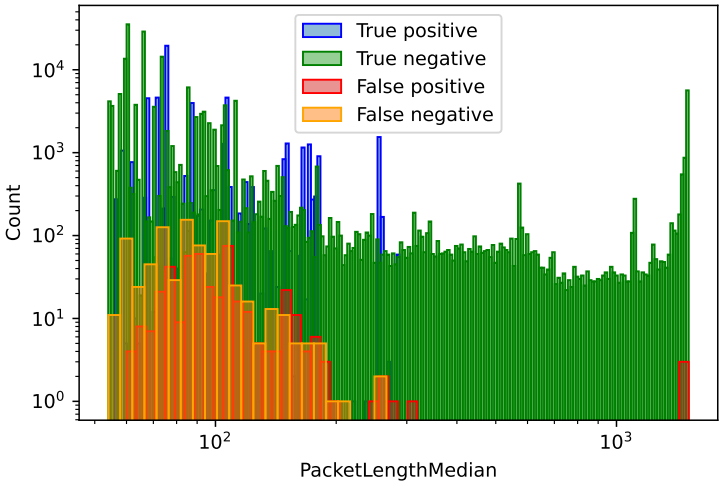

Figure 58: Histogram of the kNN classification distribution for the median packet length feature

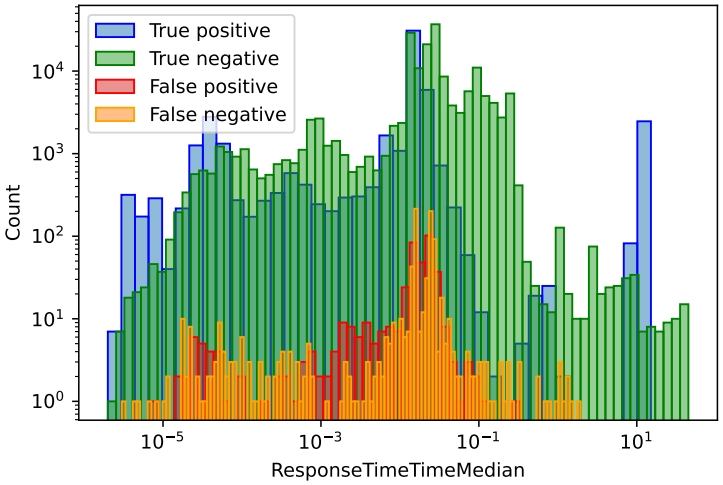

Figure 59: Histogram of the kNN classification distribution for the median response time feature

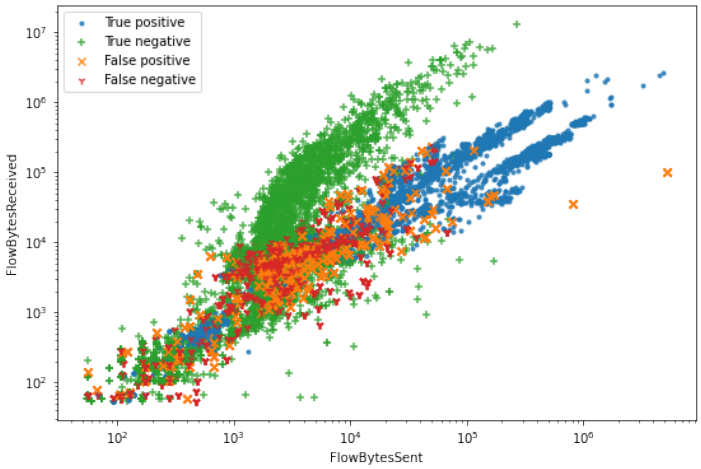

Figure 60: Scatterplot of the kNN classification for the flow bytes sent and received features

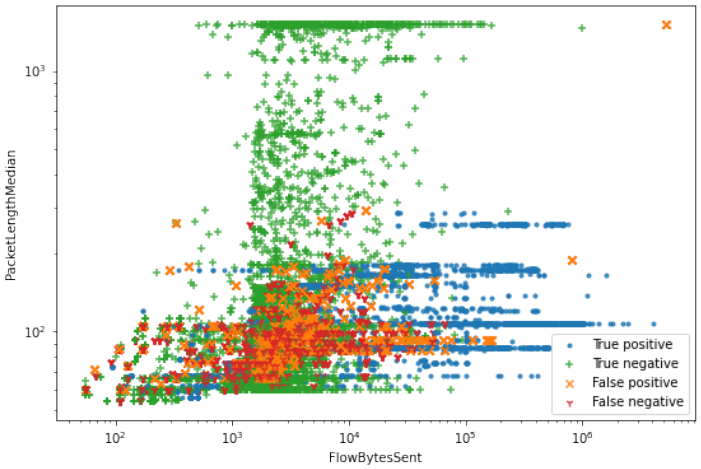

Figure 61: Scatterplot of the kNN classification for the flow bytes sent and packet length median features

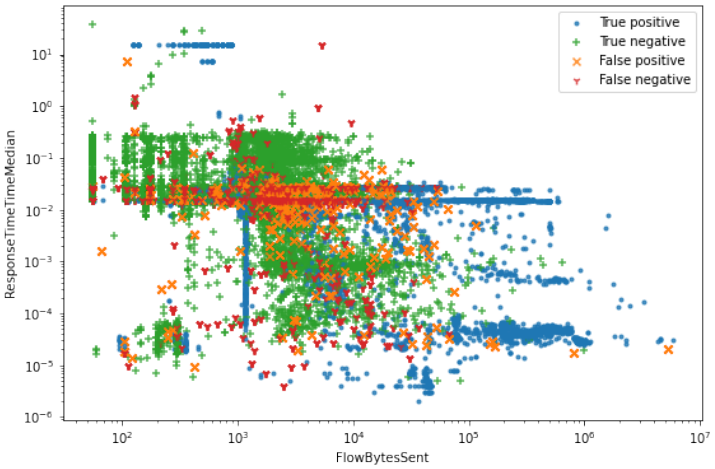

Figure 62: Scatterplot of the kNN classification for the flow bytes sent and median response time features

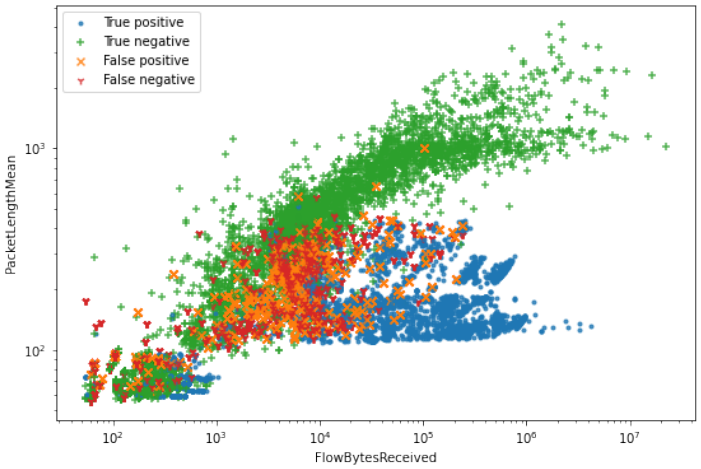

Figure 63: Scatterplot of the kNN classification for the flow bytes received and packet length mean features

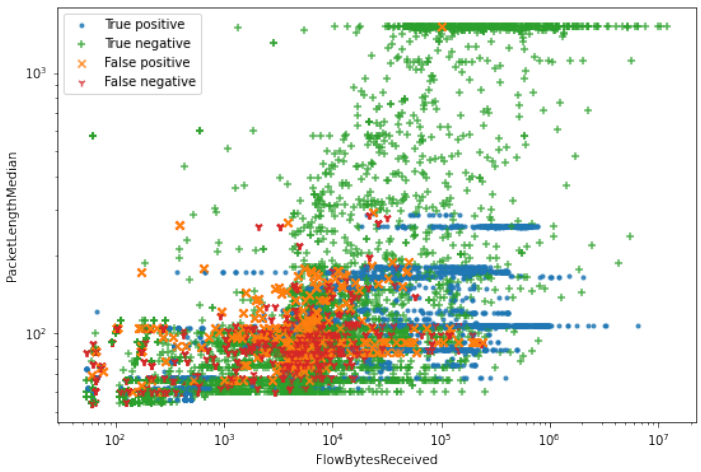

Figure 64: Scatterplot of the kNN classification for the flow bytes received and median packet length features

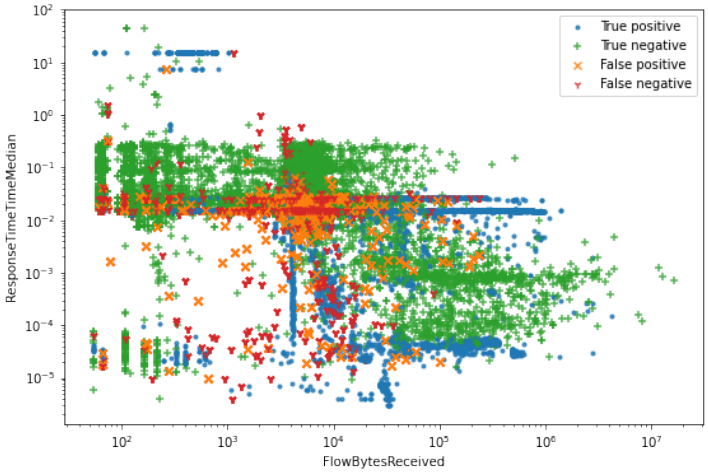

Figure 65: Scatterplot of the kNN classification for the flow bytes received and median response time features

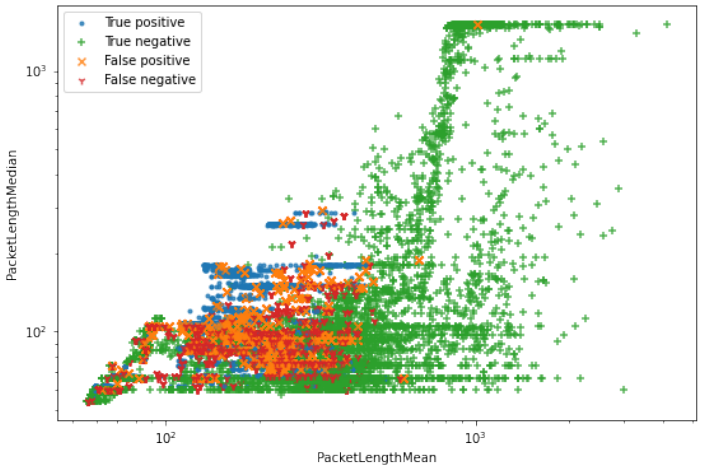

Figure 66: Scatterplot of the kNN classification for the mean packet length and median packet length features

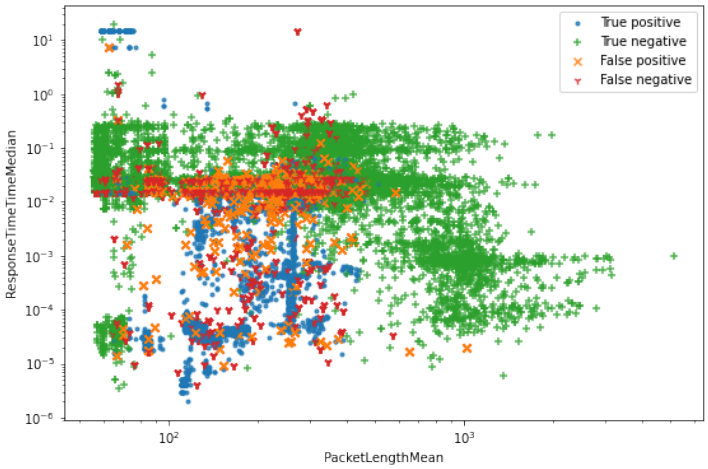

Figure 67: Scatterplot of the kNN classification for the mean packet length and median response time features

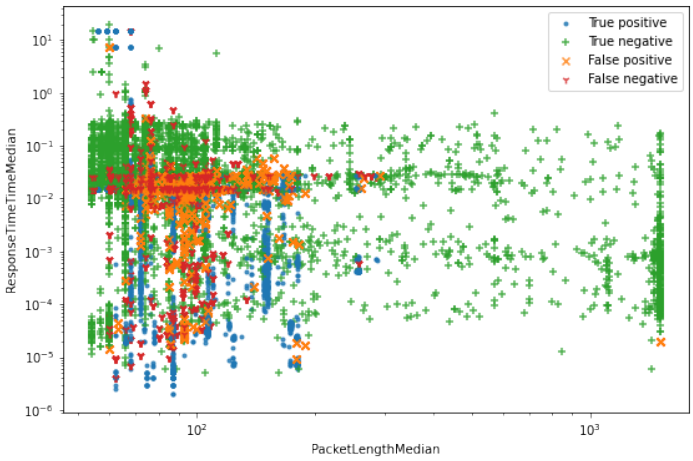

Figure 68: Scatterplot of the kNN classification for the median packet length and median response time features