



**University of  
Zurich<sup>UZH</sup>**

# **An Explorative Research into Encrypted Network Traffic Analysis**

*Lenz Baumann  
Zurich, Switzerland  
Student ID: 10-741-577*

Supervisor: Bruno Rodrigues, Prof. Dr. Burkhard Stiller  
Date of Submission: July 18, 2019



# Abstract

In daily life, more and more communication is conducted via digital devices such as mobile phones and computers. Every digital interaction between two entities, however, leaves traces in the network it traverses. Finding these traces, identifying source and target destination and extracting content, has become more and more important. Today, most digital transmissions are encrypted and it has become difficult to extract information from network traffic data. Nevertheless, the desire to gain access to as much information as possible remains a core-interest of private companies and law enforcement, but also the criminal world. This independent study investigates the possibilities and limitations of network traffic data in an explorative way. It examines the fundamental problems in network traffic data and explores how data can be generated data in a way, that circumvents problems such as categorical bias and labeling. It further explores how much discriminating power lies within network traffic data regarding the classification of the originating device and application. The features identified in the network data are then evaluated through descriptive statistics and machine learning. Finally, all findings are aggregated and used to design a network traffic classifier.



# Contents

|                                                           |          |
|-----------------------------------------------------------|----------|
| <b>Abstract</b>                                           | <b>i</b> |
| <b>1 Introduction</b>                                     | <b>1</b> |
| 1.1 Motivation . . . . .                                  | 1        |
| 1.2 Description of Work . . . . .                         | 2        |
| 1.3 Thesis Outline . . . . .                              | 2        |
| <b>2 Background and Related Work</b>                      | <b>3</b> |
| 2.1 Virtual Private Tunnel (VPN) . . . . .                | 3        |
| 2.2 Traffic Dissector . . . . .                           | 3        |
| 2.3 PCAP . . . . .                                        | 4        |
| 2.4 Flows and Flowmetering . . . . .                      | 4        |
| 2.5 Cramers V . . . . .                                   | 5        |
| 2.6 Decision Tree (DT) . . . . .                          | 6        |
| 2.7 Support Vector Machine (SVM) . . . . .                | 6        |
| 2.8 Related Work . . . . .                                | 7        |
| <b>3 Scenario</b>                                         | <b>9</b> |
| 3.1 Fundamental Problem of Network Traffic Data . . . . . | 9        |
| 3.2 Evaluation Scenario . . . . .                         | 11       |
| 3.3 Feature Generation . . . . .                          | 13       |

|          |                                                    |           |
|----------|----------------------------------------------------|-----------|
| <b>4</b> | <b>Evaluation</b>                                  | <b>19</b> |
| 4.1      | Source and Destination Ports . . . . .             | 19        |
| 4.2      | IP addresses . . . . .                             | 25        |
| 4.3      | Temporal features . . . . .                        | 30        |
| <b>5</b> | <b>Discussion</b>                                  | <b>33</b> |
| 5.1      | Dataset . . . . .                                  | 33        |
| 5.2      | Feature Extraction and Selection . . . . .         | 34        |
| 5.3      | Application- vs. Device-specific Traffic . . . . . | 35        |
| 5.4      | Conclusion . . . . .                               | 36        |
| 5.5      | Future Work . . . . .                              | 37        |
|          | <b>Bibliography</b>                                | <b>39</b> |
|          | <b>List of Figures</b>                             | <b>42</b> |
|          | <b>List of Tables</b>                              | <b>44</b> |
|          | <b>Appendix</b>                                    | <b>45</b> |

# Chapter 1

## Introduction

### 1.1 Motivation

The temporal analysis of network events can provide insights on whether a specific behavior is normal or not. Once a common pattern is identified, deviations may indicate anomalies that can be identified, interpreted and categorized [17][16]. Such analysis has great relevance in the digital forensics area, to indicate evidence of crimes and to support the investigator [15]. However, the variety of devices that generate network traffic, as well as the different network protocols and use cases, make such analysis a complex task. In the scenario where most of the generated traffic is encrypted, the task of such an analysis is to provide an investigator with evidence that indicates the existence of a particular activity using network traffic patterns, anomalies, and correlations. In most cases, the data situation of forensic analysts is not that of a complete macroscopic network: Even when large amounts of data are seized, the data is captured at one specific node within a network and therefore only contains information about this node and its neighbors, hence only a fraction of the overall network [16]. Therefore, statistical network analysis and macroscopic time-dependent analysis are not possible anymore, as they require information about all nodes and the edges that connect them [25].

Nevertheless, data contains information, that can be extracted and analyzed. Especially interesting for investigation purposes are names, locations, the use of TOR<sup>1</sup> and information about the device or application used. The focus of this work, therefore, shifted from the temporal network analysis towards the analysis of network traffic in general. Such task, however, remains a difficult one and various challenges arise from it, such as encryption, the vast amount of noise in the data and the absence of labels, to name only a few. This independent study, therefore, seeks to explore the different possibilities and limitations of network traffic data for information extraction and classification.

The main questions driving this independent study hence are: What information can be found within data? Does the data contain enough discriminating power to differentiate between background noise and application traffic, different devices and application types?

---

<sup>1</sup>The **O**nion **R**outing, is a program that protects the user's identity by bouncing communications around a distributed network of relay, run by volunteers all around the world. [41]

Finally, to achieve this goal, an architecture for a network classifier is proposed, that uses the insights gained and to draft an approach that increases the information extraction on the data, while at the same time circumnavigates the challenges connected to the data.

## 1.2 Description of Work

It is assessed, which discriminating features and pieces of information, network traffic data contains (*e.g.* protocol types, speed of transmission, IP address) and what classification is possible based on it. For the scope of this work, it is additionally assumed that the traffic is encrypted and only the headers' information can be accessed. Based on this assumption, the following three aspects are investigated in this work:

1. How can network traffic data be generated and labeled?
2. What pieces of information does the data hold, regarding device and application types and how strong are those features?
3. How would a system for network traffic analysis look like, based on the findings of the previous questions?

All analysis was done using the python-based [31] frameworks pandas [30] and scikit-learn [34]. All computing and visualizing was done in Google-Colab<sup>2</sup> [19].

## 1.3 Thesis Outline

Chapter 2 *Background and Related Work* gives a summary of the tools and technics used in this work. Section 2.8 *Related Work* gives a short overview of existing approaches dealing with problems related to network traffic classification and information extraction. In chapter 3 *Scenario* different possibilities of generating network traffic are discussed and elaborated on. A simple, but useful approach to generate network traffic is suggested and described. Chapter 4 *Evaluation* tackles the main questions of how to extract and select features within network traffic data and evaluates their performance. Finally, chapter 5 *Discussion* summarizes the findings of this work, draws conclusions and presents an outlook: Section 5.5 *Future Work* proposes an approach to create a system for information extraction and network classification.

---

<sup>2</sup>The associated IPython notebook can be found under <https://colab.research.google.com/drive/1gLccvi8cRhtFFrW3RGD7eawP1MYFqe8-> [7]

# Chapter 2

## Background and Related Work

In this chapter, the concepts, and technologies used in this study as well as related work, are explained and described.

### 2.1 Virtual Private Tunnel (VPN)

A VPN creates a tunnel between two endpoints and allows to extend a network across a public network. Both endpoints can send and receive data through the tunnel and across shared or public networks [48]. The VPN, therefore, hides an endpoints original IP address within the public network and anonymizes its identity and location. In some cases, the VPN additionally encrypts the original header sections of the network packets and therefore anonymizes the exchanged data completely [29]. One example of a VPN application is Wireguard. Wireguard is a freeware opensource VPN software that encrypts network traffic using the Noise protocol framework, Curve25519, ChaCha20, Poly1305, BLAKE2, SipHash24, and HKDF. It is highly performant, secure and straightforward to set up and to configure [2].

### 2.2 Traffic Dissector

A network traffic dissector is a program, that allows decomposing a data package following a certain set of characteristics [23]. With the help of a dissector, a certain protocol can be understood and its packets decomposed [5].

A popular and widely-used traffic dissector is Wireshark. Wireshark is a network traffic dissector and analyzer, allowing live-capture as well as offline analysis of network traffic, flow extraction, and explorative data analysis. While Wireshark possesses a graphical user interface to view the data, other network analysis tools *e.g.*, for packet capture and gathering exist, such as tcpdump and tshark. In this study, tshark was used for packet dumping and storing.

## 2.3 PCAP

The PCAP file format was developed in 1994 at the University of California at Berkeley in conjunction with Libcap, an open-source library, that provides a high-level interface to network packet capture systems [18]. Since then, it has become the standard for network capture files in the open-source community [1] and is used in applications, such as Moloch or Wireshark. Libpcap can be used to read and write PCAP files. It is available as a C library, however, wrappers have been written for various other programming languages including Java and Python.

Table 2.1 shows the general structure of a PCAP file. The global header contents are shown in table 2.2. The packet header contents are shown in table 2.3.

|               |               |             |               |             |     |
|---------------|---------------|-------------|---------------|-------------|-----|
| Global Header | Packet Header | Packet Data | Packet Header | Packet Data | ... |
|---------------|---------------|-------------|---------------|-------------|-----|

Table 2.1: Structure of a PCAP file [1]

|                      |                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------|
| <b>magic_number</b>  | an integer used to detect the file format and the byte ordering                                |
| <b>version_major</b> | an integer used to describe the major version number                                           |
| <b>version_minor</b> | an integer used to describe the minor version number                                           |
| <b>thiszone</b>      | an integer used to describe the correction time between GMT and the local time zone in seconds |
| <b>sigfigs</b>       | the accuracy of timestamps, always set to zero in practice                                     |
| <b>snaplen</b>       | the snapshot length for the capture                                                            |
| <b>network</b>       | the link-layer header type of the packet headers that follow                                   |

Table 2.2: Contents of the PCAP global header [1]

|                 |                                                                      |
|-----------------|----------------------------------------------------------------------|
| <b>ts_sec</b>   | the date and time when the packet was captured as a UNIX timestamp   |
| <b>ts_usec</b>  | the microseconds when the packet was captured as an offset to ts_sec |
| <b>incl_len</b> | the number of bytes of packet data captured                          |
| <b>orig_len</b> | the number of bytes of data in the original packet                   |

Table 2.3: Contents of the PCAP packet header [1]

## 2.4 Flows and Flowmetering

A flow is a sequence of packets sent from a particular source to a particular unicast, anycast, or multicast destination that the source desires to label as a flow. A flow can consist of all packets in a specific transport connection or a media stream [32]. Flows aggregate multiple packets and form an entity, that includes various aggregated measures such as *e.g.* duration, the number of packets exchanged and type of protocol.

The Internet Protocol Flow Information Export (IPFIX) format is a universal standard of export for Internet Protocol flow information and a widely used format to store flows.

IPFIX considers a flow to be any number of packets observed in a specific timeslot and sharing several properties, *e.g.* same source, same destination, same protocol [47].

A tool that can extract flows from captured packets is called a flowmeter. Yaf is a suite of tools to do flow metering. It takes raw packets from network transmissions in the form of a pcap-file as input, aggregates them into bidirectional flows and outputs these flow records in the form of IPFIX files [12].

To convert IPFIX files into simpler file formats, an IPFIX mediator is needed. Super Mediator is an IPFIX mediator, that collects and filters yaf output to various IPFIX collecting processes, csv-files, json-files, and MySQL databases [11]. It can, therefore, be used to transform IPFIX data into csv format.

## 2.5 Cramers V

Cramer's V is a symmetric  $\chi^2$ -based measure for assessing the association between two or multiple nominal variables if at least one of them has more than two classes. Cramer's V calculates scores between 0 (no association) and 1 (full association). Because Cramer's V is always positive, no statement about the direction of the association can be made [46].

Cramers V is calculated as follows:

$$V = \sqrt{\frac{\chi^2}{n * (k - 1)}} \quad (2.1)$$

where n and k are defined as:

$$n = \text{total samples} \quad k = \min(\text{amount of rows, amount of columns})$$

The *corrected Cramers V* ( $\tilde{V}$ ), as proposed by Bergsma, introduces a bias correction for cases with large amounts of attributes, but small sample sizes [9]. The corrected Cramers V is calculated the following way:

$$\tilde{V} = \sqrt{\frac{\max(\chi^2, 0)}{\min(\tilde{r} - 1, \tilde{c} - 1)}} \quad (2.2)$$

with values for  $\tilde{r}$  and  $\tilde{c}$  defined as:

$$\tilde{r} = r - \frac{1}{n-1}(r-1)^2 \quad \tilde{c} = c - \frac{1}{n-1}(r-1)^2$$

## 2.6 Decision Tree (DT)

Decision trees are simple learning methods, that work by segmenting the prediction space recursively [24], based on a measure of information gain. They can be used for regression and classification problems.

In a decision tree, each internal node is labeled with an input feature. Each arc originating from a node is labeled with a value or a range of values that the input feature might have. The leaf nodes correspond to the target variable that should be predicted. A tree can then be trained by splitting the training dataset recursively along the tree until the output is univocal or remains unchanged after further splitting. Figure 2.1 shows a decision tree for predicting the source of network traffic flows. Each node shows the quality of the split ("gini") and the number of samples that were separated with each split ("values").

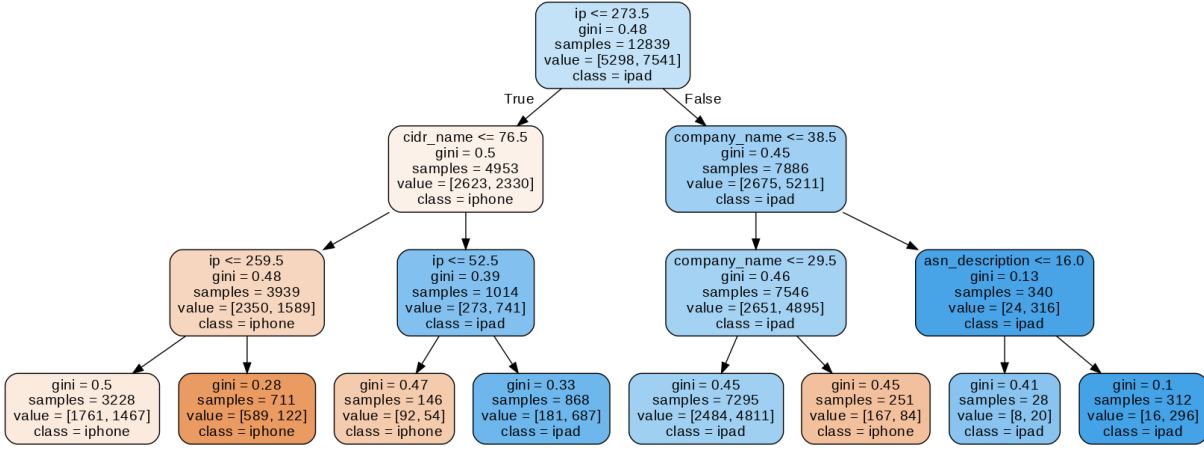


Figure 2.1: A tree showing the classification of network traffic into iPad and iPhone generated traffic.

## 2.7 Support Vector Machine (SVM)

A Support Vector Machine is a generalization of a simple and intuitive technique called the Maximal Margin Classifier (MMC) [24]. An MMC works by determining a multidimensional plane, called a hyperplane, to separate classes in a multidimensional feature space. This hyperplane is calculated in such a way that the margin, the distance between the hyperplane and the points closest to the hyperplane, is maximal. Since only these closest points influence the location of the hyperplane - the hyperplane is *supported* by them - they are called *support vectors*.

A SVM adds to this in two ways. First, it allows for some data points to lie on the wrong side of the separating hyperplane, if it improves the overall classification. Second, it allows for non-linear hyperplanes by adjusting the distance measure used to determine the margin. In practice, this is calculated using the inner product of the support vectors. The way this inner product, called the kernel, is computed, can now be adjusted depending on

the data that needs to be classified. Linear (see figure 2.2), polynomial and radial kernels, for example, are possible.

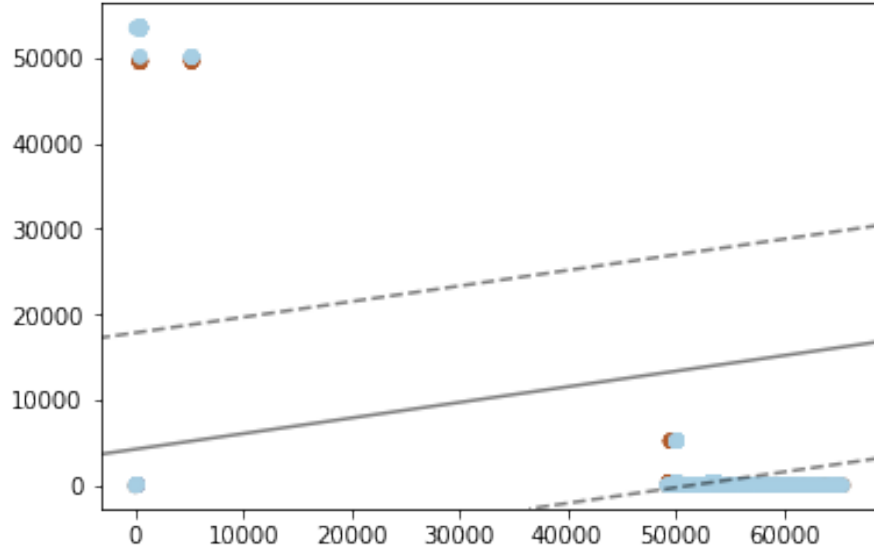


Figure 2.2: Support Vector Machine using a linear decision function (solid line). The margin is represented with the two dotted lines.

## 2.8 Related Work

Encrypted network traffic analysis and network traffic classification is a major area of research, that has produced hundreds of new approaches within the last couple of years. Earlier works mainly focus on traditional machine learning approaches such as Naive Bayes [26], Support Vector Machines [43], Decision Trees and Random Forrest [43] for classification and rely on manual feature extraction and selection based on raw packets [28], different amounts of TCP and UDP flow features [26][27][43][22] and other network layer sessions [45]. More recent approaches, though, focus on neural networks not only for classification but also for feature extraction (representation learning) [28]. Stacked Autoencoders based on Convolutional Neural Networks or Recurrent Neural Networks are promising approaches to create representations of flows and packets, that are not detectable by the human eye or by other feature extraction approaches. While the approaches per se differ greatly, the results obtained, do not differ much: Traditional as well as more recent approaches are able to distinguish well ( 95% accuracy in average) [26][27][43] between different applications, VPN<sup>1</sup> or non-VPN traffic [43] or different application layer protocols [13].

For the generation of the data, most approaches either use a proprietary dataset that contains multiple or single network traffic captures from different places [8][33][10][22][38][13] or a device-based VPN set-up [4]. Some approaches also use Adversarial Neural Networks to oversample for underrepresented classes [43].

<sup>1</sup>Virtual Private Network see section 2.1

By looking at the preparation part of the data, it can generally be observed, that the labeling part, necessary for every supervised machine learning approach, creates a lot of problems (unbalanced categories and wrong or a small number of labels). This is not surprising as the approaches all choose huge amounts of data to create working classifiers. Therefore for the labeling of the data, only automated approaches are a real option. This automated labeling process is done in most cases, by using Deep Packet Inspection tools [22][27][49][8].

For the classification most works use supervised machine learning approaches [26][27][43][8][33][10][22][38][13]. Only few works such as Rezaei and Liu and Zhang et al. use semi-supervised approaches [33][49], others like Hoechst et al. choose unsupervised approaches [22].

# Chapter 3

## Scenario

This chapter elaborates on the problems, limitations as well as possible solutions necessary to answer the questions stated in section 1.2. The generation of network traffic, the construction of a feature set and the fundamental problem of labeling network traffic, is discussed in the following sections.

### 3.1 Fundamental Problem of Network Traffic Data

In the field of network traffic analysis, datasets are often not available, difficult to generate and do usually not fit contexts other than the one they were created for [37]. Multiple reasons are responsible for this:

1. **Context:** Network data is captured in a specific environment and with a specific question in mind. The type of traffic captured, whether it is encrypted or not and the way the packets get stored, differs from use case to use case. Therefore, data that was gathered in a specific context is most likely not easily transferable to another context and another question [37].
2. **Source:** Even if network traffic data consists of well defined and structured packets, without access to the device that produced the traffic, there is no way to determine the type of device or the application a certain packet originates from [20].
3. **Economy:** Parties, that have the possibility and the means to capture network data on a large scale, are often not willing to authorize others to use the data as well. Either because the data contains sensitive information or because the data is used for internal purposes and sharing it would lead to competitive disadvantages.
4. **Artificial Generation:** Creating artificial network traffic is possible but infeasible once a certain amount of representability needs to be assured. Even if a sophisticated and expensive set-up of physical and virtual devices can create data that comes close to data seen in the real world, it is still questionable whether the generated data is a good representation of the reality and whether it captures the essential patterns [42].

In addition to these network traffic-related problems, the general problems of generating data and building a dataset come into play. Especially in the context of supervised learning and in combination with data labeling, various challenges arise. For the domain of network traffic, they can be listed as follows:

1. **Unbalanced categories:** The existence of evenly balanced values poses a central requirement for any kind of supervised machine learning. Learning on unbalanced datasets leads to biased results; among other reasons because of statistical requirements that are not given in biased data [14][49]. To mitigate the problem of unbalanced datasets, several options are available such as oversampling, undersampling or the aggregation of categories into more evenly distributed values. In the case of network traffic, the part of the traffic that should be identified, usually a minority of all the data, is outweighed by a big amount of other traffic and noise. Creating an evenly balanced dataset of size, big enough to use for machine learning is difficult with normal financial and temporal resources.
2. **Unlabeled Data:** Network packets per se are not labeled. And while it is possible to extract the type of protocol and other pieces of information about the packet from its header, it is not possible to simply infer abstract categories such as application and device types from the data itself. Solutions to this problem are either deep packet inspection or to access the source device/application that generated the traffic [20][49].
3. **Noisy Data:** The traffic relevant for classification is a minority surrounded by a majority of irrelevant background noise. To be able to accurately and efficiently learn from traffic, it is required to find a way to distinguish patterns from noise. The small amount of network traffic that is aimed to get classified, is normally not big enough to train a classifier [49].

The above-stated difficulties are a clear sign, that the establishment of a labeled, clean and balanced ground truth for any kind of classification of network traffic, is extremely difficult. And it is obvious, that this set of problems poses serious limitations not only to research but also to machine learning approaches related to network traffic classification in general.

Confronted with these problems and difficulties, the goals of this work, as stated in section 1.2, are rephrased and specified:

1. **Generation & Labeling:** Finding reliable ways to generate labeled and unbiased network traffic data, finding solutions to generate, capture and store network traffic and determine mechanisms to label network packets according to device or application type.
2. **Feature Examination:** Extracting as much information from the network data as possible and determining features that have enough discriminating power to allow the classification of network traffic. In the focus of the classification are device and application types.

In the following sections both goals are examined and solutions as well as problems, theoretically and practically, are described.

## 3.2 Evaluation Scenario

Based on the two above-mentioned goals of this work, two requirements arise to generate a balanced and exhaustive dataset that allows classifying devices and applications:

1. The traffic must be allocatable to a specific device. The traffic captured during the time of connection with a specific device can then be enriched with labels that indicate at least the device-specific information.
2. The traffic must be allocatable to a specific application. To be able to create labels for packets of specific applications, the applications must be isolated from *e.g.* operating system specific network traffic.

One way of achieving both requirements and enabling labeling by design within the data generation process is to use a VPN set-up. With the help of a VPN, all traffic of a specific device can be tunneled through an endpoint and be captured and enriched with the right labels, such as information about device and application the traffic was generated by. However, there exist various drawbacks to this approach:

- Not all manufacturers allow the simulation of devices. While *e.g.* Android devices can be simulated on a computer and specific applications can be run isolated, other types of devices, such as *e.g.* Apple devices, cannot be simulated and real physical devices are required.
- This approach is neither scalable nor extendable: For each new device type, that needs to be incorporated in the dataset, the VPN setup has to be re-established. The same applies to the labeling of any application: If a new application should be labeled, it has to be installed on all devices and the VPN set-up has to be re-run.

Nevertheless, for explorative purposes, the VPN-based approach seems to be promising, as it is relatively simple and easy to create and it accounts for both requirements.

Additionally, it can be observed, that the generation of isolated application-specific network traffic is more difficult than the generation of isolated device-specific network traffic. The reason for this problem lies in the fact, that it is impossible to fully control all of the devices' processes. And the launching of an isolated application and the generation of network traffic originating therefrom becomes impossible. The only possibility to account for application-specific network traffic in the VPN set-up is to create traffic by using one application more frequent than others and therefore increasing the probability of each packet to originate from that specific application. A hundred percent probability can, however, not be achieved. The labeling of packets according to the application that generated it, is therefore not as accurate, as according to the device/device type.

As application-specific traffic generation and labeling would require even more time and resources, it was not pursued further in this work. The labeling was done only on the device type.

The exact final set up of the network traffic generation process was achieved with the use of a RaspberryPi 3B+<sup>1</sup> and the opensource VPN software *Wireguard*<sup>2</sup> [2] as visualized in figure 3.1.

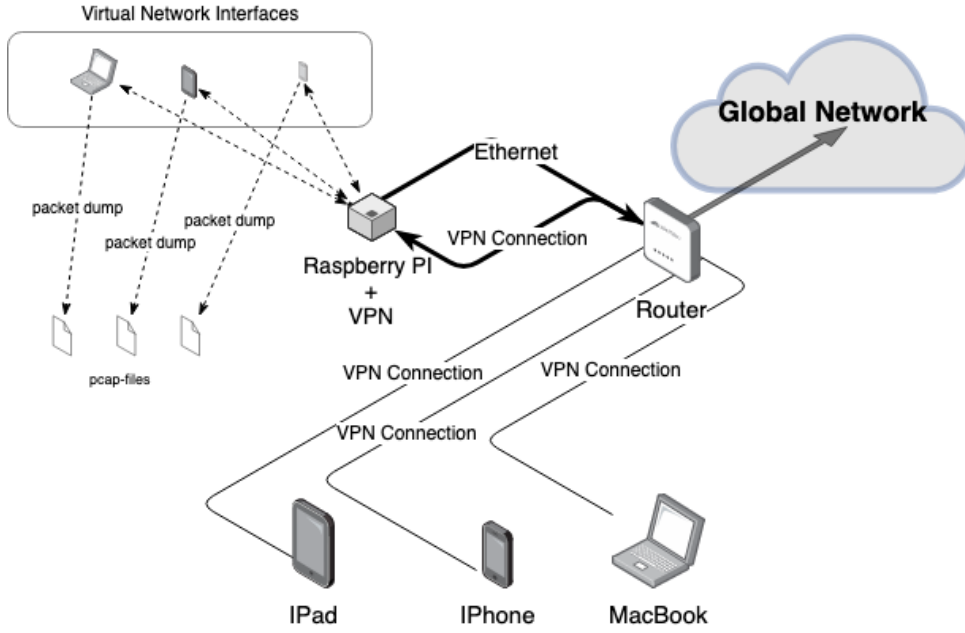


Figure 3.1: Network traffic capturing set-up using VPN and local network

In the set-up used in this study, a VPN endpoint was set up as a virtual interface on the Raspberry Pi. The Raspberry Pi was then configured in the network by connecting it through the ethernet-port with the local router. By installing the *Wireguard* mobile application on different devices (iPad, iPhone, MacBook), a virtual private tunnel was established between these devices and the Raspberry Pi. Hence, all traffic was routed from the devices to their designated destiny via the virtual interface on the Raspberry Pi. Using the command-line tool *tshark*<sup>3</sup>, the network traffic could be collected and saved in the pcap-format. With this setup, it was possible to label each captured packet, with the device that it originated from.

Additionally two types of human-device interactions were captured: The "reduced activity"-interaction, that captured background traffic and as less application-specific traffic as possible in order to account for the general influence, application-based activity has, and the "extensive activity"-interaction, that generated traffic as it occurs in a normal interaction between user and device. In the "reduced activity"-dataset, the device was left untouched and all application-related activity was reduced as much as possible, while in the "extensive activity"-dataset all applications were used freely and independently.

<sup>1</sup>The Raspberry Pi is a small low cost credit-card sized single-board computer [39]. In the current set-up, a device of the third generation was used.

<sup>2</sup>cf. section 2.2

<sup>3</sup>cf. section 2.2

The result of the above set-up, are multiple pcap-files, representing each device, that was used to create the traffic. Device types used were an **iPhone 6s**, an **iPad 11' 2018** and a **MacBook Air Mid 2013**. As these device types are quite similar to each other, a fourth category was added to the dataset in the form of an unknown device<sup>4</sup>.

The two datasets "extensive activity" and "reduced activity" consisted of:

- **Extensive activity:** Created with normal interaction of device and user, containing four device types (iPad, Iphone, MacBook and unknown device). It included 3'124'335 packets (Mac: 2'634'614, Iphone: 334'006, iPad: 154'205, unknown: 1'510) and 23'009 flows (Mac: 10'120, Iphone: 7'541, iPad: 5'298, unknown: 50). The file had a size of 3.49 MB.
- **Reduced activity:** Created with reduced interaction of device and user, containing two device types (iPad, Iphone). It included 20'555 packets (iPad: 17'607, Iphone: 2'948) and 1'424 flows (iPad: 1'239, Iphone: 185). The file had a size of 0.22 MB.

### 3.3 Feature Generation

Instead of using single network data packets, flows<sup>5</sup> were chosen as basic entity for the dataset. To create flows from the captured network packets (pcap-files), an application called yaf was used. Yaf calculates the flows from pcap files and generates an IPFIX-file as output. This IPFIX-format was then transformed by super-mediator into a csv-file. Each flow included a variety of features as can be seen in table 3.1.

Table 3.1: Flow features extracted with yaf [12]

| Flow Feature | Description                                                                          |
|--------------|--------------------------------------------------------------------------------------|
| start-time   | Flow start time in ISO 8601 format, with milliseconds                                |
| end-time     | Flow end time in ISO 8601 format, with milliseconds                                  |
| dur          | Flow duration in fractional seconds                                                  |
| rtt          | Round-trip time estimate in milliseconds in decimal format                           |
| pro          | IP protocol identifier in decimal format                                             |
| sip          | Source IPv4 address in dotted-quad format<br>or IPv6 address in RFC 2373 format      |
| sp           | Source transport port in decimal format                                              |
| pkt          | Forward count in decimal format                                                      |
| oct          | Forward octet count in decimal format                                                |
| srcMacAddr   | Source MAC Address                                                                   |
| dip          | Destination IPv4 address in dotted-quad format<br>or IPv6 address in RFC 2373 format |
| dp           | Destination transport port in decimal format                                         |

*Continued on next page*

<sup>4</sup>see <https://pcap.honeynet.org.my/v1/> Accessed: 13.7.19

<sup>5</sup>cf. section 2.4

Table 3.1 – *Continued from previous page*

| Flow Feature | Description                                                |
|--------------|------------------------------------------------------------|
| rpkt         | Reverse packet count in decimal format                     |
| roct         | Reverse octet count in decimal format                      |
| destMacAddr  | Destination MAC Address                                    |
| iflags       | Forward first-packet TCP flags                             |
| riflags      | Reverse first-packet TCP flags                             |
| uflags       | Forward nth-packet TCP flags union                         |
| ruflags      | Reverse nth-packet TCP flags union                         |
| isn          | Forward initial TCP sequence number in hexadecimal format. |
| risn         | Reverse initial TCP sequence number in hexadecimal format  |
| tag          | Forward first-packet 802.1q VLAN tag in hexadecimal format |
| end          | Reason for End of the Flow                                 |

To determine the amount of association each feature has with the target variable "device", the corrected Cramer's  $\tilde{V}$  was calculated. In table 3.5 it can be observed, that many attributes show i) low association with the target (*e.g.* iflags or srcMacAddr) or do have a high  $\tilde{V}$  value, but do ii) not have real "variance" of their categories (*e.g.* in or out). Both types of attributes (i & ii) have no real association with the target variable and were removed in the further process<sup>6</sup>.

Table 3.2: Corrected Cramers  $\tilde{V}$  association with the features and the amount of categories each feature includes.

|            | Reduced Activity |            | Extensive Activity |            |
|------------|------------------|------------|--------------------|------------|
| Features   | $\tilde{V}$      | Categories | $\tilde{V}$        | Categories |
| start-time | 0.735358         | 10559      | 0.703251           | 5859       |
| end-time   | 0.754625         | 9906       | 0.713003           | 5699       |
| dur        | 0.630377         | 4505       | 0.753976           | 2873       |
| rtt        | 0.610114         | 520        | 0.824389           | 482        |
| pro        | 0.582199         | 5          | 0.581474           | 6          |
| sip        | 0.980464         | 251        | 0.992433           | 96         |
| sp         | 0.716453         | 7116       | 0.716085           | 4673       |
| pkt        | 0.595593         | 353        | 0.626299           | 299        |
| oct        | 0.667836         | 3255       | 0.702730           | 2134       |
| at         | 0.051638         | 2          | 0.054633           | 2          |
| srcMacAddr | 0                | 1          | 0                  | 1          |
| dip        | 0.640425         | 831        | 0.670395           | 514        |
| dp         | 0.629921         | 921        | 0.622563           | 306        |

*Continued on next page*

<sup>6</sup>Throughout this work, a measure for the association of categorical variables is used. This is due to the fact, that most variables are indeed categorical. Variables with a metric level of measurement, that would allow more precise association measures, such as *e.g.* dur, rtt and pkt are not only assessed on the bases of  $\tilde{V}$  (cf. section 4.3). However, using  $\tilde{V}$  with metric variables does not lead to wrong conclusions per se, but poses some limitations for interpretation

Table 3.2 – *Continued from previous page*

| Features    | Extensive Activity |            | Reduced Activity |            |
|-------------|--------------------|------------|------------------|------------|
|             | $\tilde{V}$        | Categories | $\tilde{V}$      | Categories |
| rpkt        | 0.193376           | 370        | 0.148734         | 269        |
| roct        | 0.683263           | 3754       | 0.746144         | 2341       |
| ra          | 0.708018           | 3          | 0.579873         | 4          |
| destMacAddr | 0                  | 1          | 0                | 1          |
| iflags      | 0.247301           | 11         | 0.096494         | 10         |
| uflags      | 0.117550           | 24         | 0.079645         | 18         |
| riflags     | 0.152120           | 11         | 0.066941         | 10         |
| ruflags     | 0.136981           | 21         | 0.117003         | 17         |
| isn         | 0.529636           | 6857       | 0.541835         | 3559       |
| risn        | 0.435249           | 5139       | 0.434214         | 2849       |
| in          | 0.999957           | 2          | 0.999914         | 2          |
| out         | 0.999957           | 2          | 0.999914         | 2          |
| tag         | 0.999957           | 2          | 0.999914         | 2          |
| app         | 0.999957           | 2          | 0.999914         | 2          |
| tos         | 0.586093           | 8          | 0.579486         | 5          |
| end         | 0.117989           | 4          | 0.088899         | 4          |
| collector   | 0                  | 1          | 0                | 1          |

Based on the strength of association in table 3.5 and on the variance of the categories, the final dataset was created using python [31] and the pandas framework [30]. The exact procedure is visualized and described in figure 3.2.

### IP address

The IP address is a special feature, as it can be used for both, classification and information extraction:

- For classification, because it has a high association with the target variable and contains most likely high discriminating power.
- For information extraction, because it refers to the outside party the flow is established with. Using the external IP address in combination with DNS information, more information about the external party can be obtained and the device, the flow was established by, can be characterized further.

By using the python package *ipwhois* [21], this DNS related information could be obtained. Table 3.3 explains each field and table 3.4 contains a sample set of the DNS related features.

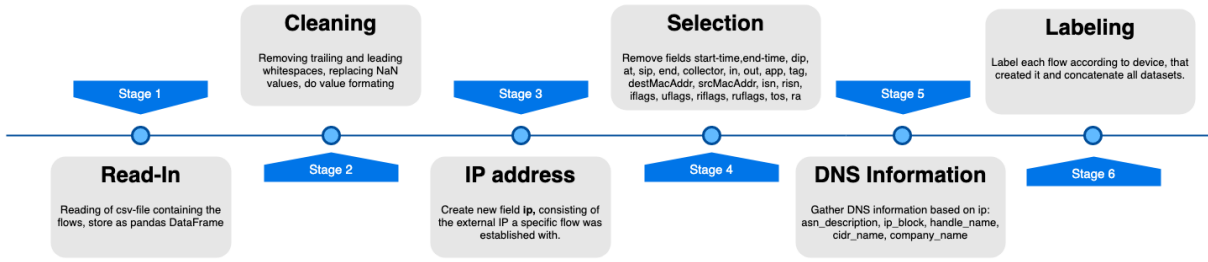


Figure 3.2: Preparation and cleaning of data for analysis

- **Read-In:** The csv file containing the flows was read with python into a pandas dataframe.
- **Cleaning:** The data was cleaned (Removing trailing and leading whitespaces, replacing 0 values, value formatting)
- **IP address:** The two fields 'sip' and 'dip' were replaced by a new field 'ip' consisting of the external IP a specific flow was established with. The local ip of the device within the subnet was discarded.
- **Selection:** The fields 'start-time', 'end-time', 'dip', 'at', 'sip', 'end', 'collector', 'in', 'out', 'app', 'tag', 'destMacAddr', 'srcMacAddr', 'isn', 'risn', 'iflags', 'uflags', 'rflags', 'rflags', 'tos', 'ra' were removed as the variance and their association with the target variable was too low.
- **DNS Information:** Based on the newly created field 'ip', the dns information was requested. From the information the DNS server returned information about the company, the ip-block, and the description of the dns entry was added to the dataset. These features are more thoroughly discussed below.
- **Labeling:** The datasets of the different devices were labeled accordingly and merged into one dataset, representing the final dataset.

Using the IP address and coupling it with external DNS information, has several advantages:

1. The frequency of calls to specific IP addresses vary among different devices. By extracting patterns from these frequencies allows distinguishing between different devices.
2. Using the IP block, the company name or other information obtained from the DNS server, allows to aggregate multiple IP addresses.
3. Using DNS information on a specific IP address, it is possible to reveal device types (*e.g.* Apple Servers) and applications (*e.g.* Telegram or WhatsApp servers) by simply extracting information.

| feature name    | feature description                                                                     |
|-----------------|-----------------------------------------------------------------------------------------|
| block           | Network routing block an IP address belongs to                                          |
| cidr_name       | The identifier assigned to the network registration for an IP address                   |
| handle_name     | Unique identifier for a registered object                                               |
| asn_description | The official registered autonomous system number of the internet service provider (ISP) |
| company_name    | The contact name                                                                        |

Table 3.3: DNS related features and description [21]

| ip              | cidr_name                  | handle_name                       | company_name             |
|-----------------|----------------------------|-----------------------------------|--------------------------|
| 23.205.182.44   | AIBV                       | NET-23-205-176-0-1                | Akamai International, BV |
| 17.253.54.125   | APPLE-WWNET                | NET-17-0-0-0-1                    | Apple Inc.               |
| 149.154.167.92  | Telegram_Messenger_Network | 149.154.164.0 - 149.154.167.255   | Telegram Messenger Inc   |
| 149.154.167.92  | Telegram_Messenger_Network | 149.154.164.0 - 149.154.167.255   | Telegram Messenger Inc   |
| 217.146.165.138 | AKAMAI-AANP                | 217.146.165.128 - 217.146.165.159 | NTS NOC                  |
| 17.253.52.125   | APPLE-WWNET                | NET-17-0-0-0-1                    | Apple Inc.               |
| 8.8.8.8         | LVL-T-GOGL-8-8-8           | NET-8-8-8-0-1                     | Google LLC               |
| 23.211.4.189    | AIBV                       | NET-23-211-4-0-1                  | Akamai International, BV |
| 104.16.205.65   | CLOUDFLARENET              | NET-104-16-0-0-1                  | Cloudflare, Inc.         |
| 8.8.8.8         | LVL-T-GOGL-8-8-8           | NET-8-8-8-0-1                     | Google LLC               |

Table 3.4: Sample Flows excluding the IP block and the ASN description

By measuring the association between the new variables block, cidr\_name, handle\_name and company\_name, it can be seen in table ??, that they contain discriminating power towards the target variable slightly lower than the IP address.

Table 3.5: Corrected Cramers V association with the features and the amount of categories each feature includes.

| Features        | Reduced Activity |            | Extensive Activity |            |
|-----------------|------------------|------------|--------------------|------------|
|                 | $\tilde{V}$      | Categories | $\tilde{V}$        | Categories |
| ip              | 0.611387         | 828        | 0.659343           | 511        |
| asn_description | 0.361512         | 76         | 0.417038           | 65         |
| block           | 0.478128         | 214        | 0.505139           | 147        |
| cidr_name       | 0.424203         | 140        | 0.446316           | 99         |
| handle_name     | 0.478128         | 214        | 0.505139           | 147        |
| company_name    | 0.378556         | 75         | 0.446098           | 62         |



# Chapter 4

## Evaluation

In the following chapter, multiple features are discussed and their discriminating and informative power is explored, through descriptive and explorative statistics as well as machine learning algorithms.

The main goal of this section is to determine, whether the gathered data includes enough discriminating power to classify different devices and enough informative power to get additional information about applications and devices from the network traffic data flows.

The features that stood out either in terms of association with the target variable (cf. section 3.3) or in terms of the additional information hidden within, were the **ports**, the **IP address** and its related features and the **temporal features**. The importance of those features for the classification of network traffic is supported also by other works [28][38].

### 4.1 Source and Destination Ports

#### Corrected Cramers V

Calculating Corrected Cramer's V ( $\tilde{V}$ ) between the ports and the device types reveals, that the ports are highly associated with the target variable (Source Port (sp): 0.71, Destination Port (dp): 0.62). The most probable explanation for this finding is that different devices use different ports more regularly than others. However, whether the use of a particular port results from the device itself, the operating system or the specific applications running at the time of the network traffic capturing, cannot be determined conclusively based on the dataset in use.

When comparing the two datasets "reduced activity" and "extensive activity", it can be observed, that for the "reduced activity"-dataset  $\tilde{V}$  is lower (cf. table 3.5) than for the "extensive activity"-dataset. As  $\tilde{V}$  controls for the sample size, the fact that the "reduced activity"-dataset contains less data, can not be taken as an explanation.

## Frequency Distribution

Looking at the frequency distributions, the distributions of different devices are indeed distinctive. Several aspects are observed:

- Source ports differ between the different devices, while the destination ports show less difference.
- Especially in the "extensive activity"-dataset (figure 4.2), the source ports between the classes "iPhone" and "iPad" differ substantially. The "reduced activity"-dataset does also seem to point into the same direction, but with far fewer data points to compare (figure 4.4).
- The destination ports in general show little or no differences between devices (figures 4.1 and 4.3)

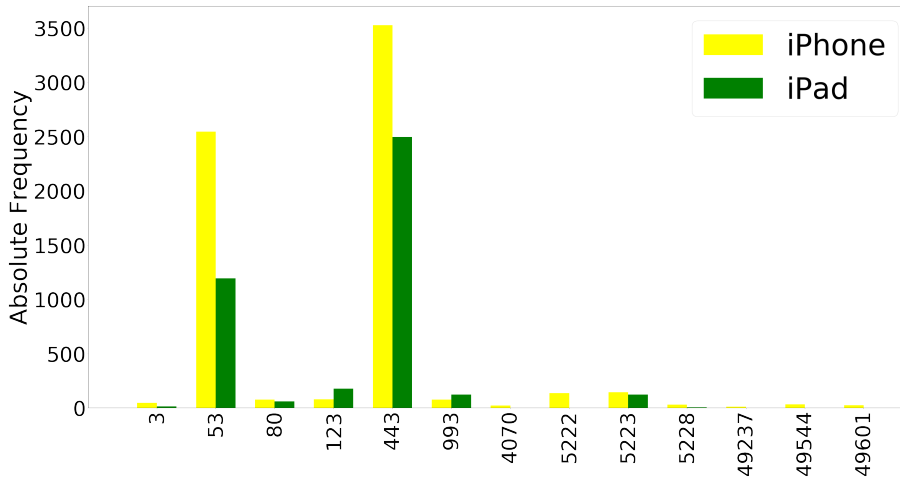


Figure 4.1: Destination Ports in "extensive activity"-dataset for iPhone and iPad (lower-bound 10)

To determine further, how performant the ports are for discriminating between devices, several machine learning algorithms were applied.

## Decision Trees

Three decision trees were calculated using three different datasets and only the two variables 'dp' (destination port) and 'sp' (source port). The same three datasets are used with the same naming throughout the whole explorative data analysis:

- The "extensive activity"-dataset
- The "extensive activity"-dataset, reduced to the two classes iPhone and iPad
- The "reduced activity"-dataset

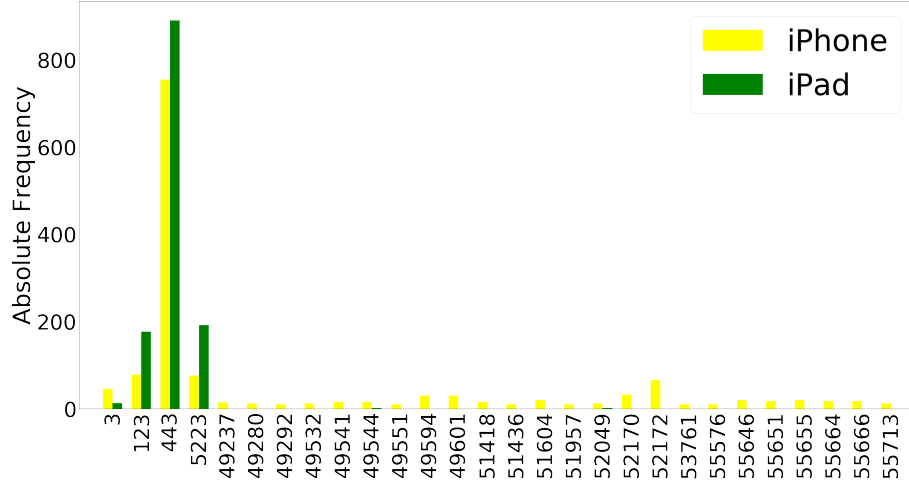


Figure 4.2: Source Ports in "extensive activity"-dataset for iPhone and iPad (lowerbound 10)

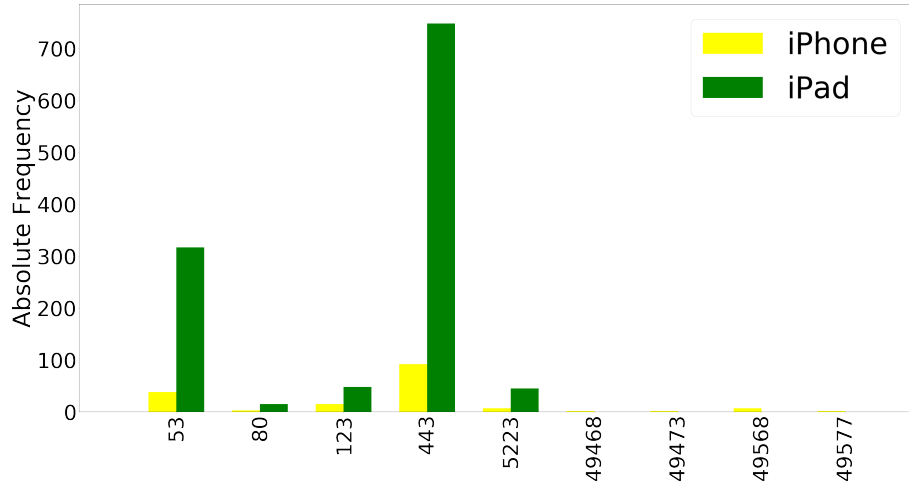


Figure 4.3: Destination Ports in "reduced activity"-dataset for iPhone and iPad (lower-bound 5)

Decision Tree a) yielded an *weighted  $F_1$  score*<sup>1</sup>( $\tilde{F}_1$ ) of 58% (figure 4.5), b) of 76% (figure 4.6) and c) of 92% (figure 4.7). The evaluation was done using cross-validation and the  $\tilde{F}_1$  scoring function. In all three decision trees, it can be observed, that the source port (sp) is more important for the DT, as it is used more often to execute a split in the data. This can be seen especially well in figure 4.7.

Applying more sophisticated decision tree algorithms (Gradient Boosted Trees) to the same three datasets, yielded an  $\tilde{F}_1$  of a) 78%, b) 86% and c) 94%.

The results for all decision trees show that with increasing data, the performance increases, while it decreases the more categories the target variable has. To check the performance of the classifier against overfitting, a Gradient Boosted Tree trained on the smallest dataset

<sup>1</sup>The weighted F1 score is the average of the F1 score of each class with weighting based on the each labels support [35]

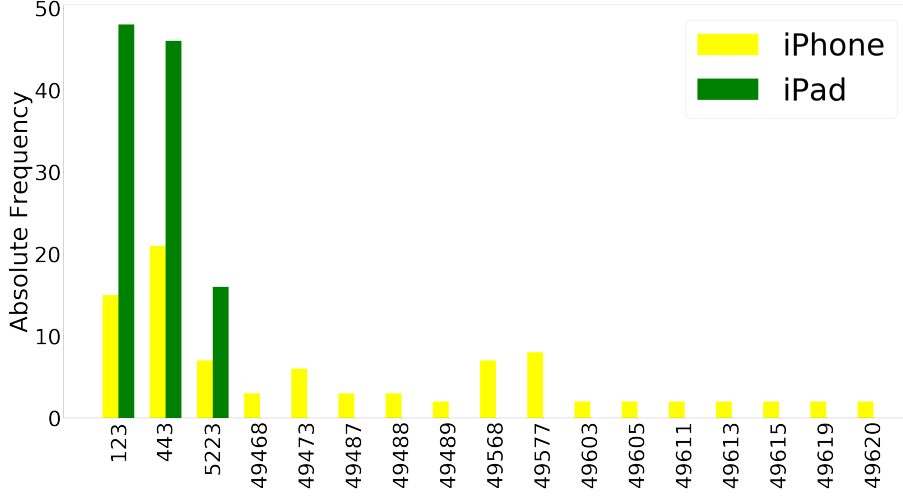


Figure 4.4: Source Ports in "extensive activity"-dataset for iPhone and iPad (lowerbound 5)

(dataset c), was cross-validated i) against the biggest dataset (dataset a) and ii) vice-versa. In this setting, the performances decreased drastically down to  $\tilde{F}_1$  scores of 54%(i) and 66%(ii)

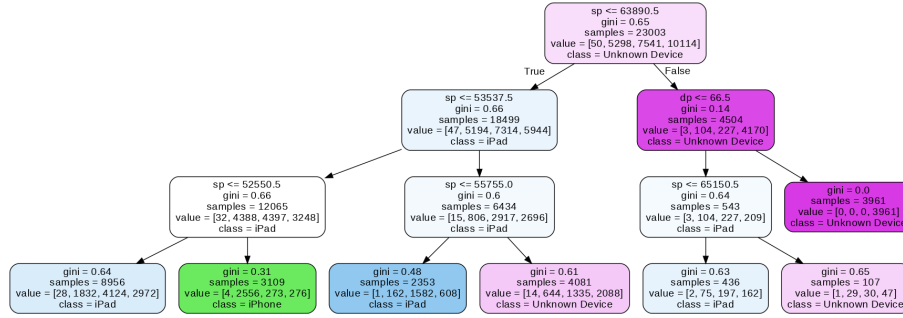


Figure 4.5: Decision Tree using "extensive activity"-dataset(a)

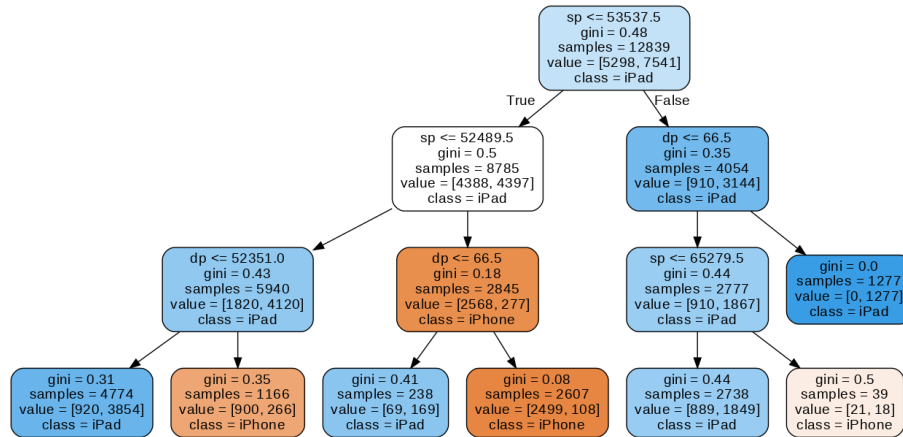


Figure 4.6: Decision Tree using "extensive activity"-dataset reduced to the classes "iPhone" and "iPad"(b)

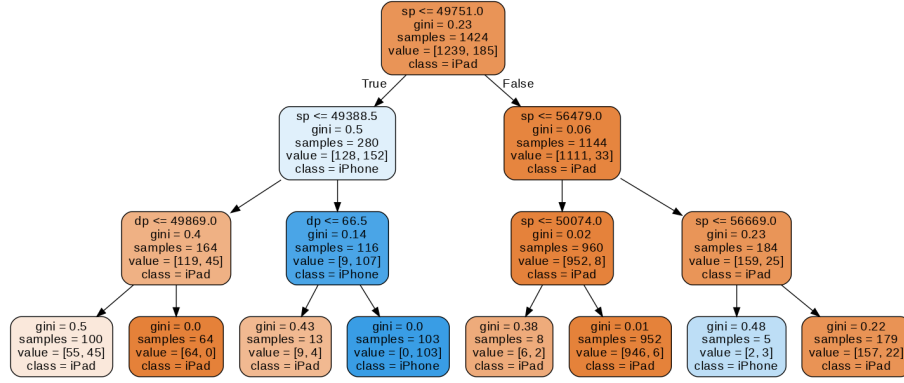


Figure 4.7: Decision Tree using "reduced activity"-dataset(c)

## Support Vector Machines

When applying more flexible classifiers like Support Vector Machines to the three datasets, the results yielded high  $\tilde{F}_1$  scores: a) 89%, b) 93% and c) 84%. In this case, the high results do not follow the same pattern as with the decision trees, as dataset a) has more classes and more data than dataset c), and still has a higher  $\tilde{F}_1$  score than the classifier c). However, when looking at the decision boundaries for all three classifiers, it becomes obvious that overfitting has taken place. It can be seen in figures 4.8 and 4.9 that the classifier distributed almost each point separately.<sup>2</sup> By looking at the amount of support vectors needed to support the classifiers decision boundaries, the overfitting can be seen well: a) 50 (class unknown device) - 2983 (class iPad) - 4843 (class iPhone)- 5362 (class mac) b) 2823 (class iPhone) - 3477 (class iPad) c) 772 (class iPhone) - 158 (class iPad).

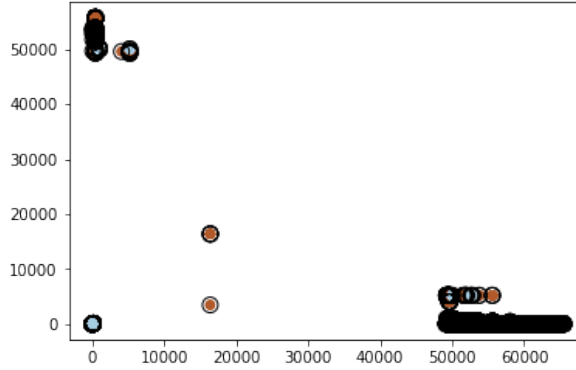


Figure 4.8: SVM Classifier on "extensive activity"-dataset(b) limited two classes "iPhone" and "iPad" using radial kernel

If forced to draw linear decision boundaries as in figures 4.10 and 4.11, the classifiers yield results with low  $\tilde{F}_1$  scores: Trained on the three datasets, the classifiers yielded  $\tilde{F}_1$  scores of a) 0.23%, b) 0.48% and c) 0.65%. The reason for these low accuracy scores becomes, however, clear when looking at the data in two-dimensional space: The data is linearly unseparable.

<sup>2</sup>As dataset a) contains more than two target classes, it can not be visualized in 2d. It is assumed, that a graphical representation would show the overfitting of the classifier as well.

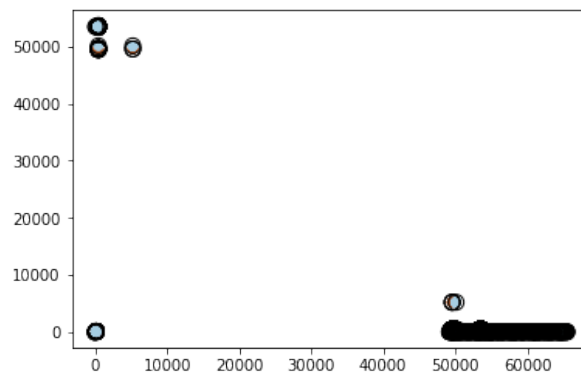


Figure 4.9: SVM Classifier on "reduced activity"-dataset(c) using radial kernel

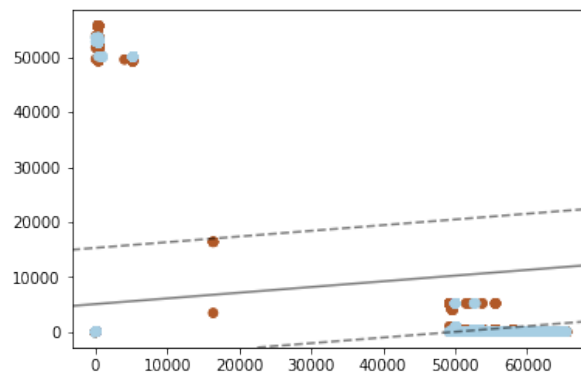


Figure 4.10: SVM Classifier on "reduced activity"-dataset(c) using linear kernel

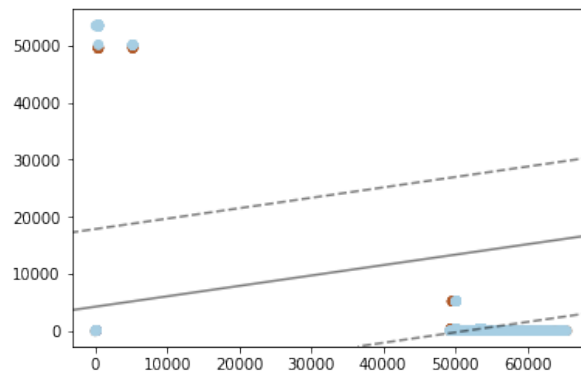


Figure 4.11: SVM Classifier on "extensive activity"-dataset(b) limited two classes 'iPhone' and 'ipad' using linear kernel

## Discussion

It can be observed, that i) the two features *source port* and *destination port* contain discriminating power towards the device type and that ii) they are not strong enough to accurately classify between different devices on their own. This conclusion is supported by both machine learning approaches, as well as the frequency distributions and the measures of association.

## 4.2 IP addresses

### Corrected Cramers V

When comparing the association of all features towards the target variable, the IP address and related features show high scores: While the IP address leads with a  $\tilde{V}$  of 0.66%, the other features follow with 50.6% (*handle\_name*, *block*), 0.44% (*cidr\_name*, *company\_name*) and 0.41% (*asn\_description*). The association between this group of features and the device type is therefore evident.

### Frequency Distribution

It can be seen, that the IP addresses vary among the different devices. And while they vary only little between similar devices like iPads and iPhones (figure 4.12), they vary much more between different device types such as *e.g.* Macs and iPhones (figure 4.13).

As mentioned in section 4.1, the reasons behind the occurrence of differences in the frequency patterns of different devices are not entirely clear. The same applies also to the patterns observed in the context of the IP address related features. Nevertheless, it seems that the use of different IP addresses allows some sort of discrimination between devices.

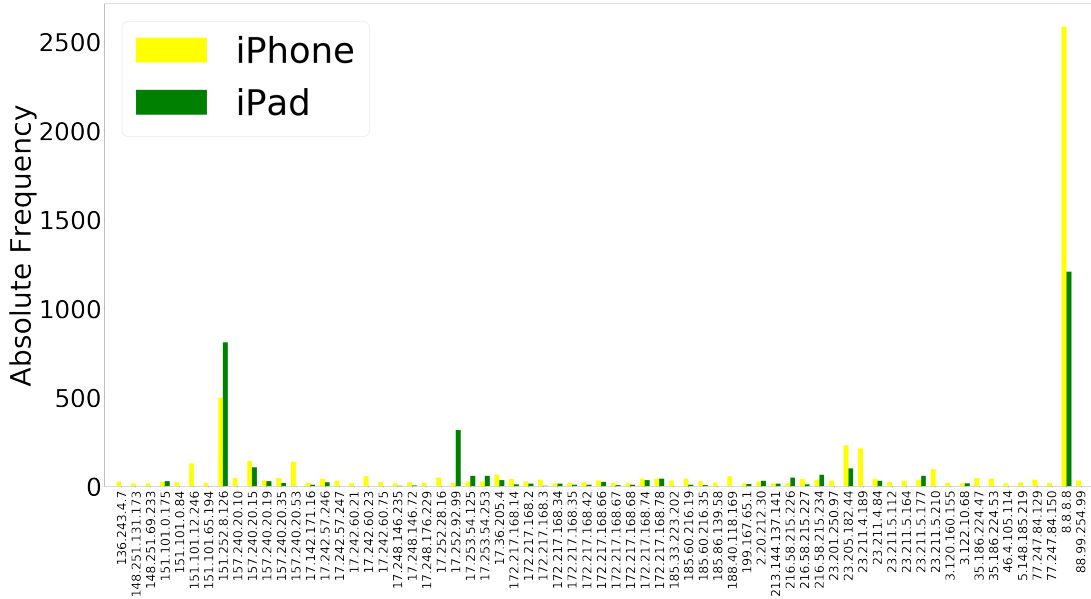


Figure 4.12: Frequency Distribution of IP address calls between iPhone and iPad in "extensive activity"-dataset(a) (lowerbound 15)

Looking at other features such as IP block or the company name, similar patterns in the frequency distribution can be observed, as with the IP address. By looking *e.g.* the IP block, it can be seen, that the frequency patterns of the used IP address blocks vary among all three device types iPhone and iPad (figure 4.14), iPhone and Mac (figure 4.15) and iPad and Mac (figure 5.1 in appendix). Other features such as company name, show

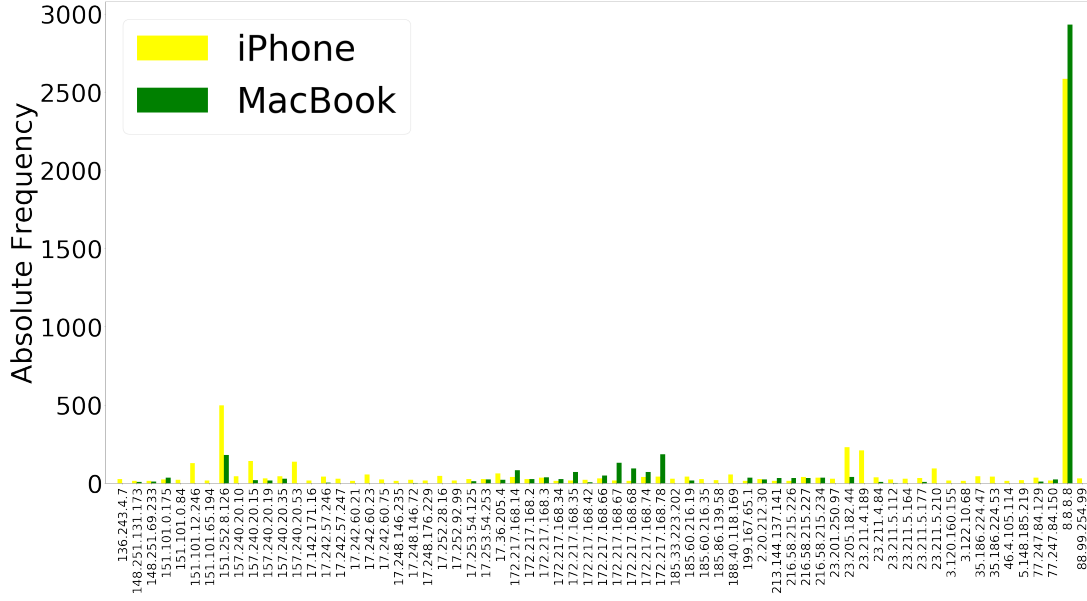


Figure 4.13: Frequency Distribution of IP address calls between iPhone and Mac in "extensive activity"-dataset(a) (lowerbound 15)

similar patterns (cf. tables 5.2, 5.3 and 5.4). In the case of the company name the pattern differences are not as strong as with the IP block.

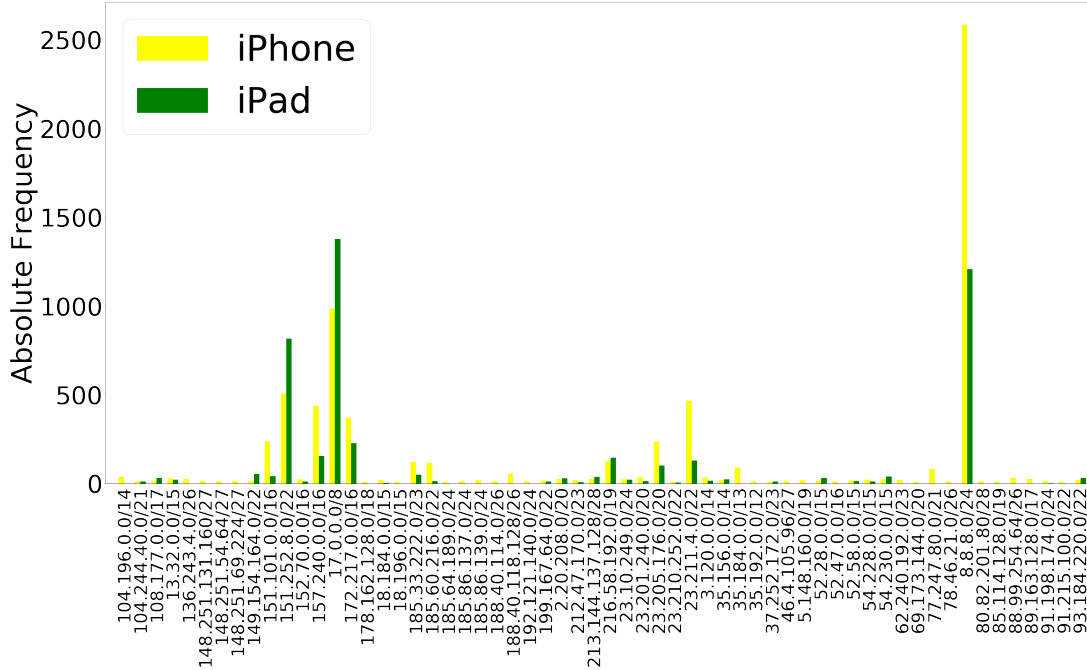


Figure 4.14: Frequency Distributions between iPhone and iPad using the IP block in "extensive activity"-dataset(a) (lowerbound 10)

Because these results, were obtained analyzing the "extensive activity"-dataset, it is possible, that the observed differences in the frequency pattern exist only within this dataset. To test this hypothesis, cross-validation was performed between the two datasets "exten-

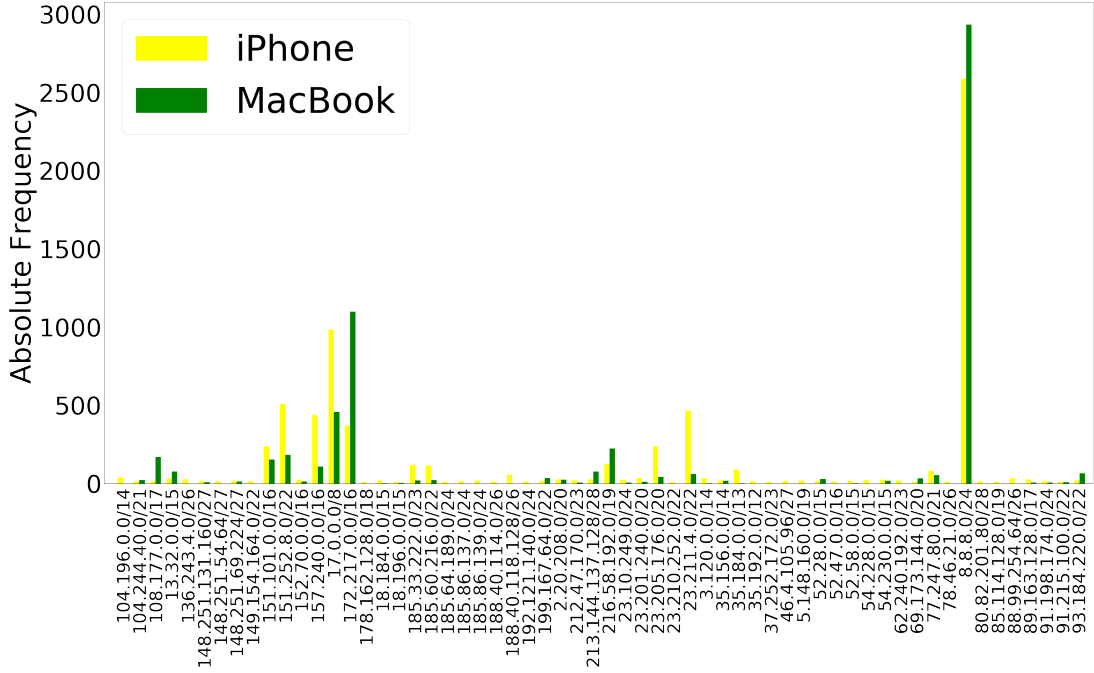


Figure 4.15: Frequency Distributions between iPhone and Mac using the IP block in "extensive activity"-dataset(a) (lowerbound 10)

sive activity" (a) and "reduced activity" (c), by comparing the same devices (iPad and iPhone) from different datasets against each other: iPhone from dataset a) with iPhone from dataset b), etc. Thus, if the results observed before are real, device-specific frequency patterns in IP address-related features could be assumed and, therefore, no differences should be seen between the same device even between different datasets.

The results showed, that for the same device, similar patterns in the frequency of IP blocks were obtained over multiple datasets. This can be seen in well table 4.16. However, as can be seen in figure 4.17, the comparison of frequencies is not possible concludingly, as the "reduced activity"-dataset contains far fewer datapoints and the patterns do not show as well as in the "extensive activity"-dataset. Similar results as with the IP block, can be found when the company name feature is compared (cf. tables 5.5 and 5.6 in appendix).

These findings support the claim, that IP address-related attributes are discriminating features, that allow distinguishing between different devices. It can be shown, that this result holds to some degree for the comparison of different devices and over different types of datasets. These findings are supported more strongly by the IP address and less strong by the block and the company name. It is assumed, that similar results could be obtained when using the other IP address-related features.

To determine, which of the IP address related features are holding the most discriminating power in classifying different devices, machine learning algorithms (decision trees and support vector machines) were trained on all of them at once ('asn\_description', 'ip', 'block', 'cidr\_name', 'handle\_name', 'company\_name'). The same datasets a), b) and c) as before were used.

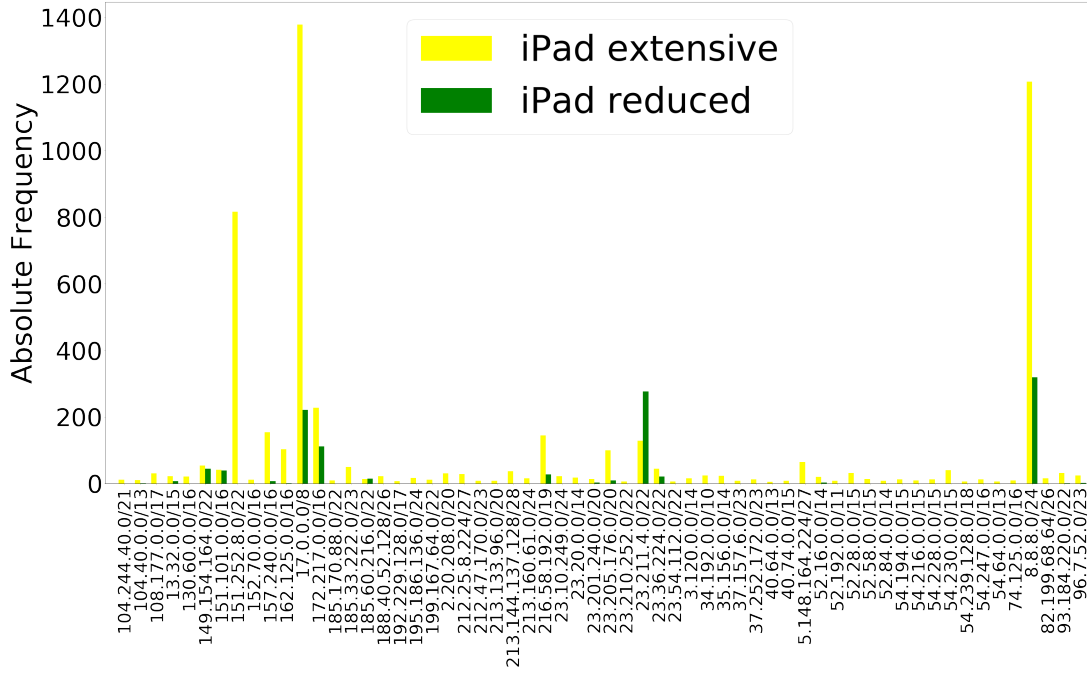


Figure 4.16: Cross Validation of iPad Ip Blocks from both datasets "extensive activity" and "reduced activity" (lowerbound 5)

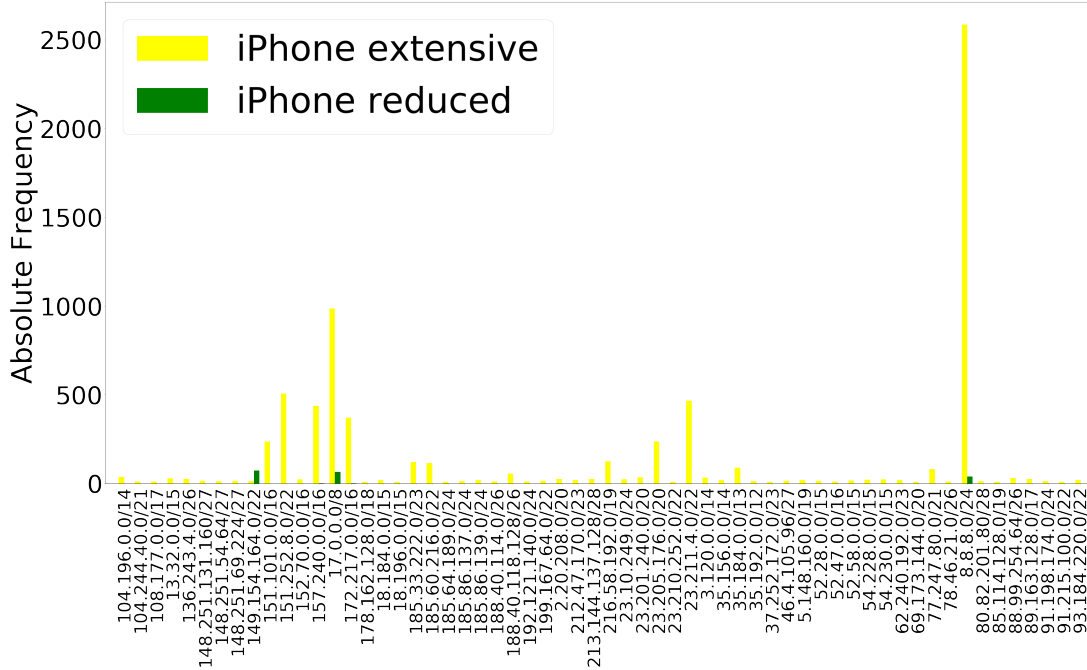


Figure 4.17: Cross Validation of iPhone Ip Blocks from both datasets "extensive activity" and "reduced activity" (lowerbound 12)

## Decision Trees

Gradient Boosted Trees yielded an  $\tilde{F}_1$  score of a) 59%, b) 69% and c) 87% for the datasets. Overfitting patterns can again be observed, but not as strong with the ports (cf. section

4.1). Using the classifier trained on dataset c), with the data b), to predict the device types, yielded a  $\tilde{F}_1$  score of 47% of balanced accuracy. The other way, 38% accuracy was obtained. This result shows, that the classifier is, indeed, overfitted and the classifiers perform maximally at chance.

By looking at the tree trained on dataset b) in figure 4.18, it can be seen that all IP address-related features were used to generate the tree.

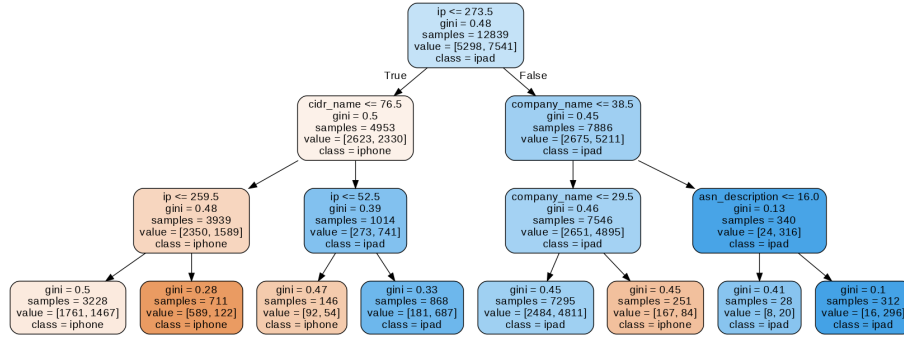


Figure 4.18: Decision Tree on "extensive activity" dataset (max depth 3)

## Support Vector Machines

Support Vector Machines using a radial kernel, yielded  $\tilde{F}_1$  scores of a) 62%, b) 70% and c) 87%. Similar to the decision trees, the performance increased with decreasing data and values of the target variable. Applying the same cross-validation used with the decision trees, yielded a result of 50% in both ways. As the radial kernel is known to overfit (cf. section 4.1), overfitting seems to be the problem here as well.

When forcing the SVMs to work with a linear decision boundary,  $\tilde{F}_1$  scores of a) 28%, b) 32% and c) 66% were obtained. However, these classifiers tend to overfit as well, as cross-validation again revealed low accuracy scores, even for classifier c): 0.46% and 49%.

## Discussion

The results show, that similar to the port features, the IP address related features contain discriminating information, that can distinguish between different devices to some degree. A classifier trained solely based on IP address-related features, however, does not perform well.

## IP address as source for information extraction

Apart from their effectiveness as features in a machine learning setting, the IP address related features contain an enormous amount of additional information not inherent to the original dataset.

By looking *e.g.* at the company name, a lot is revealed about the device, the application or the background process, that generated the traffic. Looking at the data in table 4.1 one can derive the device type (Apple device) and some user activity (facebook).

| Company Name                       | Frequency Call |
|------------------------------------|----------------|
| Google LLC                         | 3248           |
| Apple Inc.                         | 985            |
| Akamai International, BV           | 792            |
| RIPE Maintainer                    | 506            |
| Facebook, Inc.                     | 436            |
| Fastly                             | 237            |
| Hetzner Online GmbH - Contact Role | 222            |
| Abuse-C Role                       | 214            |
| A100 ROW GmbH                      | 130            |
| RIPE DBM                           | 130            |

Table 4.1: Top 10 company names from DNS information from "extensive activity"-dataset for device "iPhone"

### 4.3 Temporal features

The third group of features is generated using time as a property to distinguish between different devices. Based on physical network interfaces and based on the priority with whom servers treat different devices, one can differentiate them. A positive aspect of this approach is, that the generated features are again an aggregation of multiple data points (multiple flows), that make it independent of the single flows. Using this approach helps to eliminate the bias introduced by the different applications installed on a device.

In the dataset are two time-related features available: The **round-trip time** estimate (*rtt*) and the flow duration (*dur*). While the *rtt* describes the time that passes from the request to the server to any kind of response from the server, the *dur* describes the actual amount of milliseconds that the flow was established for. While the *rtt* is packet and content-size independent and does rely only on how fast the server responds to a specific request, the *dur* is packet and content-size dependent and needs to be normalized. Using the information about the total packets and total octets sent and received, the ratio *duration-per-packet* (*dpp*) and *duration-per-octet* (*dpo*) was calculated.

By limiting the dataset to UDP-flows and by aggregating the newly calculated features *dpp* and *dpo* for each device type, results as in table 4.2 were obtained. It can be seen that for both *dpo* and *dpp* the iPad has the shortest duration of all devices, while the Mac has the longest. This correlates with the fact that the Mac Air used in this case is from 2013, while the iPad is from 2018. This finding shows that the duration and time-related features allow to not only distinguish between different devices but also might help to classify network traffic from a yet unknown device into a class where the average speed of transmission is known.

| Devices/Features | dpo in ms | dpp in ms |
|------------------|-----------|-----------|
| Unknown device   | 0.001016  | 0.180833  |
| Mac Air          | 0.004690  | 1.067764  |
| iPad             | 0.000166  | 0.033280  |
| iPhone           | 0.000812  | 0.145315  |

Table 4.2: Duration per Packet (dpp) and Duration per octet (dpo) on the "extensive activity"-dataset

Similar results were obtained by using the rtt with TCP and ICMP protocols from the dataset. By just aggregating the rtt's for each device without any normalization, the results were obtained according to table 4.3.

| Devices        | rtt in ms |
|----------------|-----------|
| Unknown Device | 0.169500  |
| Mac Air        | 0.356845  |
| iPad           | 0.064378  |
| iPhone         | 0.196798  |

Table 4.3: Average round trip time (rtt) on "extensive activity"-dataset

Similar to the UDP flow aggregated measures (dpp and dpo), the TCP flows show a hierarchy in speed, that correlates with the physical network interfaces of the used devices.



# Chapter 5

## Discussion

In the following section, the problems found and the observations made during the data generation (chapter 3) and exploration (chapter 4) are discussed concludingly and put into perspective to the design of a network traffic classification system, discussed in the next chapter.

### 5.1 Dataset

As already discussed in chapter 3, the generation of network traffic data is not straightforward and subject to various issues such as:

- bias towards unwanted categories (noise)
- unbalanced categories
- non-exhaustiveness of dataset regarding devices and applications
- labeling
- representative and scalable generation

### Generation and set-up

By using physical or simulated devices and by tunneling and capturing network traffic through a VPN set-up, all the above-stated problems can, theoretically, be omitted when labeling different devices is the goal. The problems, however, still exist towards the classification of applications.

The approach to generate "reduced activity"-datasets to reduce the application-specific data and yield more device-specific data has proven to be weak. The explorative analysis showed that the data still is biased and certain applications also run in the background

and generate application rather than device-specific network traffic. Instead of trying to reduce application-based traffic generation further, it is suggested that all network traffic is used for inspection and training of a classifier for the following reasons:

1. It is difficult to limit the device's traffic to only device-specific data.
2. In daily use, most network traffic is produced by applications and not by the devices' operating system itself. For some device types, it is not possible to remove all of its applications (*e.g.* Apple Devices).
3. By reducing the devices' network traffic generation, hardly any packets get generated and it becomes difficult to generate sufficient amounts of data.
4. The "reduced activity"-dataset did not prove to be more powerful for classification and it can be assumed, that its features did not contain more discriminating power than the "extensive activity"-dataset.

For the generation of data it is, therefore, suggested, to create network traffic through the use of a VPN set-up and normal interaction between the user and the device.

An additional drawback in the experimental set-up for the generation of the dataset is the fact, that only one physical device was used for each device type (One iPhone was used for all the iPhone traffic). Therefore, the generalization of the results, as well as the power of the features towards discriminating the target variable, is reduced. By using more physical (or simulated) devices of the same type, a more abstract representation of a device type could have been represented in the dataset.

## Flow- vs Packet-based Dataset

Even though it is arguable, that especially in combination with more sophisticated temporal features, a packet-based dataset could yield good results, the flow-based dataset has proven to be a good baseline for two reasons:

1. Flows already represent an aggregated and more stable representation of network traffic data than single packets.
2. Flows can be categorized and filtered. They are therefore more suitable for certain aggregated features such as temporal features (dpo and dpp).

## 5.2 Feature Extraction and Selection

The explorative data analysis showed that various features exist, that contain discriminating powers. They can be grouped into three categories: The ports, IP address-related features, and temporal features.

Looking at the frequency distributions of these groups independently from each other, differences between different devices could be observed. While more similar devices such as iPhones and iPads showed similar patterns, the comparison of a MacBook and an iPhone revealed bigger differences. Due to the restricted experimental set-up used to generate the data, it was not possible to compare two completely different devices (*e.g.* iPhone and PC using Microsoft Windows), because there were no such devices at hand.

While the frequency distributions give some kind of hint, that there are differences between the devices regarding specific ports used, IP addresses called and flow durations, the frequency patterns can not be used directly for the classification of any kind. The major reason for this conclusion is, that the data was not sufficient to identify clear patterns. All frequency-based observations were made by the human eye and the differences observed could not be quantified. On top of that, it remains unclear where the differences in the frequency patterns come from. It is likely, that they result mainly from different applications installed on the different devices rather than from device types.

Similar to the frequency distributions, also the applied machine learning algorithms supported the discriminating power of the three feature groups. Using each group of features independently and training different classifiers (decision trees and support vector machines) yielded classifiers with medium performance and accuracy. The classifiers tended to overfit due to the rather small datasets. Controlling the classifiers for overfitting affected the classifiers accuracies quite similar: All of them went down to chance performance.

In addition to the strength of the discussed features for classification, the IP address contained a big source of external information when combined with the information obtained from DNS servers. Especially interesting is the fact, that the names of the companies, that registered a specific IP address, can be extracted. In many cases, these names help identify accurately a specific device or application type.

## 5.3 Application- vs. Device-specific Traffic

A general problem in all network traffic based classification of devices and applications, especially in the VPN set-up used in this work, is the fact, that it is not possible to know, where the discriminating information is coming from: The device or the applications on the device.

The only possibility to overcome this issue systematically would be to wipe the devices of all applications. This, however, would be connected to an even greater effort needed to create the dataset. In some cases, this would not even be possible (*e.g.* Apple devices). In the current set up, that relies mostly on physical devices, such an attempt is almost infeasible. Especially if more devices should be incorporated into the dataset.

This observation has a major implication: It is not possible to create a generalization of a specific device type, as the specific configuration of the device used to create the network traffic cannot be isolated from the device itself. The costs to circumvent this issue, by creating clean devices or by overlapping many different datasets from the same device types are too high. This is additionally problematic due to the fact, that all device

type-specific network traffic, was generated using only one physical device (cf. section 5.1).

However, even if it is unlikely to classify device types as such, it is possible to classify specific devices. Based on a dataset, where the devices are all known, it is possible to identify a new or an already seen device, based on the patterns of its features.

## 5.4 Conclusion

Based on the results, obtained with datasets generated through a VPN-based set-up, calculated with explorative data analysis and machine learning methods, and concerning the explorative goals mentioned in section 1.2, the following conclusions can be drawn:

- The generation of a labeled, unbiased dataset is difficult and not feasible without an enormous amount of time and work. This difficulty holds for the generation, the maintainability and the extendability of a network traffic dataset.
- The generation of application-specific and/or device type-specific network traffic is extremely difficult, as the user has no absolute control over the device itself. However, the labeling of a specific device can be achieved with little effort, when using a set-up, where the network traffic of that specific device can be captured and labeled according to said device (*e.g.* VPN set-up).
- The distinction of different specific devices is possible, while the classification of abstract device types is infeasible, due to the problems stated above.
- Network traffic flows contain many discriminating features, that allow to assign them to the source they originated from (device). These features were generated on flow- as well as on aggregated-flow(device) level. The features with the highest discriminating power are the ports, the IP address related features and the temporal features.
- The DNS information contains useful information, that helps in some cases to classify devices and applications even with little data.

The main focus of this work was to explore the possibilities and limitations of network traffic data as a basis for the identification of device types and applications. It showed that the identification of specific devices is possible, using patterns of certain features and external DNS information. The question, however, whether abstract device types can be distinguished from each other remains unanswered at this point.

This work also showed, that network traffic classification of devices is not as simple as it sounds: None of the classifiers were able to accurately distinguish between the different devices without overfitting. And even though it is thinkable, that more sophisticated machine learning approaches, combined with hyperparameter tuning would yield stronger, more performant classifiers and that bigger and cleaner datasets would reveal different

results, patterns and conclusions regarding the classification of devices and applications, it is also possible, that network traffic classification simply is difficult. Especially, when the data is not artificially generated or labeled using experts but instead gathered in a natural real-life scenario, where the amount of available data at hand is not large, the data is unclean and the categories are unbalanced.

Based on these conclusions, the next section focuses on describing a system design that makes use of the knowledge gathered in the preceding chapters.

## 5.5 Future Work

Because of the difficult data generation process, the difficulty to generate isolated application data and the limited discriminative powers of the features, a classical supervised machine learning approach has to be rejected for a practical application. Based on this conclusion, a network traffic classification system needs to allow a) the extraction of external information of a flow and b) the recognition of device-specific patterns.

### System Components

Minimally two components are needed to account for both goals:

- (A) **Extraction of Information:** By looking up external information on IP addresses many additional insights can be found about specific applications, operating systems and background processes (cookies, tracking services). This allows telling, what applications are used more frequently than others and what background processes could be running even without the user knowing. Even though this information is not directly usable for classification, it can be evaluated by an information extraction system and later be used for pattern detection (cf. B).
- (B) **Recognition of device-specific patterns:** Based on features such as *e.g.* ports, IP addresses, temporal information, and others it is possible to distinguish two devices. Even though two abstract device types may not be identified as such (*e.g.* recognize a device as iPhone Xs), two specific devices used by two different people may, still, be distinctive. Based on a minimal set of devices known beforehand, incoming traffic can be analyzed and categorized into the already known categories. The result is then a growing set of known devices, that can be used to detect activities by known or by new yet unknown devices. Using the additional information of point A, changes of the same device over time can be captured as well: Applications that get installed or uninstalled can be detected and registered.

This, however, requires data, where the origin of a packet can be determined: It needs to be known, what flows belongs to what device. This requirement can be fulfilled, if *e.g.* a local IP address is known.

## Architecture

Based on the previous conclusions and requirements, a semi-supervised pattern recognition system could be a way to use network traffic data best. A similar approach has been chosen by Tongaonkar et al. [40]. In their paper, the authors propose a system for network traffic classification, that overcomes traditional systems shortcomings by learning the application signatures right on the network, where the traffic is deployed and thus eliminating manual intervention. The system combines automated signature generation algorithms with real-time traffic classifiers and can automatically learn application signatures tailored to a specific network [40].

Grolman et al. use a similar approach called Co-Training, a semi-supervised algorithm that can be used for labeling unlabeled examples using a small set of labeled examples [20].

An approach similar to the one proposed by Tongaonkar et al. or Grolman et al. seems to be promising for the problem at hand. By gathering only limited initial ground truth about relevant devices and their network traffic flows, flows from unknown devices can be detected as either a) already seen, based on their feature-based signature (incl. DNS information) and the patterns can be updated with the new information or b) as yet unknown devices, whose signatures need to be calculated and stored for the recognition of new devices in the future. The major advantage of such a system is, that it needs relatively little data for classification. Because the signature can be updated every time a new device can be classified, the system can make more accurate decisions the longer it runs.

# Bibliography

- [1] Wireshark wiki - libpcap file format. <https://wiki.wireshark.org/Development/LibpcapFileFormat>. Accessed: 2019-04-15.
- [2] Jason A. Donenfeld. WireGuard: fast, modern, secure VPN tunnel. <https://www.wireguard.com/>. Accessed: 2019-05-12.
- [3] Khaled Al-Naami, Swarup Chandra, Ahmad Mustafa, Latifur Khan, Zhiqiang Lin, Kevin Hamlen, and Bhavani Thuraisingham. Adaptive encrypted traffic fingerprinting with bi-directional dependence. pages 177–188, 12 2016. doi: 10.1145/2991079.2991123.
- [4] Hasan Faik Alan and Jasleen Kaur. Can Android Applications Be Identified Using Only TCP/IP Headers of Their Launch Time Traffic? pages 61–66, 2016. doi: 10.1145/2939918.2939929.
- [5] Alessandro Balzano. How to write generic dissectors in Wireshark Diplomats can code too! <http://wintermade.it/blog/posts/how-to-write-generic-dissectors-in-wireshark.html>. Accessed: 2019-07-17.
- [6] Frederic Amblard, Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. On the temporal analysis of scientific network evolution. *Proceedings of the 2011 International Conference on Computational Aspects of Social Networks, CASoN’11*, pages 169–174, 2011. doi: 10.1109/CASON.2011.6085938.
- [7] Lenz Baumann. IP Evaluation.ipynb. <https://colab.research.google.com/drive/1gLccvi8cRhtFFrW3RGD7eawPlMYFqe8->. Accessed: 2019-07-17.
- [8] Dmitri Bekerman, Bracha Shapira, Lior Rokach, and Ariel Bar. Unknown malware detection using network traffic classification. *2015 IEEE Conference on Communications and Network Security, CNS 2015*, pages 134–142, 2015. doi: 10.1109/CNS.2015.7346821.
- [9] Wicher Bergsma. A bias-correction for Cramér’s V and Tschuprow’s T. *Journal of the Korean Statistical Society*, 42(3):323–328, sep 2013. ISSN 1226-3192. doi: 10.1016/J.JKSS.2012.10.002. URL <https://www.sciencedirect.com/science/article/pii/S1226319212001032>.
- [10] Enrico Bocchi, Luigi Grimaudo, Marco Mellia, Elena Baralis, Sabyasachi Saha, Stanislav Miskovic, Gaspar Modelo-Howard, and Sung Ju Lee. MAGMA network

- behavior classifier for malware traffic. *Computer Networks*, 109:142–156, 2016. ISSN 13891286. doi: 10.1016/j.comnet.2016.03.021. URL <http://dx.doi.org/10.1016/j.comnet.2016.03.021>.
- [11] Carnegie Mellon University. Super Mediator. [https://tools.netsa.cert.org/super\\_mediator/index.html](https://tools.netsa.cert.org/super_mediator/index.html). Accessed: 2019-06-04.
- [12] Carnegie Mellon University. Yet Another Flowmeter. <https://tools.netsa.cert.org/yaf/index.html>, 2006. Accessed: 2019-06-04.
- [13] Zhitang Chen, Ke He, Jian Li, and Yanhui Geng. Seq2Img: A sequence-to-image based approach towards IP traffic classification using convolutional neural networks. *Proceedings - 2017 IEEE International Conference on Big Data, Big Data 2017*, 2018-January:1271–1276, 2018. doi: 10.1109/BigData.2017.8258054.
- [14] David A Cieslak, Nitesh V Chawla, and Aaron Striegel. Combating Imbalance in Network Intrusion Datasets. *IEEE*, 2006. URL <https://www3.nd.edu/~dial/publications/cieslak2006combating.pdf>.
- [15] Daniel Cunningham, Sean F. Everton, and Philip J. Murphyl. Casting more light on dark networks: A stochastic actor-oriented longitudinal analysis of the noordin top terrorist network. *Illuminating Dark Networks*, pages 171–185, 2013. doi: 10.1017/CBO9781316212639.012.
- [16] Sean F Everton. Disrupting Dark Networks With Sna. <https://www4.uwm.edu/people/haas/sna/evertonsnaguide.pdf>. Accessed: 2019-07-17.
- [17] Sean F. Everton and Dan Cunningham. Detecting significant changes in dark networks. *Behavioral Sciences of Terrorism and Political Aggression*, 5(2):94–114, 2013. ISSN 19434480. doi: 10.1080/19434472.2012.725225.
- [18] Luis Martin Garcia. Programming with libpcap-sniffing the network from our own application. *Hakin9-Computer Security Magazine*, 2:2008, 2008.
- [19] Google. Welcome To Colaboratory - Colaboratory. <https://colab.research.google.com/notebooks/welcome.ipynb>. Accessed: 2019-07-15.
- [20] Edita Grolman, Andrey Finkelshtein, Rami Puzis, Asaf Shabtai, Gershon Celniker, Ziv Katzir, and Liron Rosenfeld. Transfer learning for user action identification in mobile apps via encrypted traffic analysis. *IEEE Intelligent Systems*, 33(2):40–53, 2018. ISSN 15411672. doi: 10.1109/MIS.2018.111145120.
- [21] Philip Hane. ipwhois. <https://github.com/secynic/ipwhois>. Accessed: 2019-07-06.
- [22] Jonas Hoechst, Lars Baumgartner, Matthias Hollick, and Bernd Freisleben. Un-supervised Traffic Flow Classification Using a Neural Autoencoder. *Proceedings - Conference on Local Computer Networks, LCN*, 2017-Octob:523–526, 2017. doi: 10.1109/LCN.2017.57.

- [23] INCIBE-CERT. Understanding industrial network traffic, dissectors and Lua and Kaitai. <https://www.incibe-cert.es/en/blog/understanding-industrial-network-traffic-dissectors-and-lua-and-kaitai>. Accessed: 2019-07-17.
- [24] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.
- [25] Renaud Lambiotte, Martin Rosvall, and Ingo Scholtes. Understanding Complex Systems: From Networks to Optimal Higher-Order Models, 2018. URL <http://arxiv.org/abs/1806.05977>.
- [26] Wei Li and Andrew W Moore. A machine learning approach for efficient traffic classification. In *2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 310–317. IEEE, 2007.
- [27] Manuel Lopez-martin, Senior Member, and Belen Carro. Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things. *IEEE Access*, 5:18042–18050, 2017. doi: 10.1109/ACCESS.2017.2747560.
- [28] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammadsadegh Saberian. Deep packet: a novel approach for encrypted traffic classification using deep learning. *Soft Computing*, 2019. ISSN 14337479. doi: 10.1007/s00500-019-04030-2.
- [29] Andrew G. Mason. *Cisco Secure Virtual Private Network*. Cisco Press, 2002.
- [30] Pandas. Python Data Analysis Library. <https://pandas.pydata.org/>.
- [31] Python. Python.org. <https://www.python.org/about/>. Accessed: 2019-07-15.
- [32] Jarno Rajahalme, Alex Conta, Steve Deering, and Brian E. Carpenter. IPv6 Flow Label Specification. <https://tools.ietf.org/html/rfc3697>, 2004.
- [33] Shahbaz Rezaei and Xin Liu. How to Achieve High Classification Accuracy with Just a Few Labels: A Semi-supervised Approach Using Sampled Packets. *arXiv preprint arXiv:1812.09761*, 2018.
- [34] Scikit-learn. scikit-learn: machine learning in Python. <https://scikit-learn.org/stable/>. Accessed: 2019-07-15.
- [35] Sklearn.  $f_1$  weighted. [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1\\_score.html#sklearn.metrics.f1\\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score). Accessed: 2019-07-09.
- [36] Salvatore Stolfo, German Creamer, and Shlomo Hershkop. A temporal based forensic analysis of electronic communication. pages 23–24, 01 2006. doi: 10.2139/ssrn.2028734.
- [37] Shan Suthaharan. Big Data Classification: Problems and Challenges in Network Intrusion Prediction with Machine Learning. *Performance Evaluation Review*, 41(4): 70–73, 2014.

- [38] Vincent F. Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. Robust Smartphone App Identification via Encrypted Network Traffic Analysis. *IEEE Transactions on Information Forensics and Security*, 13(1):63–78, 2018. ISSN 15566013. doi: 10.1109/TIFS.2017.2737970.
- [39] The Raspberry Pi Foundation. What is a Raspberry Pi? <https://www.raspberrypi.org/help/what-is-a-raspberry-pi/>. Accessed: 2019-07-18.
- [40] Alok Tongaonkar, Ruben Torres, Marios Iliofotou, Ram Keralapura, and Antonio Nucci. Towards self adaptive network traffic classification. *Computer Communications*, 56:35–46, 2015. ISSN 1403664. doi: 10.1016/j.comcom.2014.03.026.
- [41] Tor Project. TOR. <https://www.torproject.org/>. Accessed: 2019-07-17.
- [42] Kashi Venkatesh Vishwanath and Amin Vahdat. Realistic and Responsive Network Traffic Generation. In *SIGCOMM*, Pisa, Italy, 2006.
- [43] Ly Vu, Cong Thanh Bui, and Quang Uy Nguyen. A Deep Learning Based Method for Handling Imbalanced Problem in Network Traffic Classification. pages 333–339, 2017. doi: 10.1145/3155133.3155175.
- [44] Ly Vu, Hoang V. Thuy, Uy Nguyen, Tran N. Ngoc, Diep Nguyen, Hoang Dinh Thai, and Eryk Dutkiewicz. Time series analysis for encrypted traffic classification: A deep learning approach. pages 121–126, 09 2018. doi: 10.1109/ISCIT.2018.8587975.
- [45] Wei Wang, Ming Zhu, Jinlin Wang, Xuwen Zeng, and Zhongzhen Yang. End-To-end encrypted traffic classification with one-dimensional convolution neural networks. *2017 IEEE International Conference on Intelligence and Security Informatics: Security and Big Data, ISI 2017*, pages 43–48, 2017. doi: 10.1109/ISI.2017.8004872.
- [46] Wikipedia. Cramers V. [https://de.wikipedia.org/wiki/Kontingenzkoeffizient#Cramers\\_V](https://de.wikipedia.org/wiki/Kontingenzkoeffizient#Cramers_V), . Accessed: 2019-07-04.
- [47] Wikipedia. IPFIX. [https://en.wikipedia.org/wiki/IP\\_Flow\\_Information\\_Export](https://en.wikipedia.org/wiki/IP_Flow_Information_Export), . Accessed: 2019-07-06.
- [48] Wikipedia. Virtual private network. [https://en.wikipedia.org/wiki/Virtual\\_private\\_network](https://en.wikipedia.org/wiki/Virtual_private_network), .
- [49] Jun Zhang, Xiao Chen, Student Member, Yang Xiang, Senior Member, Wanlei Zhou, and Jie Wu. Robust Network Traffic Classification. *IEEE/ACM TRANSACTIONS ON NETWORKING*, 23(4):1257–1270, 2015. ISSN 10636692. doi: 10.1109/TNET.2014.2320577.

# List of Figures

|      |                                                                                                                                    |    |
|------|------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.1  | A tree showing the classification of network traffic into iPad and iPhone generated traffic. . . . .                               | 6  |
| 2.2  | Support Vector Machine using a linear decision function (solid line). The margin is represented with the two dotted lines. . . . . | 7  |
| 3.1  | Network traffic capturing set-up using VPN and local network . . . . .                                                             | 12 |
| 3.2  | foo bar . . . . .                                                                                                                  | 16 |
| 4.1  | Destination Ports in "extensive activity"-dataset for iPhone and iPad (lower-bound 10) . . . . .                                   | 20 |
| 4.2  | Source Ports in "extensive activity"-dataset for iPhone and iPad (lower-bound 10) . . . . .                                        | 21 |
| 4.3  | Destination Ports in "reduced activity"-dataset for iPhone and iPad (lower-bound 5) . . . . .                                      | 21 |
| 4.4  | Source Ports in "extensive activity"-dataset for iPhone and iPad (lower-bound 5) . . . . .                                         | 22 |
| 4.5  | Decision Tree using "extensive activity"-dataset(a) . . . . .                                                                      | 22 |
| 4.6  | Decision Tree using "extensive activity"-dataset reduced to the classes "iPhone" and "iPad"(b) . . . . .                           | 22 |
| 4.7  | Decision Tree using "reduced activity"-dataset(c) . . . . .                                                                        | 23 |
| 4.8  | SVM Classifier on "extensive activity"-dataset(b) limited two classes "iPhone" and "iPad" using radial kernel . . . . .            | 23 |
| 4.9  | SVM Classifier on "reduced activity"-dataset(c) using radial kernel . . . .                                                        | 24 |
| 4.10 | SVM Classifier on "reduced activity"-dataset(c) using linear kernel . . . .                                                        | 24 |
| 4.11 | SVM Classifier on "extensive activity"-dataset(b) limited two classes 'iPhone' and 'ipad' using linear kernel . . . . .            | 24 |

|      |                                                                                                                                 |    |
|------|---------------------------------------------------------------------------------------------------------------------------------|----|
| 4.12 | Frequency Distribution of IP address calls between iPhone and iPad in "extensive activity"-dataset(a) (lowerbound 15) . . . . . | 25 |
| 4.13 | Frequency Distribution of IP address calls between iPhone and Mac in "extensive activity"-dataset(a) (lowerbound 15) . . . . .  | 26 |
| 4.14 | Frequency Distributions between iPhone and iPad using the IP block in "extensive activity"-dataset(a) (lowerbound 10) . . . . . | 26 |
| 4.15 | Frequency Distributions between iPhone and Mac using the IP block in "extensive activity"-dataset(a) (lowerbound 10) . . . . .  | 27 |
| 4.16 | Cross Validation of iPad Ip Blocks from both datasets "extensive activity" and "reduced activity" (lowerbound 5) . . . . .      | 28 |
| 4.17 | Cross Validation of iPhone Ip Blocks from both datasets "extensive activity" and "reduced activity" (lowerbound 12) . . . . .   | 28 |
| 4.18 | Decision Tree on "extensive activity" dataset (max depth 3) . . . . .                                                           | 29 |
| 5.1  | Frequency Distributions between iPad and Mac using the IP block in "extensive activity"-dataset(a) (lowerbound 10) . . . . .    | 47 |
| 5.2  | Frequency Distributions between iPhone and iPad using the company name                                                          | 48 |
| 5.3  | Frequency Distributions between iPhone and Mac using the company name                                                           | 49 |
| 5.4  | Frequency Distributions between iPad and Mac using the company name .                                                           | 50 |
| 5.5  | Cross Validation of iPad Ip Blocks from both datasets "extensive activity" and "reduced activity" . . . . .                     | 51 |
| 5.6  | Cross Validation of iPhone Ip Blocks from both datasets "extensive activity" and "reduced activity" . . . . .                   | 52 |

# List of Tables

|     |                                                                                                       |    |
|-----|-------------------------------------------------------------------------------------------------------|----|
| 2.1 | Structure of a PCAP file [1]                                                                          | 4  |
| 2.2 | Contents of the PCAP global header [1]                                                                | 4  |
| 2.3 | Contents of the PCAP packet header [1]                                                                | 4  |
| 3.1 | Flow features extracted with yaf [12]                                                                 | 13 |
| 3.2 | Corrected Cramers V association with the features and the amount of categories each feature includes. | 14 |
| 3.3 | DNS related features and description [21]                                                             | 17 |
| 3.4 | Sample Flows excluding the IP block and the ASN description                                           | 17 |
| 3.5 | Corrected Cramers V association with the features and the amount of categories each feature includes. | 17 |
| 4.1 | Top 10 company names from DNS information from "extensive activity"-dataset for device "iPhone"       | 30 |
| 4.2 | Duration per Packet (dpp) and Duration per octet (dpo) on the "extensive activity"-dataset            | 31 |
| 4.3 | Average round trip time (rtt) on "extensive activity"-dataset                                         | 31 |



# Appendix

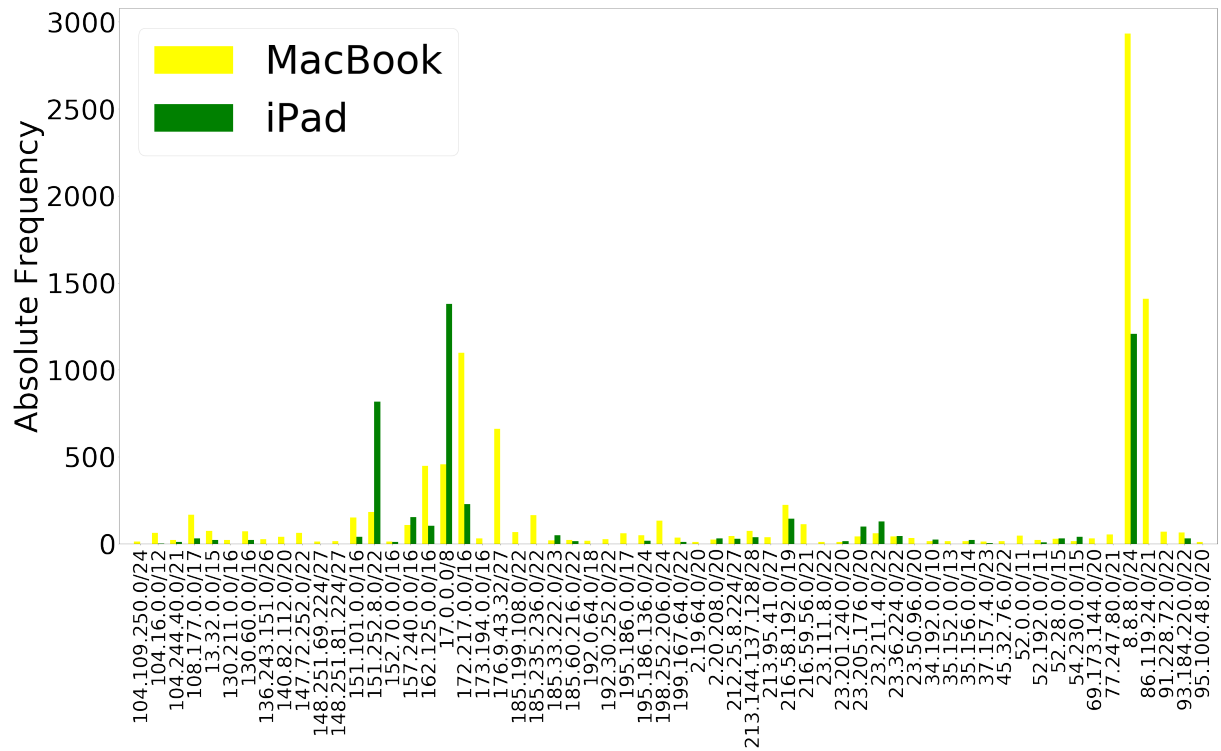


Figure 5.1: Frequency Distributions between iPad and Mac using the IP block in "extensive activity"-dataset(a) (lowerbound 10)

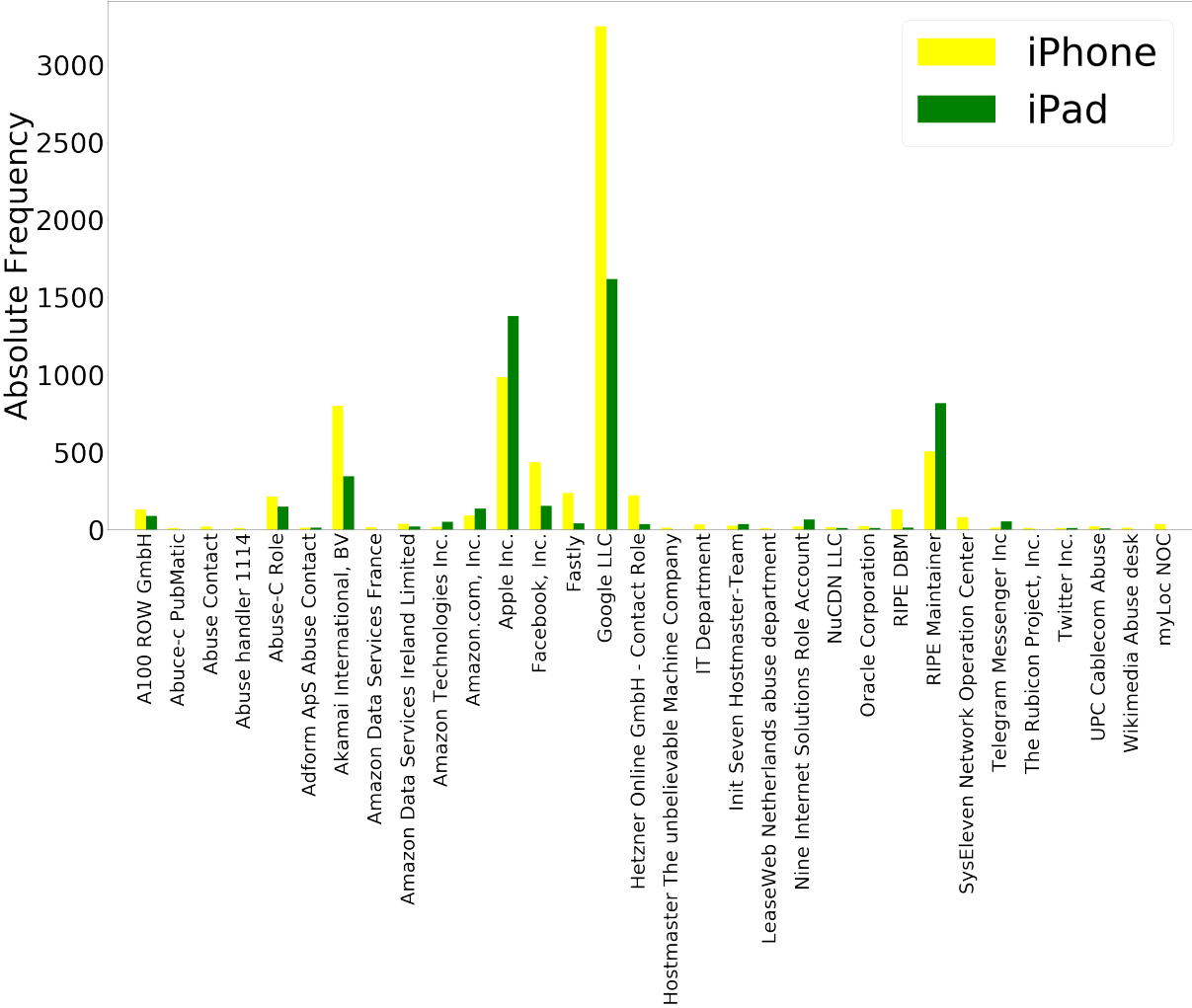


Figure 5.2: Frequency Distributions between iPhone and iPad using the company name

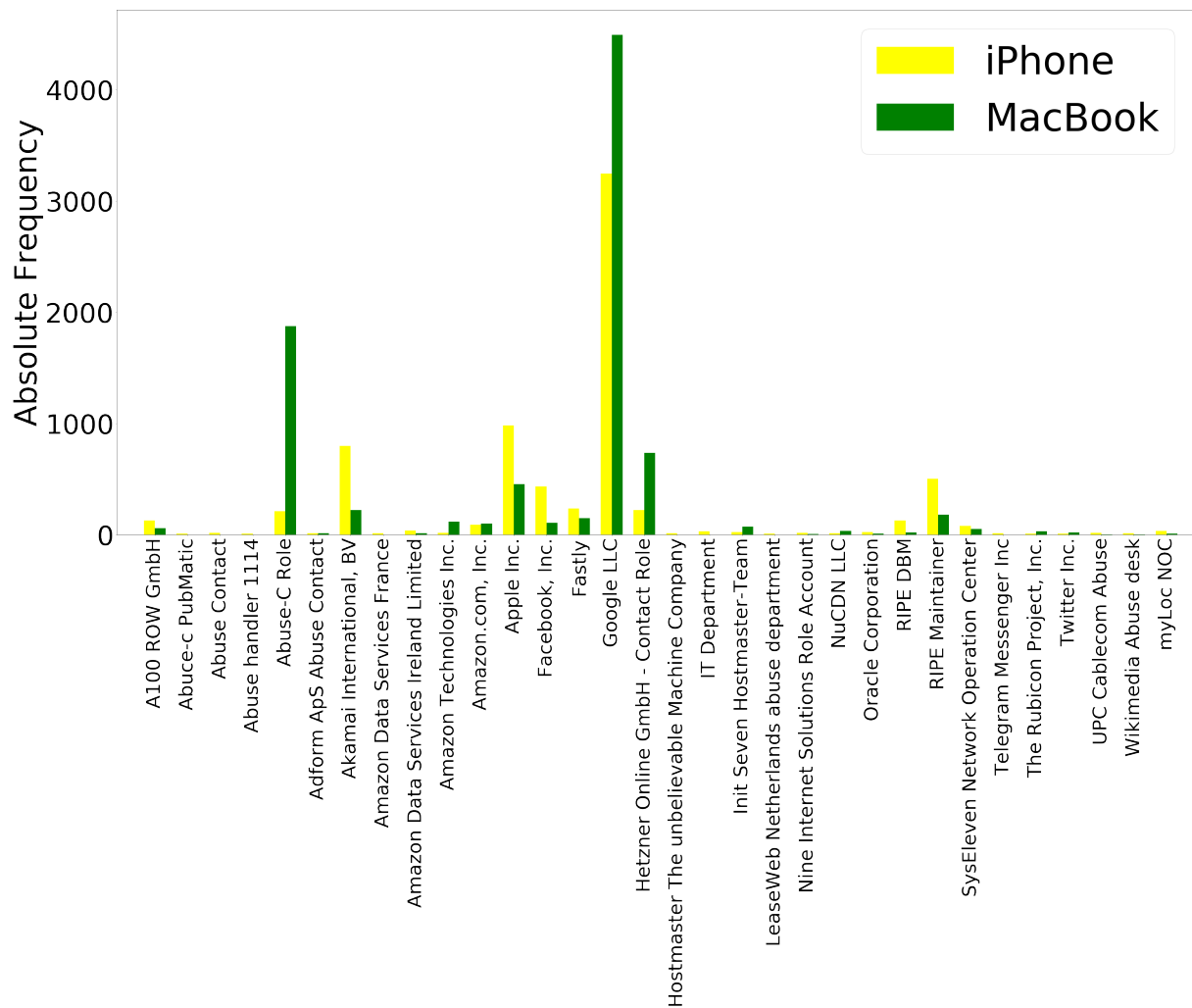


Figure 5.3: Frequency Distributions between iPhone and Mac using the company name

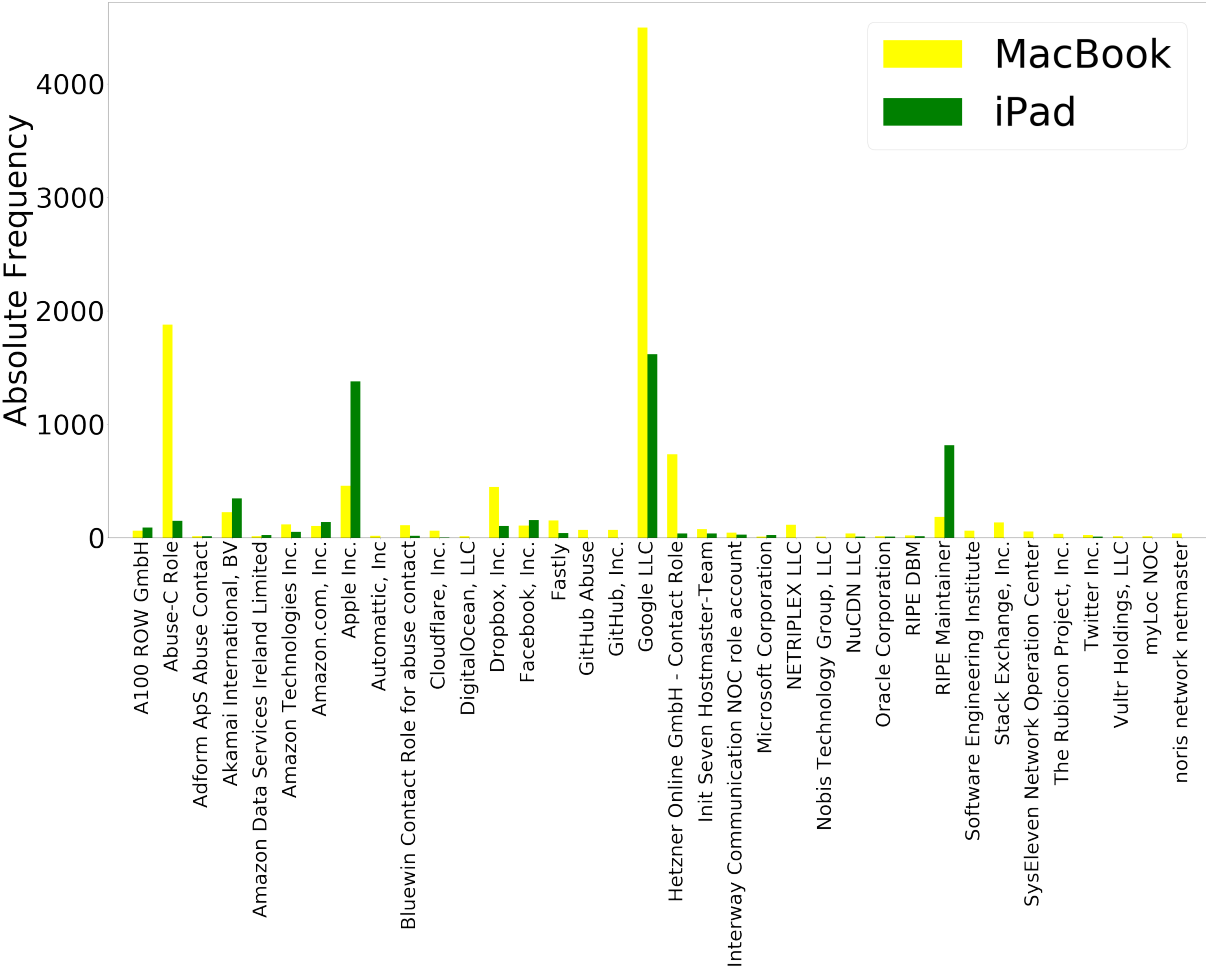


Figure 5.4: Frequency Distributions between iPad and Mac using the company name

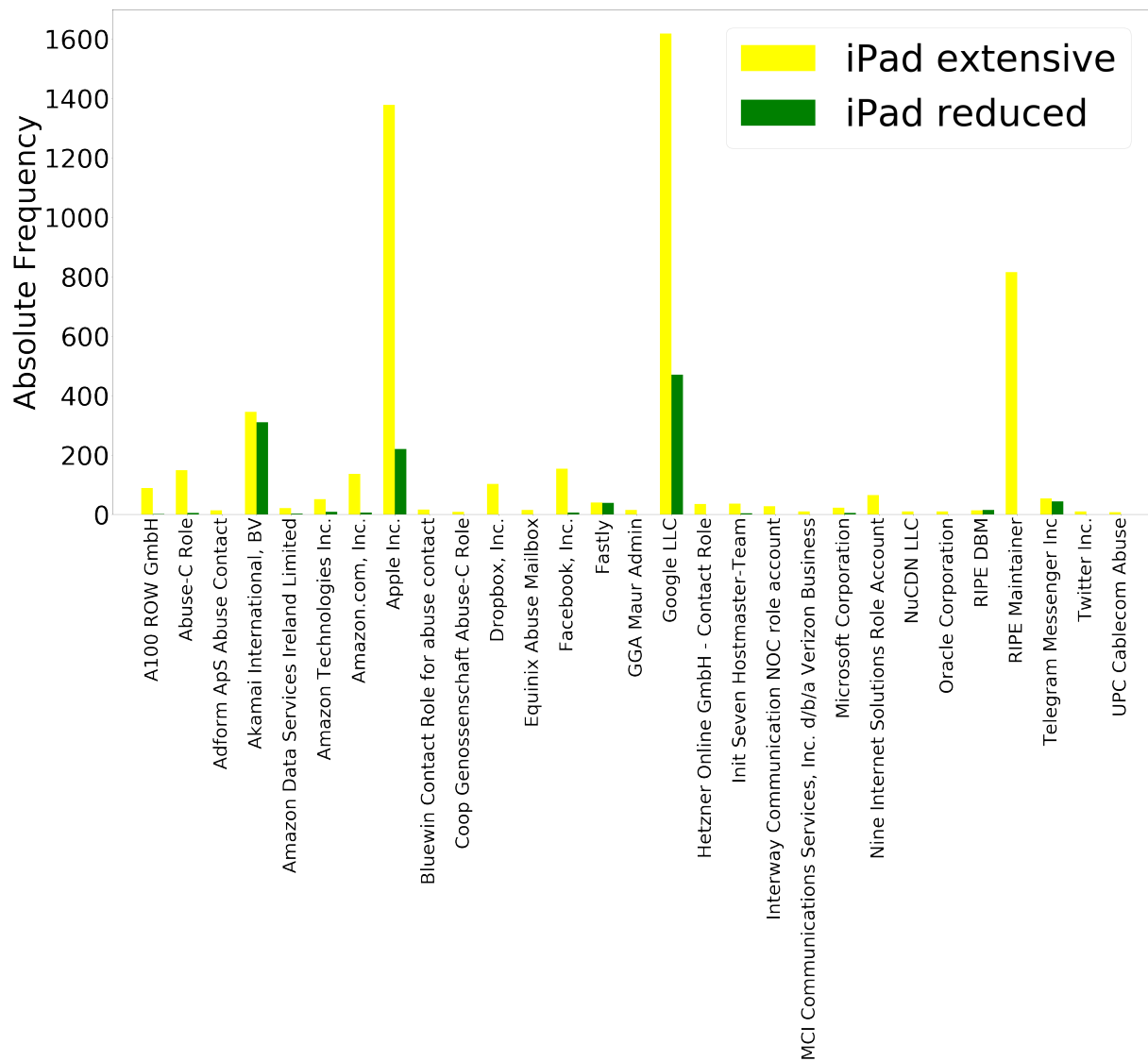


Figure 5.5: Cross Validation of iPad Ip Blocks from both datasets "extensive activity" and "reduced activity"

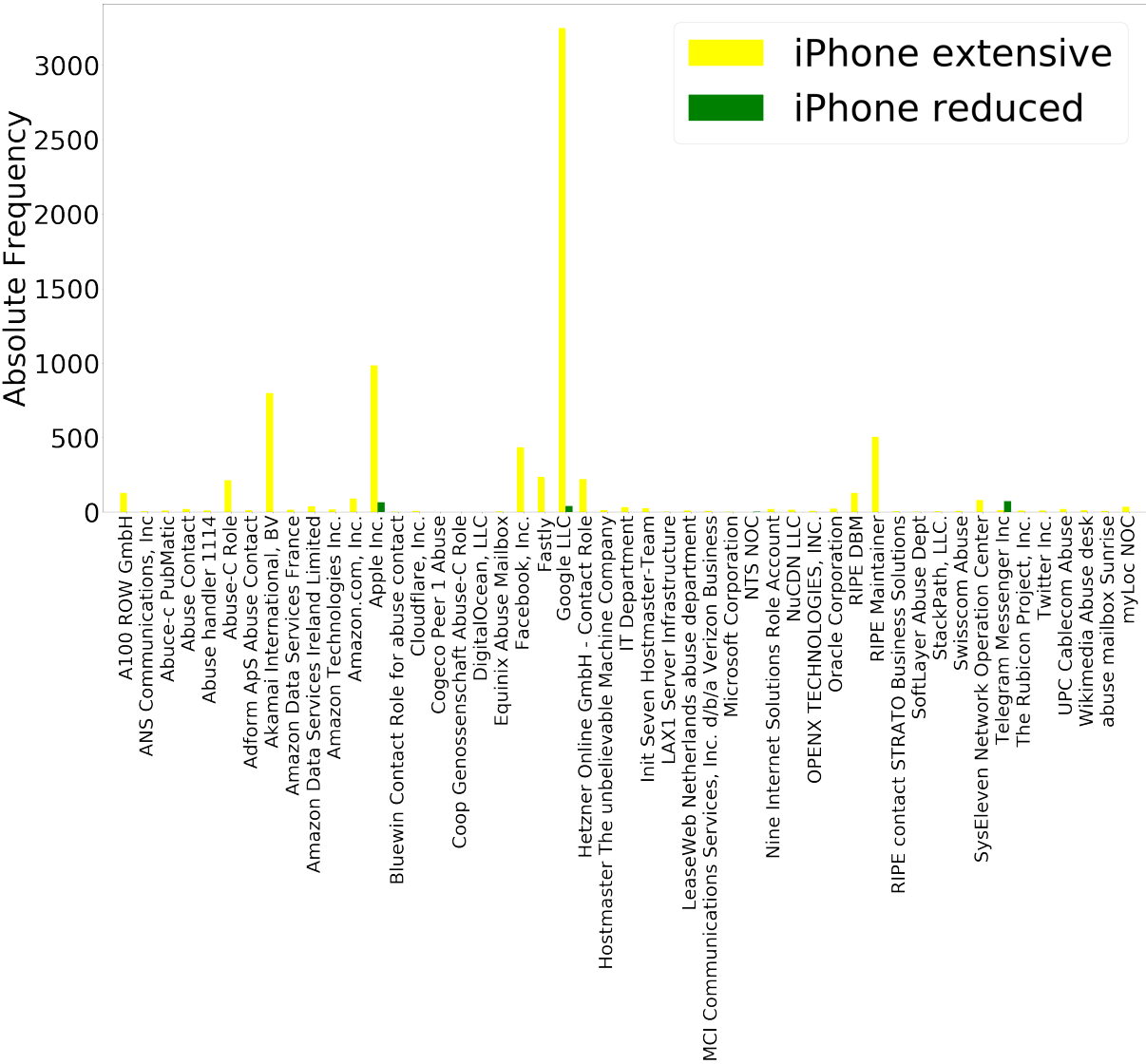


Figure 5.6: Cross Validation of iPhone Ip Blocks from both datasets "extensive activity" and "reduced activity"