POLITECNICO DI TORINO

Master's degree in ICT for Smart Societies

Master's Degree Thesis

Machine Learning for automatic assessment of the risk related to web tracking

Supervisors:
Prof. Marco Mellia
Dott. Martino Trevisan
Dott. Luca Vassio

Candidate:
Marzia Maffei

October 2020

Acknowledgements

I want to thank the supervisors of this thesis, Prof. Marco Mellia, Dott. Martino Trevisan, and Dott. Luca Vassio, who have guided me during the work in the best way possible.

This work marks the end of a long journey and I would like to thank all the people who have shared it with me.

The most important "thank you" goes to Andrea, who has been by my side throughout all the good and the not-so-good of these years, and to my family, my parents and my sister, who have supported and encouraged me every day. And these years in university wouldn't have been the same without my friends, the old ones and the ones I've met along the way, so I want to thank all of them, and especially those who have helped me in these last months of work.

Abstract

This work aims at understanding today's tracking ecosystem and using machine learning tools to automatically assess the risk connected to web trackers and assigning to websites a risk indicator score. The web is a highly dynamic ecosystem and each user browses dozens of websites everyday, encountering a large number of trackers. Trackers serve different purposes, and while some of them help to improve a user's experience on a website, others can be more or less malicious, collecting different kinds and different amounts of data in order to build user profiles, and users are often unaware of their presence. Assigning a risk indicator to websites would make users better aware of the whole web ecosystem and would improve the user's experience as a first step toward a better protection of their data.

In this thesis, machine learning algorithms are used to classify third party domains into non-tracking and tracking domains, based on features extracted from HTTP requests. Then, a risk indicator score is assigned to first party websites depending on the number of trackers contacted and the pervasiveness of these trackers. Trackers that appear on many websites and that collect a high amount of users' data are considered more dangerous in terms of users' privacy.

The classification performs well enough and shows that machine learning algorithms can be considered for the detection of trackers in the web. The estimation of the tracking risk associated to a first party website represents a first step towards a more detailed labelling that should help users to be more aware of tracking practices and how much they are used on websites they wish to visit.

The results of this work, both from the classification part and from the risk indicator score assignment, also give a picture of the web itself and of its tracking ecosystem, showing how much trackers are present, even if they often are unnoticed by users in everyday activities.

Contents

Chapter 1

Introduction

1.1 Web Tracking and PIMCity

Web tracking is the widespread practice of collecting users' data and behaviour from a website. In the last years, personal behavioural data has become a currency of its own and today we live in a data-driven economy, where learning what an individual does with their time allows to more efficiently target products and services and the amount of data a company holds has a direct contribution to its overall market valuation. Online advertising is the most visible and well known product of this data-driven economy, but it is just one of many. The importance of personal data has extended to many sectors, including governance, and it is crucial to find a balance that protects users' privacy and choice.

Users' data can be collected either by the first party, which is the website the user intended to visit, or by a third party domain, different from the visited websites. The information collected about a user can be various: what pages the user visits and how much time they spend on them, website preferences, shopping habits, personal interests, etc. From the perspective of trackers, the larger a user profile it can create, the better service it can provide to its customers, like advertising systems, or to the user, in terms of customisation, for example. From the users' perspective, on the other hand, larger profiles, even when used for performance improvement, might mean a greater loss of privacy.

First party tracking is generally done through cookies, that are primarily used by websites to recognise the user and keep them logged in without the need to re-enter username and password at each request, but they are also used on websites that do not require a log in (e.g. news websites) to remember a user's preferences or track a user's behaviour. The visited website sets a cookie on the user's device, containing a randomly generated ID, and this cookie will be appended to any request made by that device to the website, in this way the website can identify the users and track their behaviour as if they were logged in. Websites want to know which users are behind which pages' visits for reasons that go from improving their product to

serving target ads.

Third party tracking is usually more subtle, the user is often unaware of it, and it can either be single-website or multi-website. Single-website trackers keep data coming from different domains separated and are therefore unable to link a user's activity on different websites. Multi-website trackers, instead, connect a user's behaviour across multiple websites and can build a much more detailed profile of the user, which would allow for more precise targeted advertising, or tailored prices. Third party tracking in general, whether it is single- or multi-website, can be done in several ways, from cookies to more sophisticated methods, like device fingerprinting. In the case of cookies, single-website trackers rely on first party cookies, therefore they assign a different user ID on different websites, not allowing to recognise the same user across websites, while multi-website trackers set their own cookies, identifying the same user across multiple domains. Since cookies can be easily deleted from storage, trackers use also alternative methods. Web beacons, generally a single pixel image or 1×1 GIF, require the user to send a request to download the object, and in doing so the user provides information (e.g. IP address, time of request, type of web browser) about the device that can be used for tracking their behaviour. Device fingerprinting works by associating certain settings of the device (e.g. operating system, type and version of browser, language) to a certain ID, and using this to identify a user across multiple websites.

The research community interest in web tracking is relatively recent, the earliest measurement studies began in 2005 and most of the published works came after 2009, but the practice of web tracking began as soon as the World Wide Web started to grow in the second half of the 1990s. At the beginning, there were few trackers and each website contacted a small number of them, in the early 2000s only 5% of sites made requests to at least 5 third parties [1]. However, several measurement studies throughout recent years have shown that the number of trackers and their pervasiveness is increasing, from the low numbers of the early 2000s the percentage of sites that contacted at least 5 third parties had increased to 40% by 2016. Following the concerns on data privacy raised by the multiple studies, different legislations have been introduced to increase transparency in user tracking and help privacy protection, for example the GDPR (General Data Protection Regulation) in the EU.

In this scenario, where the web is becoming a large data market and there is an increasing need to provide users with control over their data, the European Union funded the project PIMCity [2]. The project implements a PIMS (Personal Information Management Systems) development kit, then combines this with mechanisms to increase users' awareness: the Personal Data Avatar (PDA), that allows users to control the information shared to third parties, and the Transparency Tags (TTs), that show users essential information about the services they access.

The PDA is the interface between the user and the services, thanks to it the user can decide which data to share with which service. From the point of view of the

user, the PDA makes them the only owner of their data, increasing consciousness about privacy; from the point of view of the services (e.g. advertisers), the PDA provides them with information validated by the user, thus more precise than those extracted from opaque tracking companies.

TTs are analogue to Nutrition Labels for food, they should help the users understand the web services they are accessing and the risk these web services pose to their personal data. Starting from privacy metrics (e.g. which information the system is collecting, or if it shares it with third parties), the TTs summarizes the privacy risk of a specific website in a score. The TTs would then give an indication of how much safe a certain website is for the user's privacy, websites that do not pose a threat to personal data are labelled as safe with a low risk score, while websites that are more dangerous, so websites that collect more data, or share these data with many third parties, are instead labelled as dangerous and would have a higher risk score.

1.2 Goal

This thesis is related to PIMCity's Transparency Tags, and its final goal is to have a method to automatically assess how dangerous a website is in terms of tracking. Having an indicator of tracking risk for websites is a first step towards a better protection of users' data, it could improve the users' experience on the web and could make users better aware of the whole web ecosystem and of the presence and pervasiveness of web trackers.

In order to reach the final goal of assigning to websites a score indicating the risk they pose to users' privacy based on the quantity and quality of trackers that they contact, the work is divided into different steps:

1. Data analysis and characterization;
2. Feature engineering and selection;
3. Classification of third party domains into tracking and non-tracking ones;
4. Indicator of tracking risk for websites.

The HTTP requests collected by HTTPArchive [3] form the dataset used for this work. This dataset is analysed, characterizing the three classes in which the web domains are divided (*first party*, *generic third party*, or *tracker*), and from the HTTP requests, features are extracted for each second level domain. These features will be used in machine learning classifiers (decision tree, random forest, support vector machine, k-nearest neighbours) to classify third party domains into tracking and non-tracking ones. After the classification, a risk indicator score is assigned to first party websites depending on the number of trackers contacted and the estimated risk they pose to users' data privacy. The risk a tracker poses to

the privacy of users is evaluated using graph mining to determine the tracker's pervasiveness and using information related to the HTTP requests to estimate how many data the tracker exchanges with the website.

Chapter 2

State of the Art

2.1 Web measurements and tracking ecosystem

The web is more articulated than what appears at a first glance. A user's visit to a single websites often results in multiple HTTP requests being sent to numerous servers under the control of different administrative entities. Some requests are necessary to obtain the content from the site owner's servers or Content Distribution Network (CDN) sites, while others are for tracking a user's movement on the web. Tracking is typically done for analytics, targeted advertising, and other forms of personalization that enhance a user's experience. There are, however, trackers that may have malicious intents and that pose a threat to users' privacy. With the growth of the web and the increased usage of the Internet during the 2000s, concerns about users' privacy rose and many studies on the web ecosystem and the web tracking practice started to be published toward the end of the decade.

Krishnamurthy and Wills [4] published one of the first web measurements focused on tracking, in particular on third party domains that track users by setting third party cookies, by using JavaScript with state saved in first party cookies, or by serving ads URLs with tracking information. Performed between 2005 and 2008 on Alexa's popular sites, their results show that the top 10 third party domains were used by 40% of first party servers in 2005, a percentage that increased to 70% in 2008.

As the web evolved, so did the tracking ecosystem. Acar et al. [5] studied more advanced tracking mechanisms than cookies and URL parameters, they focused on canvas fingerprinting, evercookies and the use of cookie syncing. Canvas fingerprinting is a type of browser or device fingerprinting that uses the browser's Canvas API to draw invisible images and extract a fingerprint to identify the user, evercookies exploit browser's storage mechanisms to restore removed cookies, and cookie syncing allows different trackers to share user identifiers with each other. All these tracking methods were developed to circumvent trackers blocking systems, and they generally succeed to do so, and, unlike common cookies, they usually act without

the user's knowledge.

In 2016, Englehardt and Narayanan [6] performed a large scale measurement of the web on the top 1 million sites, with over 90 million requests. They found out that over 81,000 third parties are present on at least two first parties, but only 123 out of these are present on more than 1% of sites and only major entities, like Google, Facebook, and Twitter, are present on more than 10% of sites. The level of tracking also varies on different categories of sites, news sites being the ones with the higher numbers of third parties, while sites belonging to government organizations, universities, and non-profit entities tend to have lower numbers.

In order to fight malicious trackers and the misuse of users' personal data, several countermeasures have been studied and implemented in the years.

2.2 Ad blockers and other tracking countermeasures

The increasing number and pervasiveness of trackers, the evolution of tracking mechanisms, and the concerns for users' privacy have led to the introduction of new legislations by governments and to the birth of different tracking protection systems, like AdBlock.

Governments have intervened with new legislation to protect users' privacy, in particular the European Union introduced the GDPR, which went into effect on May 25, 2018. Companies that offer services in the EU are required to be compliant with the GDPR, even if their headquarter is located outside of the EU. The GDPR specifies under which circumstances personal data may be processed, and includes rights of data subjects and obligations for those processing personal data of EU-citizen.

There are different actions that can be taken to defend against web tracking, and Roesner et al. [7] analysed the most common defence systems, showing what are some of their limitations:

- Third-party cookie blocking: this defence is often insufficient, if the blocking is not strict enough some websites that are visited directly (e.g. `facebook.com`) can still set cookies on other websites as a third party, on the other hand, a more strict blocking could compromise functionalities like widgets and buttons.

- Clearing client-side state: this allows to regularly receive new identifiers from trackers, which is a way to prevent user's profiling, but once again it would not stop tracking from trackers to which the user has identified as a particular account holder or from trackers that set first-party state on the websites that embed them.

- Blocking popups: this is usually a default setting on browsers, but popups can

still be opened in response to user's clicks and the user can still be redirected to the tracker's domain and back using javascript.

- Disabling JavaScript: it is a blunt method that is very effective at preventing tracking scripts from running, but it also renders much of the web unusable, since most websites require JavaScript to work properly.
- Private browsing mode: cookies are cleared when exiting private browsing mode, which can help protect user's privacy, but the main aim of this method is to protect browser state from adversaries with physical access to the machine, not to defend user's privacy against trackers.

A solution to block ads and disable tracking comes in the form of the several ad blockers and tracker blockers available as browser extensions (e.g. AdBlockPlus [8]). Ad blockers and tracker blockers rely on filter lists to detect tracking coming from known tracking domains and ad-related requests, therefore trackers may easily avoid being blocked by registering a new domain or incorporating tracking behaviour into functional website content. A recent study [9] on filter lists and tracking behaviour showed that filter lists miss from 25% to 30% of trackers, meaning that privacy browser extensions are not able to block all the tracking systems that a user can encounter while navigating the web and that alternatives to filter lists are needed to have a better tracking detection.

Chapter 3

Preliminary work

Studying the dataset and understanding the available data is a fundamental step in order to build the features that will be used for the classification. Such features will then be ordered by their importance with respect to the class, which can either be *first party*, *non-tracking third party* or *tracker*. First party are known, so the classification is performed on all third party domains, which can be trackers or generic non-tracking domains, like content providers. After the classification, first party websites are assigned an indicator of tracking risk based on the quantity and quality of the trackers they contact.

3.1 Dataset

The dataset used for this work has been collected by the HTTPArchive [3], an open source project that tracks how the web is built. It was started in 2010 and it expands on what Internet Archive has already been doing since 1996. While Internet Archive collects and permanently stores the Web's digitalized content, creating a repository of web history, HTTPArchive records how this digitalized content is constructed and served.

In order to build its repository of web performance information (e.g. size of pages, requests, technologies utilized), HTTPArchive crawls millions of URLs on both desktop and mobile monthly. The list of URLs to crawl is taken on the 1^{st} of each month from the Chrome User Experionce Report, which is a dataset of real user performance data of the most popular websites, and the crawling is done on the 1^{st} and the 15^{th} of every month. Each URL is loaded three times with an empty cache and the data from the median run are collected via HAR files, which are parsed to extract meaningful summarized data. HTTPArchive makes a lot of information available via curated reports, but it is also possible to access the raw data, which are available to the public via BigQuery. HTTPArchive provides full HAR tables storing the data of all the crawled pages, of the HTTP requests for

each resource, and of the response bodies for each request, but these tables are extremely large (2.5TB as of August 2018), therefore HTTPArchive also provides summarized tables, which are the ones used for this thesis: `summary_pages` tables and `summary_requests` table. The `summary_pages` tables contain information about the visited pages, including, for example, the page ID and the number of requests (detail can be seen in the example table provided by HTTPArchive in Figure 3.1a). The `summary_requests` tables contain information about all HTTP requests made by the visited pages, so about every single object loaded by all of the pages. The information available from the `summary_requests` tables are shown in Figure 3.1b (example table provided by HTTPArchive) and they include, for example, the object MIME type, the request method and the size of the response body.

From URLs in `summary_requests` tables, I extracted all the second level domains to which requests have been made and that have to be classified as first party, non-tracking third party, or tracking domains. For example, if the URL is `http://www.sample.example.com/index.html`, the second level domain extracted from it will be `example.com`. In order to perform the classification using supervised machine learning methods (decision tree, random forest, support vector machine, k-nearest neighbours), the second level domains must be labelled according to a ground truth. The first party domains are all the domains that have been visited directly, therefore they are stored in the `summary_pages` tables. If a certain second level domain appears in the list of second level domain extracted from the `summary_pages` tables, that domain is labelled as a first party. The tracker domains are instead identified using two common lists of trackers: Disconnect [10] and EasyList&EasyPrivacy (EL&EP) [11]. The Disconnect list contains a list of tracking second level domains divided by category (advertising, analytics, cryptomining, fingerprinting, social); Disconnect identifies a total of 2,220 domains. EasyList and EasyPrivacy are filter lists that work according to a set of rules originally designed for Adblock [12] and that act on the single log or on the domain, EasyList is the primary filter list that removes most adverts from webpages, while EasyPrivacy is a supplementary list thatthat completely removes all forms of tracking from the internet . For this work the list of blocked second level domains has been extracted from EL&EP, and there are 29,386 distinct tracking domains. Together, EL&EP and Disconnect identify a total of 30,371 distinct tracking domains. The domains extracted from the `summary_requests` tables are compared to the domains in the EL&EP and Disconnect lists, and if there is a match, the domain is labelled as a tracker. All the second level domains that are not identified as first party nor as tracker are then labelled as non-tracking third party domains.

3.1.1 Data characterization

For this work, I considered 16 million HTTP requests collected by the HTTPArchive project on February 1^{st}, 2017, for a total of 180,881 distinct second level

pageid	96742262
createDate	1536321692
archive	All
label	Sep 1 2018
crawlid	564
wptid	180901_FR_YRZR
wptrun	2
url	https://www.google.com/
urlShort	https://www.google.com/
urlhash	55122
cdn	Google
startedDateTime	1536317523
TTFB	196
renderStart	400
onContentLoaded	387
onLoad	1198
fullyLoaded	3101
visualComplete	500
PageSpeed	
SpeedIndex	415

rank	
reqTotal	18
reqHtml	6
reqJS	5
reqCSS	0
reqImg	5
reqGif	0
reqJpg	0
reqPng	4
reqFont	2
reqFlash	0
reqJson	0
reqOther	0
bytesTotal	399913
bytesHtml	68452
bytesJS	268461
bytesCSS	0
bytesImg	32220
bytesGif	0
bytesJpg	0

bytesPng	30726
bytesFont	30780
bytesFlash	0
bytesJson	0
bytesOther	0
bytesHtmlDoc	68452
numDomains	7
maxDomainReqs	10
numRedirects	0
numErrors	0
numGlibs	0
numHttps	18
numCompressed	6
numDomElements	381
maxageNull	0
maxage0	6
maxage1	1
maxage30	0
maxage365	11
maxageMore	0

gzipTotal	338311
gzipSavings	0
_connections	6
_adult_site	FALSE
avg_dom_depth	11
document_height	696
document_width	1024
localstorage_size	25
sessionstorage_size	0
num_iframes	0
num_scripts	18
doctype	html
meta_viewport	
reqAudio	0
reqVideo	0
reqText	0
reqXml	0
reqWebp	0
reqSvg	0
bytesAudio	0

bytesVideo	0
bytesText	0
bytesXml	0
bytesWebp	0
bytesSvg	0
num_scripts_async	4
num_scripts_sync	0
usertiming	0

(a) `summary_pages` table example: `google.com` domain

requestid	2404461864
pageid	96977653
startedDateTime	1536471176
time	223
method	GET
url	https://httparchive.org/static/img/ha.png
urlShort	https://httparchive.org/static/img/ha.png
redirectUrl	
firstReq	FALSE
firstHtml	FALSE
reqHttpVersion	orl
reqHeadersSize	376
reqBodySize	
reqCookieLen	0
reqOtherHeaders	

status	200
respHttpVersion	
respHeadersSize	1709
respBodySize	11034
respSize	11034
respCookieLen	0
expAge	43200
mimeType	image/png
req_accept	image/webp,image/apng,image/*,*/*;q=0.8
req_accept_charset	
req_accept_encoding	gzip, deflate, br
req_accept_language	en-US,en;q=0.9
req_connection	
req_host	
req_if_modified_since	

req_if_none_match	
req_referer	https://httparchive.org/
resp_accept_ranges	bytes
resp_age	12
resp_cache_control	public, max-age=43200
resp_connection	
resp_content_encoding	
resp_content_language	
resp_content_length	11034
resp_content_location	
resp_content_type	image/png
resp_date	Sun, 09 Sep 2018 05:32:45 GMT
resp_etag	"1529495684.0-11034-3925150968"
resp_expires	Sun, 09 Sep 2018 17:32:45 GMT
resp_keep_alive	

resp_last_modified	Wed, 20 Jun 2018 11:54:44 GMT
resp_location	
resp_pragma	
resp_server	gunicorn/19.7.1
resp_transfer_encoding	
resp_vary	
resp_via	1.1 google
resp_x_powered_by	
_cdn_provider	Google
_gzip_save	0
crawlid	564
type	image
ext	png
format	png

req_user_agent	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.81 Safari/537.36 PTST/180904.190957
respOtherHeaders	status = 200, content-security-policy = script-src 'self' cdn.httparchive.org www.google-analytics.com use.fontawesome.com cdn.speedcurve.com spdcrv.global.ssl.fastly.net 'nonce-bAbi22ocKQi0HbqA'; img-src 'self' discuss.httparchive.org avatars.discourse.org www.google-analytics.com s.g.doubleclick.net stats.g.doubleclick.net; connect-src 'self' cdn.httparchive.org discuss.httparchive.org raw.githubusercontent.com www.webpagetest.org www.google-analytics.com stats.g.doubleclick.net; default-src 'self'; font-src 'self' fonts.gstatic.com; style-src 'self' 'unsafe-inline' fonts.googleapis.com, x-content-type-options = nosniff, x-content-security-policy = script-src 'self' cdn.httparchive.org www.google-analytics.com use.fontawesome.com cdn.speedcurve.com spdcrv.global.ssl.fastly.net 'nonce-bAbi22ocKQi0HbqA'; img-src 'self' discuss.httparchive.org avatars.discourse.org www.google-analytics.com s.g.doubleclick.net stats.g.doubleclick.net; connect-src 'self' cdn.httparchive.org discuss.httparchive.org raw.githubusercontent.com www.webpagetest.org www.google-analytics.com stats.g.doubleclick.net; default-src 'self'; font-src 'self' fonts.gstatic.com; style-src 'self' 'unsafe-inline' fonts.googleapis.com, strict-transport-security = max-age=31556926; includeSubDomains, x-xss-protection = 1; mode=block, x-frame-options = SAMEORIGIN, referrer-policy = strict-origin-when-cross-origin

(b) `summary_requests` table example: requests of a png image from the `httparchive.org` domain

Figure 3.1: HTTPArchive summary table examples

domains. In order to study the dataset and collect statistics for the feature engineering and selection part, only domains with more than 10 requests have been used, which are 120,660. Out of these, 111,687 have been visited directly, according to the information in the `summary_pages` table of February 1^{st}, 2017, and are therefore labelled as first party domains; 1,231 domains are present in one or both tracking lists (EL&EP and Disconnect) and are labelled as trackers; the remaining 7,742 domains are then labelled as non-tracking third parties.

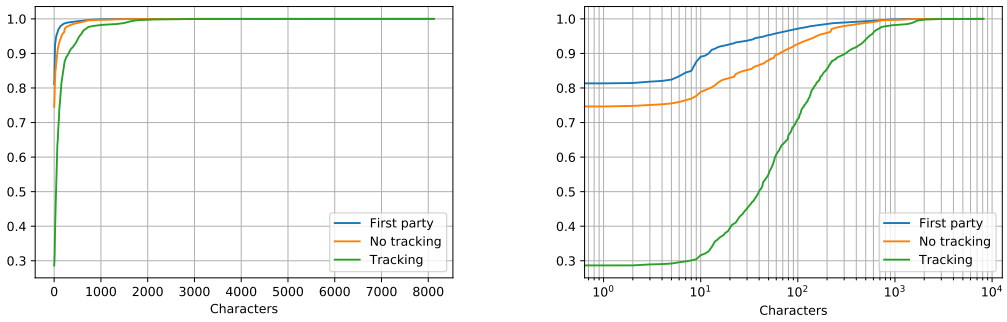
Data have been processed, selecting columns that are useful to characterise the domains (as listed in Table 3.1), and the cumulative distribution of the processed columns have been plotted. As an example, in Figure 3.2 there are the plots showing the distributions of URL length, number of requests and number of referrers for each class. The cumulative distribution of URL lengths (Figure 3.2a) for the trackers class is visibly different from the distributions of first party and non-tracking third party domains, which, on the other hand, are similar to one another. This means

that a feature like the average length of URL in the requests of a domain may be useful to separate trackers from non-tracking domains, while it will not be as much informative to tell apart first party domains from third party ones. The same reasoning can be applied to the number of referrers (Figure 3.2b), whose cumulative distributions show again a difference between trackers and non-tracking domains, while it is similar for first and third parties. The cumulative distribution of the response time is instead very close for all three classes, therefore a feature built as the average of a domain's response times may not be very useful to separate the three classes of second level domains. However, there are small differences among the cumulative distributions in certain points, so taking some of the percentiles as features may help.

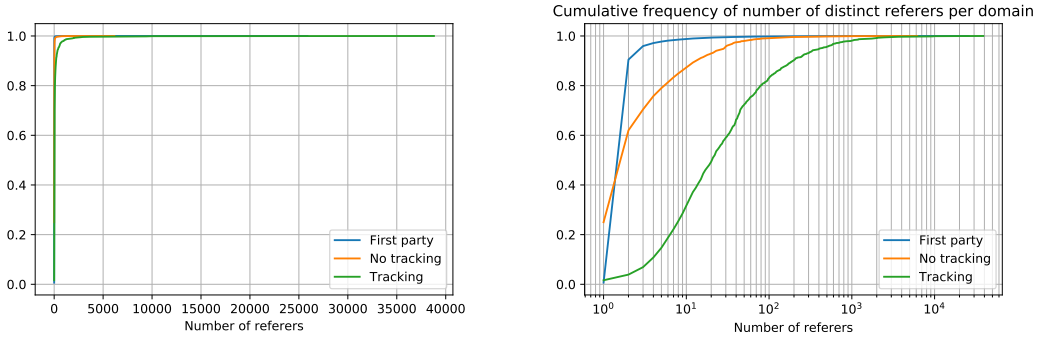
Plots for the cumulative distributions of all the other processed columns can be seen in Appendix B.1.

The `summary_requests` tables also contain the HTTP request method, the object's MIME type, and the object's format and extension extracted by HTTPArchive from the original HTTP requests. In Figure 3.3 it is possible to see the histograms representing the percentages of requests per class for the request methods (Figure 3.3a), objects' types (Figure 3.3b), formats (Figure 3.3c), and extensions (Figure 3.3d). As it can be expected from HTTP requests, the most common method for all three classes is the GET method, which accounts for almost the totality of requests. Therefore, the percentages of requests' methods will likely not be highly informative features. Looking at the types, there is a clearer separation among the classes, especially looking at the image type, which is prevalent in non-tracking domains, while tracking domains tend to have a higher number of objects labelled as "other", which could be uncommon objects' types or objects for which HTTPArchive has not been able to extract the type from the HTTP request. HTTPArchive has also extracted from the original HTTP requests two fields, the object's format and extension, that are similar as a concept to one another and that depend on how the information was extracted from the request. The histograms of the objects' formats and of the objects' extensions present some differences, but they both show a prevalence of .gif objects for trackers with respect to non-tracking domains, while first parties and non-tracking third parties tend to have more .jpg and .png objects, which is consistent with the fact that images are more present in these domains. The histogram of the extensions also shows that .js objects are common in trackers more than in other domains.

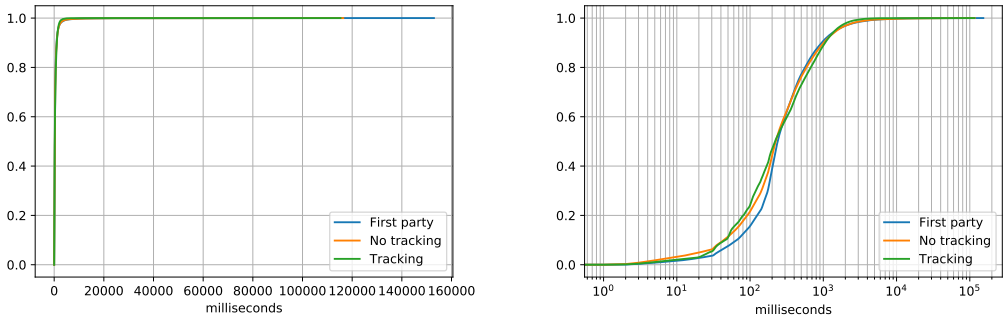
The consistency among types, formats and extensions means that, while some features are likely to be considered very informative by a maximum relevance algorithm, at the same time they can be highly redundant with one another, as they all give the same information to the classifier.



(a) Cumulative distribution of URL length

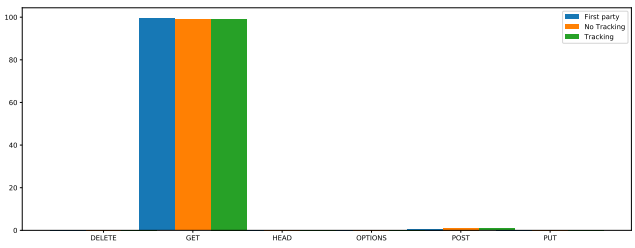


(b) Cumulative distribution of the number of referrers per domain

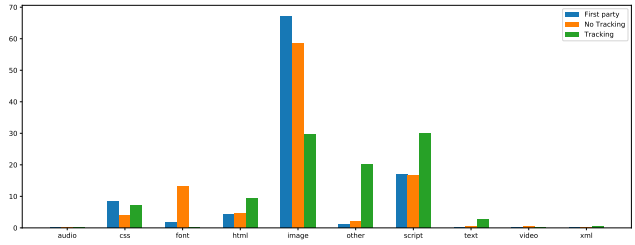


(c) Cumulative distribution of response time

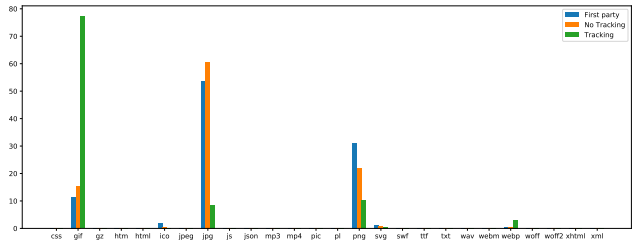
Figure 3.2: Cumulative distributions of different parameters



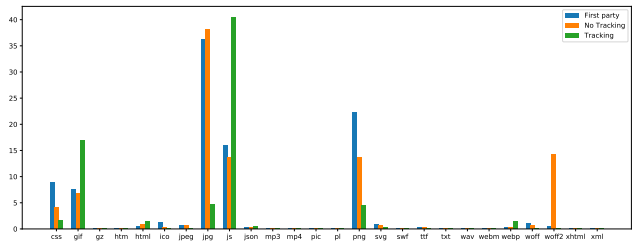
(a) Requests' methods



(b) Objects' types



(c) Objects' formats



(d) Objects' extensions

Figure 3.3: Bar plots with percentages of requests' methods, object types, formats and extensions per class

3.2 Feature engineering and feature selection

Once the second level domains from the dataset have been labelled, it is necessary to extract for each domain features that can be used to classify the third party domains into trackers and non trackers. These extracted features will be ordered by importance according to a criterion of maximal relevance and minimal redundancy, and then the classification can be performed. The classifiers will be applied first using only the four most important features and then slowly increasing the number of features to use, in order to see if all features are necessary to achieve a good performance, or if a subset works better.

The features for each second level domains are extracted from the `summary_requests` tables. The chosen features must characterise the domains and help the classifiers in identifying the ones that perform web tracking. Statistical data, such as the average or the standard deviation, from the most meaningful fields of the `summary_requests` tables are used to build the features of each domain for the classification. The columns taken into account and analysed in order to build the features are listed in Table 3.1.

Field	Description
time	time in milliseconds to complete the request
URL	URL to which the request is made; from here it is possible to extract the URL length, the number of parameters and the length of the parameters
request cookies length	length (bytes) of cookies in the request header
response body size	size (bytes) of the response
response cookies length	length (bytes) of cookies in the request header
expiring age	difference between the expiring date from the cache and the time when the request was made
ETag	value of the entity tag
referer	address of the page making the request
MIME type	request's MIME type
extension	object extension
format	object format
method	HTTP request method

Table 3.1: HTTPArchive `summary_requests` table's field considered in order to build the features

To order the extracted features according to their importance for the classification, a mRMR (minimum Redundancy Maximum Relevance) algorithm for feature selection is used. The mRMR method [13] is an iterative algorithm that combines two criteria: maximal relevance and minimal redundancy. The relevance defines how informative a certain feature is with respect to the class. Two features that carry the same kind of useful information may be highly relevant to the class, but using them both will not actually add any information for the classifier, it would be just redundancy. That is why using only the relevance as a criterion may be misleading and it is better to consider also a redundancy criterion.

The algorithm (1) starts by computing the relevance D of each feature in the set of features F with respect to the class in order to sort them by decreasing D. The most relevant feature is added to the subset of chosen features S and removed from F. For each of the remaining features in F, the algorithm computes the redundancy R with respect to the feature in S, then it evaluates the Mutual Information Difference as $MID = D - R$, and the feature with the highest MID is added to S. The mRMR algorithm repeats the computation of R and MID until all features have been added to S sorted by the combined criteria of maximal relevance and minimal redundancy. Both relevance and redundancy are computed using the Mutual Information, which measures the mutual dependence between two variables.

Algorithm 1 mRMR

for i in $features$ **do**
  compute D_i
end for
add feature with max D to S
for k in range(0, number of features) **do**
  for i in F **do**
    $R_i = \frac{1}{|S|^2} \sum_{j \in S} I(i; j)$
    $MID_i = D_i - R_i$
  end for
  add i to S for i with $\max_{i \in F} MID_i$ and remove i from F
end for

The features built from the HTTPArchive `summary_requests` for each domain are a total of 84 and are listed in Table 3.2.

The features ordered by importance after the application of mRMR method are listed in the first table in Appendix B.2. As it was already visible from the cumulative distributions shown in Section 3.1.1, the number of referrers and the average length of the URL are the most important features for classifying third party domains into tracking and non-tracking domains. According to the redundancy criteria, since the average length of the URL was already selected, the average

length of the parameters, which gives a similar kind of information, resulted last in the order, even though it has a high relevance.

Feature considered	Values
response time	average, standard deviation, and $10^{th}, 25^{th}, 50^{th}, 75^{th}, 90^{th}$ percentiles
URL length	average, standard deviation, and $10^{th}, 25^{th}, 50^{th}, 75^{th}, 90^{th}$ percentiles
request cookies length	average, standard deviation, and $10^{th}, 25^{th}, 50^{th}, 75^{th}, 90^{th}$ percentiles
response body size	average, standard deviation, and $10^{th}, 25^{th}, 50^{th}, 75^{th}, 90^{th}$ percentiles
response cookies length	average, standard deviation, and $10^{th}, 25^{th}, 50^{th}, 75^{th}, 90^{th}$ percentiles
expiring age of the cache	average, standard deviation, and $10^{th}, 25^{th}, 50^{th}, 75^{th}, 90^{th}$ percentiles
ETags length	average, standard deviation, and $10^{th}, 25^{th}, 50^{th}, 75^{th}, 90^{th}$ percentiles
number of URL parameters	average, standard deviation, and $10^{th}, 25^{th}, 50^{th}, 75^{th}, 90^{th}$ percentiles
URL parameters length	average, standard deviation, and $10^{th}, 25^{th}, 50^{th}, 75^{th}, 90^{th}$ percentiles;
distinct referrers	count
distinct response servers	count
objects' types	percentage of css, html, script, image, and other
objects' extensions	percentage of css, gif, html, jpg, js, png, and other
objects' formats	percentage of gif, jpg, png, and other
request methods	percentage of GET, POST, and other

Table 3.2: Features selected

Chapter 4

Classification

The first step in this work is to see if it is possible to recognise tracking domains via machine learning methods, starting from the features extracted from HTTP requests. Given that first party websites are known, since the user deliberately visits them, the goal is to classify the third party second level domains into tracking and non-tracking domains. Out of the total 8,973 third party domains contacted by the first party websites, 1,231 have been labelled as *tracker* using the trackers lists EL&EP and Disconnect, while the remaining 7,742 are labelled as *non-tracking* third party domains.

The classification is performed using classifiers implemented by the scikit-learn library in Python [14]. Four different classifiers are applied: decision tree, random forest, support vector machine, and k-nearest neighbour. Different subset of features are used, starting from the first 4 most informative and then gradually adding the next ones according to the mRMR order, until all 84 are used. For each classifier and for each subset of features, a grid search with a 10-fold cross validation is performed to find the best hyperparameters.

The grid search builds a model on each possible combination of hyperparameters and returns as best combination the one that gives the highest value of the chosen performance metric (e.g. accuracy, precision, F1 score). In this case the performance metric is evaluated using a 10-fold cross validation method, meaning that the training set of data is split into 10 equal parts and the model is trained on 9 subsets and tested on the 10^{th} to evaluate the performance metric, and the process is repeated 10 times, each one using a different subset as test and training on all the others. The final performance metric for that combination of hyperparameters is the mean of the performance metric across the 10 validations.

4.1 Classification metrics

The performance metric chosen to evaluate the classifiers in this thesis is the macro F1 score, which combines precision and recall and is computed as the average of the F1 scores of the two classes.

Precision and recall for each class are based on the number of True Positives (TP), False Positves (FP) and False Negatives (FN) in a class. TP is the number of correctly predicted samples in a class; FP is the number of samples predicted in a class but belonging to the other one; FN is the number of of samples belonging to a class but predicted in the other one. Then, precision, which shows what proportion of positive identifications is actually correct, and recall, which indicates what proportion of actual positives was identified correctly, are defined as:

$$Precision = \frac{TP}{TP + FP};$$
$$Recall = \frac{TP}{TP + FN}.$$

The F1 score combines precision and recall and it is defined as the harmonic mean of the two metrics:

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

In the case of a balanced dataset, the F1 score is equal to the accuracy, which is the fraction of overall correct predictions; if the dataset is heavily imbalanced, so if one class has considerably more samples than the other one, accuracy becomes unreliable, because it is biased by the class with the more samples, and therefore it is better to rely on the F1 score.

4.2 Classifiers

Decision trees are among the most used methods of classification because they are easy to understand and interpret, require little data preparation, and the cost of predicting data is logarithmic in the number of data points used to train the model. Decision tree algorithms divide the data into subsets, in a binary way, after evaluating at each step the optimal threshold for the feature taken into account at that step. The decision tree starts the splitting from the most informative feature selection, so it performs its own feature selection. On the other hand, decision trees have some disadvantages, they can be unstable, so that small variations in the data result in very different trees being generated, and they can easily go into overfitting. For the decision tree, the parameters that have been tested are the criterion to measure the quality of a split and the maximum depth of the tree (Table 4.1a).

Random forests are ensemble methods based on decision trees and they introduce randomness in the classifier construction. The prediction of the ensemble is given as the averaged prediction of the individual classifiers. Individual decision trees may exhibit high variance and may overfit, and the use of a random forest algorithm can reduce the variance by combining various trees, usually leading to an overall better model. For the random forest, the parameters tested have been the number of trees, the maximum depth of the trees, and the criterion to measure the quality of a split (Table 4.1b).

Support vector machine (SVM) methods construct a hyper-plane, or set of hyper-planes, in a high dimensional space, and use this for classification. A good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class, the larger this functional margin is, the lower the generalization error of the classifiers is. The support vectors are the samples closest to the margin boundaries, or the samples within the margin boundaries if the problem isn't linearly separable. SVMs are effective in high dimensional spaces, but the fit time of the SV Classification implementation in scikit-learn scales at least quadratically with the number of samples and may be impractical with a very high amount of data. For the SVM the tested parameters are C, which is the regularization parameter, and the kernel function to be used (Table 4.1c).

Nearest neighbours (NN) is one of the simplest, most intuitive methods for classification, but it has been proven successful in a large number of classification problems, especially when the decision boundary is very irregular. The classification is computed from a simple majority vote of the nearest neighbours of each point: the point is assigned the class which has the most representatives within its k nearest neighbours. The main disadvantage of k-NN is that it may be affected by the curse of dimensionality with high dimensional data, that make difficult to calculate distances and find the nearest neighbours of a given point.The k-NN has been tested with different values of k and with two different weight functions: *uniform*, meaning that all points are weighted equally, or *distance*, which weighs the points according to the inverse of distance, so closer points have a greater influence than distant ones (Table 4.1d).

The best classifier with the best set of hyperparameters is the one that has the highest average F1 score over the ten folds of validation.

4.3 Results

The four different classifiers have been applied with different numbers of features, starting with the first four features by importance and gradually adding the next ones to the subset, until all 84 features were used. The training and validation part, used to choose the best set of hyperparameters, was performed on the 70% of samples of each class, while the remaining 30% was used to test the classifiers with the chosen values of the hyperparameters. The optimal hyperparameters for each

Parameter	Values
Criterion	*gini*, *entropy*
Tree depth	6, 8, 10, 12, 14, 16, 18, 20, 22, 24

(a) Decision Tree

Parameter	Values
Criterion	*gini*, *entropy*
Tree depth	6, 12, 18, 24
Number of trees	50, 75, 100, 125

(b) Random forest

Parameter	Values
C	1,5,6
Kernel function	*linear*, *RBF* (Radial Basis Function)

(c) Support Vector Machine

Parameter	Values
k	2, 5, 10, 20
Weight function	*uniform*, *distance*

(d) k-Nearest Neighbours

Table 4.1: Hyperparameters for the different classifiers

classifier are reported in Table 4.2 and the F1 score of each classifier with respect to the number of features is shown in Figure 4.1a.

Classifier	Parameters
Decision Tree	criterion=*gini*, max depth=16
Random Forest	criterion=*entropy*, max depth=10, number of trees = 125
SVM	C=10, kernel=*linear*
k-NN	$k = 10$, weights=*distance*

Table 4.2: Best hyperparameters of each classifier

From the values of the F1 score on validation, the classifier that has the best performance is the Random Forest, with 40 features and the hyperparameters in Table 4.2. The other plots in Figure 4.1 show the values of the other classification metrics and accuracy values (Figure 4.1b) are higher for every classifier if compared

with the respective F1 score values. This happens because the dataset is slightly imbalanced, with the non-tracking class having 7 times more samples than the trackers class.

The Random Forest classifier with the best parameters is then applied on the test set, which is made of samples never seen in the training step, considering the first 40 most informative features. The resulting F1 score is equal to 0.78, while the values of precision and recall per class are reported in Table 4.3. Both precision and recall are higher for the non-tracking class, and this is an expected result since the non-tracking class is more numerous than the other one.

class	precision	recall
non-tracking	0.96	0.96
tracking	0.59	0.62

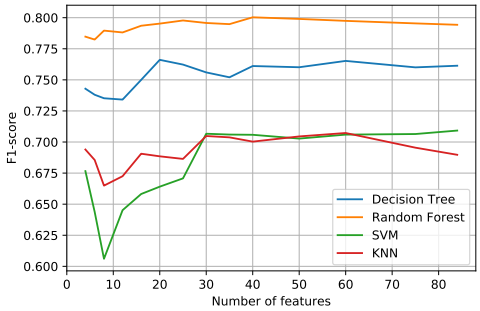
Table 4.3: Precision and recall per class on the test samples

These results can be compared to those of a classifier trained and tested on a balanced dataset. By performing the cross-validation on an equal number of non-tracking domains and trackers, the F1 score (Figure 4.2) increases for all classifiers with respect to the values obtained in the imbalanced case. In particular, for the best classifier, which is again the Random Forest, the F1 score raises from 0.8 to 0.9. The performance is equally good on a balanced testing set, on which the Random Forest with the best hyperparameters (Table 4.4), using the first 40 most informative features, returns a F1 score of 0.89.

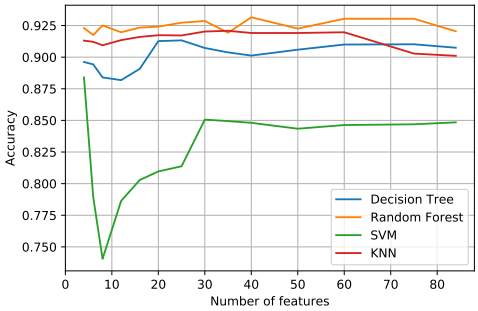
The results on a balanced dataset show that the performance can be very good, however the results on the imbalanced dataset, which reflects the actual proportion of non-tracking domains and trackers in the web, can be also considered a good starting point for the detection of tracking domains.

Looking into the second level domains that the classifier labelled as *tracking*, but that are not present in either EL&EP or Disconnect list, it is worth to note that while some are mistakes made by the classifiers, others are domains that provide advertising services on the web (for example `uadexchange.com`) and therefore they probably rely on users' personal data, that may be collected by themselves or by other trackers that then share the information with them. And some other domains are actual trackers (for example `ad4mat.net`) that are not identified by EL&EP and Disconnect, which are the two most commonly used lists for trackers.

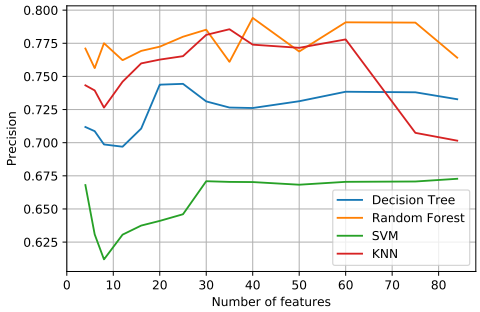
A second classification to separate first party domains from all third party ones (both trackers and not) has been performed during this thesis, and the results are reported in Appendix A.



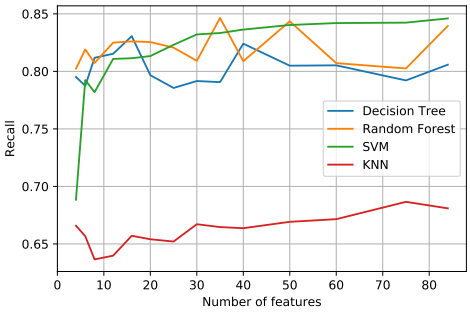
(a) F1 score



(b) Accuracy



(c) Precision



(d) Recall

Figure 4.1: Classification metrics on validation set

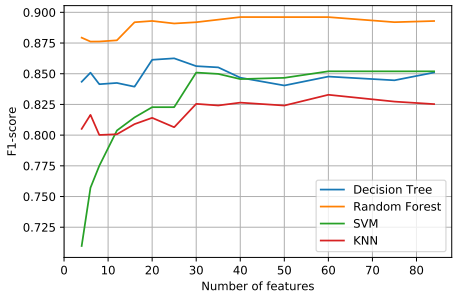


Figure 4.2: F1 score after validation on a balanced dataset

Parameter	Value
criterion	*gini*
max depth	14
number of trees	75

Table 4.4: Best hyperparameters of Random Forest on a balanced dataset

4.4 Impact of time

In order to see if a machine learning method evaluated on a given year retains the same performance as the web changes, the best classifier has been applied on data

collected in a different period. Two datasets of 16 million requests, always collected by HTTPArchive, one on February 1^{st}, 2018, and one on February 1^{st}, 2019, have been considered, and a Random Forest classifier with the hyperparameters reported in Table 4.2 has been trained and tested on the new datasets. The training has been performed on the 70% of samples and the test on the remaining 30%, as it was done for the 2017 dataset.

The number of second level domains per class in each dataset and the results of the classification are reported in Table 4.5, along with the corresponding values from the 2017 dataset. The values in the table show that, as the number of requests considered is the same for all three years, the number of first party domains decreases consistently, while the number of third parties, especially from 2018 to 2019, and trackers increases. Overall, each first party websites makes a larger number of requests to third parties, both tracking and non-tracking. The F1 score is consistent across the years, showing that a classifier evaluated in 2017 retains the same performance at a two years distance.

	2017	2018	2019
first party	111,687	87,220	53,808
third party	7,742	7,970	41,013
tracker	1,231	1,325	1,360
F1 score	0.78	0.78	0.79

Table 4.5: Comparison of dataset and results on different years

Chapter 5

Indicator of tracking risk

Once the trackers have been recognised by the classifier, first party websites are assigned a score estimating the threat they pose to users' privacy. This score depends on the quantity and quality of trackers that each first party embeds, the higher the number of trackers and the deeper their pervasiveness, the more dangerous a website is considered.

The estimated risk of a website i is based on the risk associated with every single tracker j embedded in it, which has been summarized in three different components:

- tracker j's popularity;
- the information exchanged between website i and tracker j;
- the ability of tracker j to exchange information with other trackers.

The popularity of a tracker is given by the probability of encountering that tracker on a website. Trackers with high popularity are trackers that appear on many web pages and therefore have the possibility of collecting a large amount of users' data. The probability p_j of encountering tracker j is computed as the number of distinct websites tracker j appeared on divided by the maximum number of distinct websites over all the considered trackers. Then, if I is the set of trackers contacted by website i, the first component of the risk is:

$$f_{1,i} = \sum_{j \in I} f_{1,ij} = \sum_{j \in I} (p_j \cdot \frac{\log(p_j \cdot N)}{\log N}),$$

where N is the total number of first party websites considered. The logarithm is applied in order to smooth the function and prevent the very few most popular trackers from weighing too much.

The more information a website i exchanges with trackers, the higher is the chance that the trackers can collect users' personal data, so the website can be considered more risky. To estimate the quantity of information exchanged between

website i and tracker j, two different components have been considered: the total number of parameters in the URL of all requests from website i to tracker j and the total length of cookies set by tracker j on website i. Therefore, the second component of the risk is formed by two functions: $f_{2,i} = \sum_{j \in I}(f_{\text{URL},ij} + f_{\text{cookie},ij})$. Each of these two functions is built as:

$$f_{x,ij} = \frac{\log(x_{ij}) \cdot \frac{\log(x_{ij})}{\max_{j \in I}(\log(x_{ij}))}}{\max_{j \in I}(\log(x_{ij}) \cdot \frac{\log(x_{ij})}{\max_{j \in I}(\log(x_{ij}))})},$$

where x_{ij} is the total number of URL parameters exchanged between i and j for $f_{\text{URL},ij}$ and the total length of cookies set by j on i for $f_{\text{cookie},ij}$. The logarithm is used to smooth the function as in f_1, i, and the function is divided by the maximum over all the trackers contacted by i to normalize the single components.

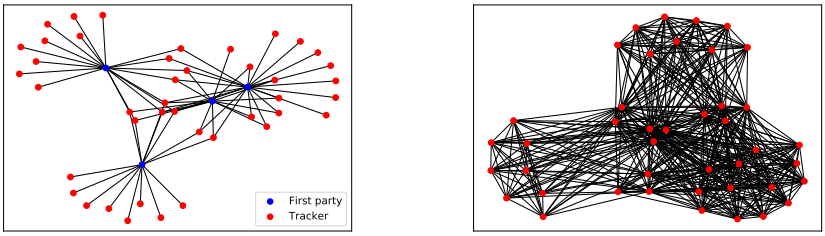
The third component is extracted using graph mining. At first the bipartite graph (Figure 5.1a) with the connections between first party websites and trackers is built, so there are the nodes labelled as first parties and the nodes labelled as trackers. This bipartite graph is then projected onto the trackers (Figure 5.1b), so two tracker nodes are connected if they have at least one first party website in common. From the projected graph the closeness centrality (CC) measure of each tracker is evaluated. The closeness centrality is a graph metric that is calculated as the reciprocal of the sum of the length of the shortest paths between each node and all other nodes, giving an indication of how close a node is to all the others, and therefore how central it is in the graph. A tracker that has a high value of closeness centrality is a tracker that can reach many other trackers and so it can exchange information about a user with these trackers, for example by performing cookie syncing with them [5, 15], thus posing a potentially higher threat to a user's privacy. The third component of risk is therefore estimated as:

$$f_{3,i} = \sum_{j \in I} f_{3,ij} = \sum_{j \in I} \text{CC}_j.$$

Each component is summed over all the trackers contacted by the first party website and scaled between 0 and 1, so that they are all in the same range. The resulting three values can be weighted to have more or less importance in the final computation of risk.

Therefore, if I is the set of trackers contacted by website i and c_k is the weight given to each component, the total risk, ranging from 0 to 1, of a website i is estimated as:

$$R_i = c_1 \cdot f_{1,i} + c_2 \cdot f_{2,i} + c_3 \cdot f_{3,i} = c_1 \cdot \sum_{j \in I} f_{1,ij} + c_2 \cdot \sum_{j \in I}(f_{\text{URL},ij} + f_{\text{cookie},ij}) + c_3 \cdot \sum_{j \in I} f_{3,ij}.$$



(a) Bipartite graph with first party and tracker nodes

(b) Projected graph with links between trackers that have at least one first party in common

Figure 5.1: Example of small bipartite graph and graph projected onto trackers

5.1 Results

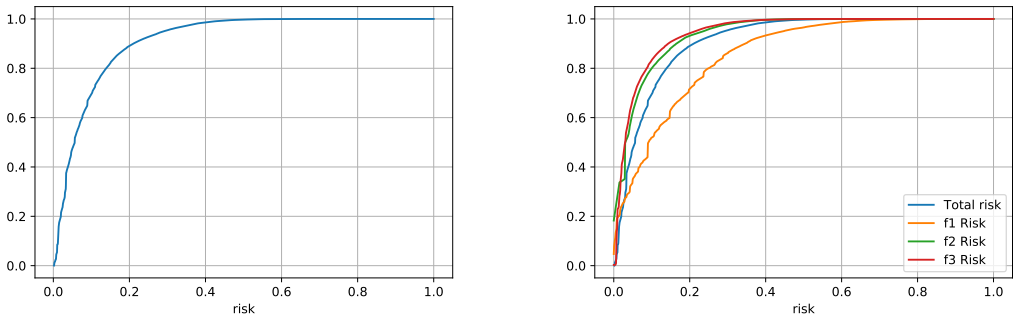
To obtain a first estimation of the risk, the second level domains considered as trackers are the ones labelled according to the trackers lists EL&EP and DIsconnect.

The 69% of first party websites considered embed some kind of tracker, and therefore have an estimated risk greater than 0, the remaining 31% do not embed trackers according to the HTTP requests considered. Setting all the weights $c_k = 1/3$, giving the same weight to all components, and considering the first party websites that have a risk greater than 0, the resulting cumulative distribution of the tracking risk values is shown in Figure 5.2a. The distribution is heavily shifted towards the left side of the graph, so towards small values of estimated risk, and has a long tail for larger values. In fact, it can be seen from the plot that almost 90% of the first party websites has an estimated risk lower than 0.2 on a [0,1] range.

The cumulative cumulative distributions of the three different components of the risk with respect to the final total risk can be observed in Figure 5.2b. By considering only the first component f_1, indicating the popularity of the contacted trackers, the distribution is shifted towards higher values of risk, the tail becomes less prominent and the percentage of websites with a risk lower than 0.2 decreases from almost 90% to 70%. On the contrary, the distributions of the other two components, f_2 and f_3, are shifted to lower values of risk, with longer tails.

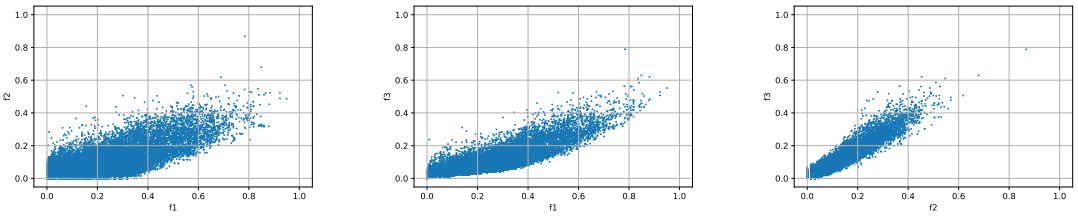
From the scatter plots in Figure 5.3 it is possible to see the correlation between the different components, so how each components varies with respect to the other two. The closer the distribution of the dots in the plot is, the more correlated the two components are. For example, the scatter plot in Figure 5.3c clearly follows the trend of a straight line with positive slope, so there is a positive correlation between the components f_2 and f_3. A positive correlation means that if f_2 has a high value, f_3 is expected to have a high value too, and vice versa. This means that

f_2 and f_3 return the same kind of information for most of the considered websites. The first plot (Figure 5.3a) shows that f_1 has a weaker correlation to the values of f_2, so the information it carries is different, while there is a more visible correlation with f_3 (Figure 5.3b). A correlation between f_1 and f_3 is expected, since they are both related to the popularity of the tracker, the first one reflects the probability of encountering the trackers on a first party website and the other one reflects the connection of the trackers with other trackers, which partly depends on the number of first party websites a tracker appears on.



(a) (b)

Figure 5.2: Cumulative distributions of risk values



(a) f_2 vs f_1 (b) f_3 vs f_1 (c) f_3 vs f_2

Figure 5.3: Scatter plots of the different components of the risk with respect to one another

Ordering the websites by increasing total risk, it is possible to see the variation of the three components (Figure 5.4). The f_3 component, extracted from the trackers graph and representing the closeness centrality of the embedded trackers, is the most uniform out of the three, while the other two, especially f_1, are wider and less consistent across the websites. This plot also reflects the cumulative distributions, since f_1 has visibly higher values than the other two, which in contrast are in a similar range, even though f_2 is slightly wider than f_3.

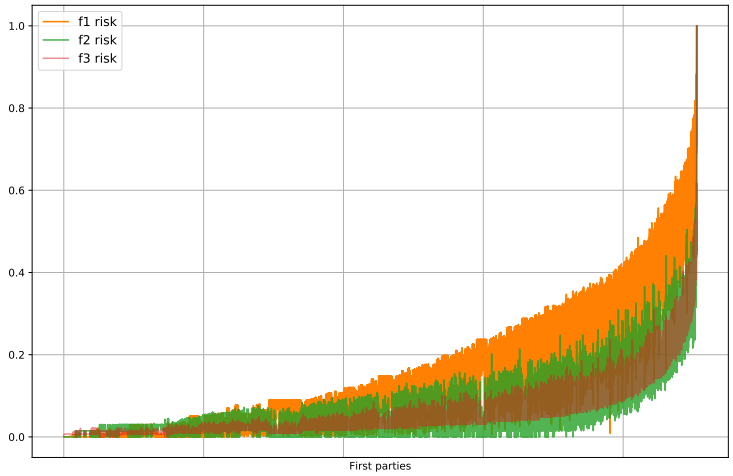


Figure 5.4: Risk components ordered by increasing total risk

In order to make this indicator of tracking risk more intuitive and highlight the first party websites that can potentially threaten the safety of users' personal data, a coloured labelling (tag) could be used instead of a numeric score ranging between 0 and 1. Similar to traffic lights, the colours chosen for the tag are three: *green* for the websites that can be considered safe from a privacy point of view, *yellow* for the websites that may include a larger number of trackers or trackers with higher pervasiveness, an *red* for the websites that are considered at high risk of tracking because they embed many trackers with very high pervasiveness.

Considering the very long tail of the distribution, the intervals for the tags have been chosen to balance it. Therefore, as it can be seen in Figure 5.5, the *green* tag is assigned to the websites that have an estimated risk between 0 and 0.2, which are the 89% of the considered websites, the *yellow* tag to websites with a risk greater than 0.2 and lower than 0.5, and the *red* tag to websites with a risk from 0.5 to 1, which are the 0.3%.

The top 50 most risky websites obtained with this estimation are reported in Table B.3, along with the values for the total risk and the three separate components. The most risky second level domain is `blogspot.com`, that has the highest values of all three components and, consequently, of the total risk. Since Blogspot is a platform for web blogs, it has several subdomains (like `example.blogspot.com`) that lead to different pages, and each of these pages could embed many trackers. It is expected that a second level domain of this type, that includes so many subdomains, would turn out to be at high tracking risk, so high in fact that there is quite a big distance between it and the second most risky domain (`adobedtm.com`).

By looking at the rest of the table, it is possible to see that the values of f_1 are generally higher than those of the other two components, as it was already visible from the cumulative distributions and the plot in Figure 5.4.

It is important to note that these results depend on the number of requests taken into account (16 million) and the number of first party websites and trackers involved in such requests, thus different results may be achieved by extending the dataset with a larger number of requests or choosing a different dataset altogether.

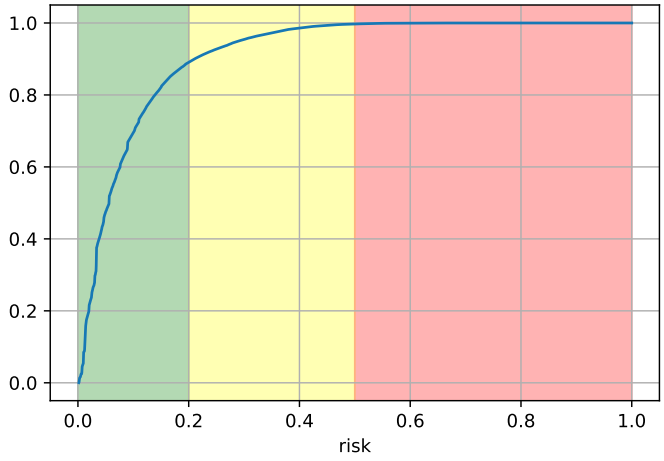


Figure 5.5: Possible labelling of first party websites according to the indicator of tracking risk

Chapter 6

Conclusions

The results obtained during this work show that it is possible to classify third party domains into non-tracking and tracking ones using machine learning algorithms and starting from HTTP requests. Informative features can be easily built from information like the length of the request's URL, the size of the response's body, or the object's format from the HTTP requests made to a certain second level domain. Classifiers that rely on these features have been shown to have a good performance, in particular Random Forests models, and are able to identify a large portion of the trackers labelled as such from the trackers lists (EL&EP and Disconnect). It is worth to note that the trackers lists used as ground truth for the labelling of second level domains are not exhaustive and they have been showed to miss some trackers [9], thus, while the classifier may miss some of the trackers listed in EL&EP and Disconnect, it may be able to identify others that have been missed by the lists.

The second goal of this work was to start the development of a method to assign an indicator of tracking risk to first party websites. To obtain an estimation of the tracking risk of a website in terms of users' privacy, different components have been taken into account (popularity of the embedded trackers, quantity of information gathered by such trackers, and their capability of exchanging this information with other trackers) and used to compute the associated risk. As for the classification, HTTP requests can be used to define the risk: the number of parameters in a request's URL and the length of cookies can give an estimation how many information are exchanged between a tracker and the first party that embeds it. Along with the data extracted from HTTP requests, other information can be gathered from the graph representing trackers connections, for this work the closeness centrality was considered to obtain a measure of how easily a tracker can exchange data with other trackers.

Starting from the results presented in this thesis, further studies are necessary to understand how the risk associated to a website may change in time and to better define the concept of tracking risk itself, especially what is perceived as a

privacy threat by everyday users of the web. Taking into account which kind of personal data users are more willing to share and which not, and how trackers may use those data, would help to build detailed TTs (Transparency Tags) for websites. Used along with a PDA (Personal Data Avatar), detailed TTs will improve users awareness of the web tracking ecosystem and help protect data privacy.

Appendix A

Classification of domains into first party and third party domains

First party domains are usually known, since they are visited directly by the user when navigating the web. However, given a dataset made of HTTP requests, like the one built by HTTPArchive, it is interesting to see how well a classifier would perform in separating first party domains from third party ones, both trackers and non-trackers. As for the third party classification, the mRMR method has been used to order the features according to the combined criteria of maximal relevance and minimal redundancy. The ordered features are reported in Table B.2.

For this classification, the second level domains labelled as *tracker* or *generic third party* have been merged into a single *third party* class. The classifiers that have been applied are the same used for the tracking vs non-tracking classification and the grid search was performed with a 10-fold cross validation over the same values of hyperparameters (Section 4.2). The training step of the classification has been performed both on the imbalanced dataset, in which the number of first party domains is more than 10 times the number of third party ones, and on a balanced number of samples.

Considering once again the mean F1 score across the 10 folds of validation as the performance indicator, the best classifier is the Random Forest with the hyperparameters in Table A.1 and using all 84 features, resulting in a F1 score of 0.85, which can be considered a good performance. The chosen classifier has then been tested on previously unseen data, returning a F1 score of 0.85 and the values of precision and recall per class reported in Table A.2.

In the balanced case, the F1 score is shown in Figure A.1 and the best hyperparameters for the Random Forest using the first 50 most informative features are reported in Table A.3. Using the best classifier on a balanced test set made of

previously unseen samples, the F1 score reaches a value of 0.88, which is equal to the performance of the tracking vs non-tracking classifier on a balanced dataset, and it can be considered a good result.

Parameter	Value
criterion	*gini*
max depth	16
number of trees	100

Table A.1: Best hyperparameters of Random Forest

class	precision	recall
first party	0.98	0.98
third party	0.70	0.75

Table A.2: Precision and recall per class on the test samples

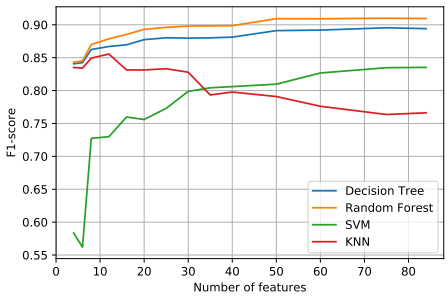


Figure A.1: F1 score after validation on a balanced dataset

Parameter	Value
criterion	*entropy*
max depth	24
number of trees	50

Table A.3: Best hyperparameters of Random Forest on a balanced dataset

Appendix B

Figures and Tables

B.1 Dataset analysis - cumulative distributions

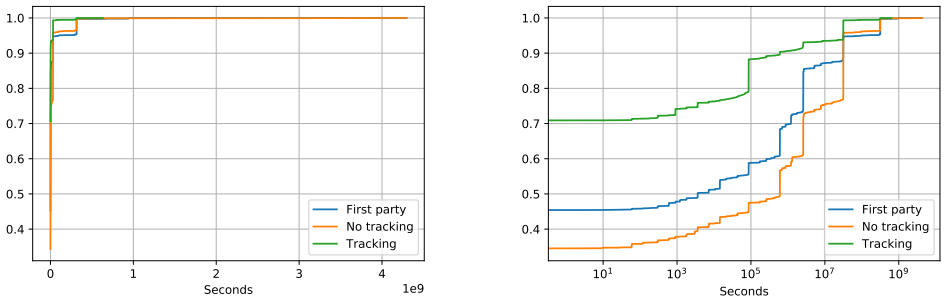


Figure B.1: Cumulative distributions of expiring age of cache value

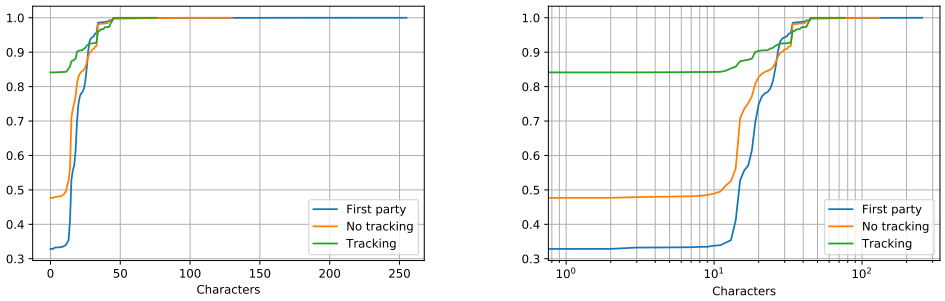


Figure B.2: Cumulative distributions of ETags length

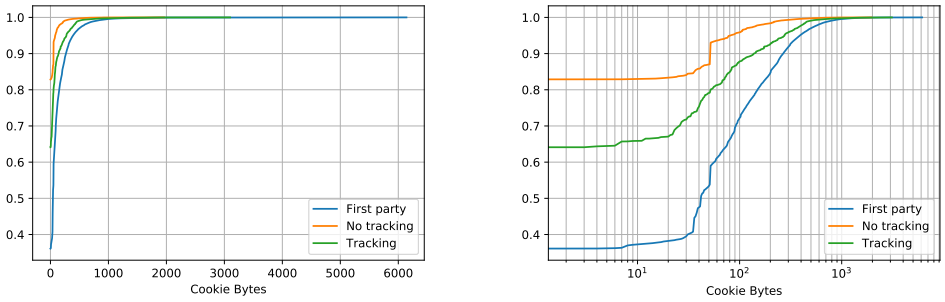


Figure B.3: Cumulative distributions of request cookies length

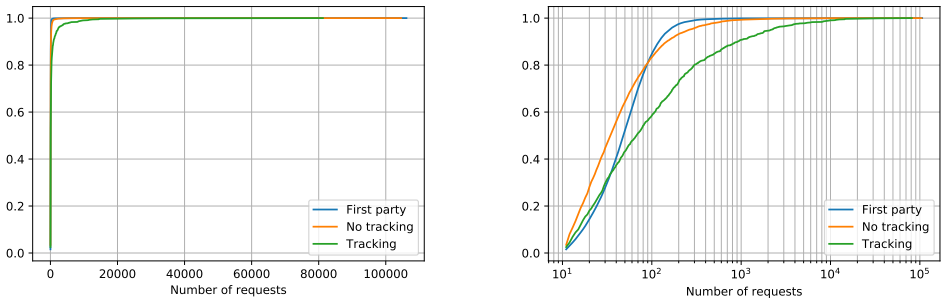


Figure B.4: Cumulative distributions of the number of requests

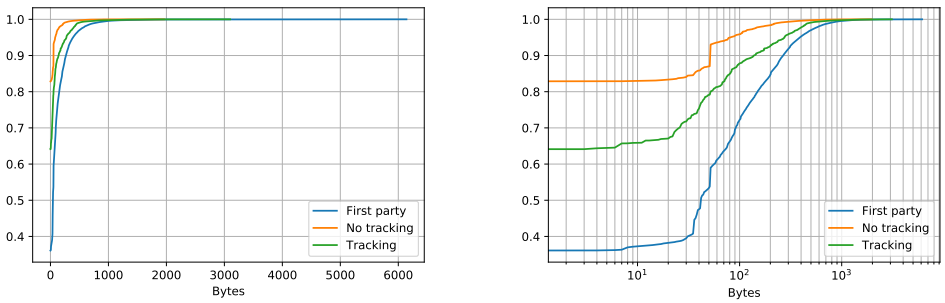


Figure B.5: Cumulative distributions of response body size

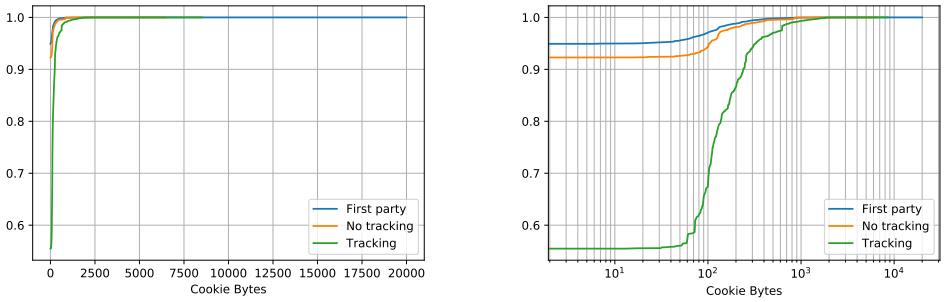


Figure B.6: Cumulative distributions of response cookies length

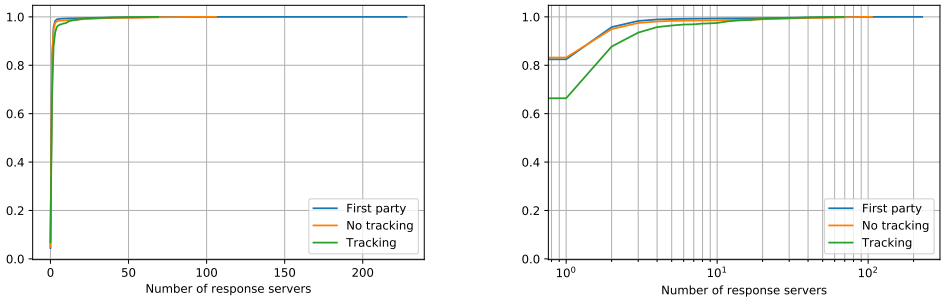


Figure B.7: Cumulative distributions of response cookies length

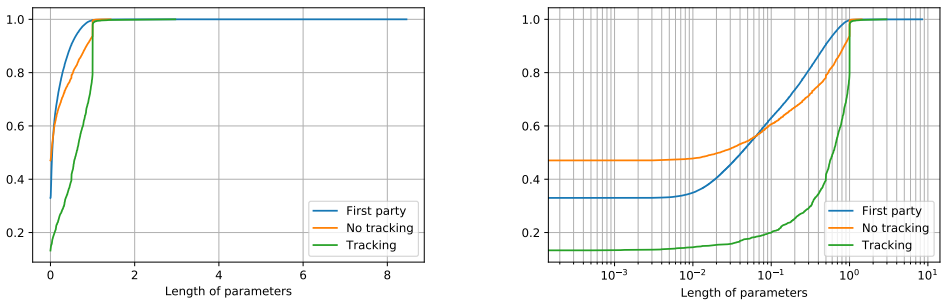


Figure B.8: Cumulative distributions of URL parameters length

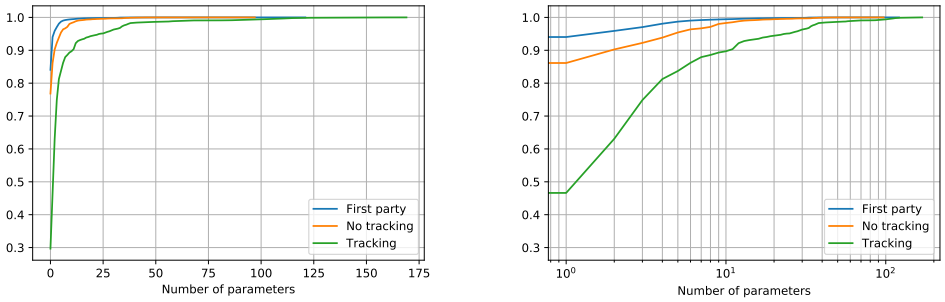


Figure B.9: Cumulative distributions of number of URL parameters

B.2 Feature selection

Features ordered by importance after applying the mRMR method, for the first party vs third party domains classification and for the trackers vs non tracking domains classification.

Trackers vs non-tracking domains

1. Referers	2. average Url Length	3. average Response Body
4. Response Servers	5. Jpg Extension	6. 25% Response Body
7. 10% Request Cookies	8. 75% ETag	9. Png Extension
10. 90% Response Cookies	11. Other Type	12. 10% Url Parameters Number
13. 90% Url Length	14. Gif Format	15. Css Extension
16. 50% Expiring Age of cache	17. std Time	18. Other Format
19. 50% Response Body	20. std Url Parameters Number	21. average Response Cookies
22. Jpg Format	23. 50% Url Length	24. 25% ETag
25. std Response Body	26. Html Type	27. Js Extension
28. Html Extension	29. 10% Response Body	30. 25% Url Parameters Number
31. average Expiring Age of cache	32. 75% Url Length	33. GET Method
34. 10% Time	35. 75% Response Body	36. 90% Url Parameters Length
37. std Request Cookies	38. 10% Url Length	39. Other Extension

40. std Url Length	41. 10% ETag	42. Css Type
43. 50% Url Parameters Number	44. Png Format	45. std Response Cookies
46. 90% Time	47. 25% Response Cookies	48. 90% Response Body
49. 75% Response Cookies	50. average ETag	51. 90% Url Parameters Number
52. Other Method	53. 25% Expiring Age of cache	54. average Request Cookies
55. 75% Url Parameters Length	56. 25% Url Parameters Length	57. Image Type
58. 50% Response Cookies	59. average Url Parameters Number	60. 50% Time
61. 90% Request Cookies	62. 10% Response Cookies	63. 25% Request Cookies
64. 10% Expiring Age of cache	65. 50% ETag	66. POST Method
67. 50% Url Parameters Length	68. 75% Request Cookies	69. std Expiring Age of cage
70. std ETag	71. 10% Url Parameters Length	72. 50% Request Cookies
73. Gif Extension	74. 75% Url Parameters Number	75. Script Type
76. 90% Expiring Age of cache	77. 90% ETag	78. 75% Time
79. 25% Url Length	80. 25% Time	81. 75% Expiring Age of cache
82. average Time	83. std Url Parameters Length	84. average Url Parameters Length

Table B.1: Features ordered by importance for the classification of domains into non-trackers and trackers

First party domains vs all other domains

85. Html Type	86. 10% Time	87. 10% Url Length
88. Response Servers	89. Other Method	90. 75% Response Body
91. std Request Cookies	92. 75% Url Parameters Length	93. 50% Response Cookies

94. 10% Url Parameters Length,	95. 75% Expiring Age of cache	96. Html Extension
97. Gif Format	98. 25% Url Parameters Length	99. std Time
100. std ETag	101. 10% Response Cookies	102. 10% Url Parameters Number
103. Referers	104. 10% Response Body	105. 25% Response Cookies
106. GET Method	107. 25% Url Parameters Number	108. 90% Response Cookies
109. 50% Url Parameters Length	110. 10% Request Cookies	111. 25% Url Length
112. 50% Time	113. 90% Url Parameters Length	114. std Response Body
115. 25% ETag	116. 75% Response Cookies	117. Png Format
118. 25% Request Cookies	119. 50% Url Parameters Number	120. average Request Cookies
121. 10% Expiring Age of cache	122. 50% Response Body	123. POST Method
124. std Url Length	125. std Response Cookies	126. 75% Time
127. Js Extension	128. std Expiring Age of cage	129. 50% UrlLen
130. 75% Url Parameters Number	131. 10% ETag	132. 90% Time
133. 25% Response Body	134. Other Format	135. Other Type
136. 25% Time	137. 90% Response Body	138. 90% Url Parameters Number
139. 50% Request Cookies	140. Jpg Extension	141. 25% Expiring Age of cache
142. 90% ETag	143. Gif Extension	144. average Response Cookies
145. average Response Body	146. 75% Url Length	147. average Time
148. Png Extension	149. 50% Expiring Age of cache	150. average ETag

151. 90% Url Length	152. Css Type	153. Jpg Format
154. 90% Expiring Age of cache	155. 75% Request Cookies	156. 50% ETag
157. average Url Length	158. Other Extension	159. average Expiring Age of cache
160. 75% ETag	161. Image Type	162. 90% Request Cookies
163. std Url Parameters Number	164. Script Type	165. Css Extension
166. std Url Parameters Length	167. average Url Parameters Number	168. average Url Parameters Length

Table B.2: Features ordered by importance for the classification of domains into firts parties and third parties

B.3 Top 50 most risky websites

website	f_1	f_2	f_3	Total risk
blogspot.com	1.000	1.000	1.000	1.000
adobedtm.com	0.784	0.868	0.789	0.814
gossiponthis.com	0.849	0.679	0.630	0.719
weknowmemes.com	0.836	0.547	0.610	0.664
tribunist.com	0.950	0.486	0.552	0.663
tweaktown.com	0.881	0.453	0.620	0.651
okbob.net	0.923	0.523	0.507	0.651
timescolonist.com	0.923	0.489	0.527	0.646
javabeat.net	0.840	0.510	0.585	0.645
eliteprospects.com	0.811	0.526	0.561	0.633
2012un-nouveau-paradigme.com	0.884	0.523	0.485	0.631
manatelugumovies.net	0.860	0.511	0.503	0.625
wikistrike.com	0.866	0.521	0.452	0.613
gta5cheats.com	0.807	0.451	0.561	0.606
elwatan.com	0.689	0.618	0.506	0.604
laughingsquid.com	0.819	0.443	0.540	0.601
backpacker.com	0.804	0.530	0.436	0.590
palingseru.com	0.706	0.569	0.495	0.590
foodiecrush.com	0.817	0.470	0.479	0.589
manaserials.com	0.783	0.543	0.438	0.588
thaimom.net	0.794	0.462	0.503	0.586

popolay.com	0.818	0.467	0.455	0.580
kshb.com	0.721	0.490	0.516	0.576
idnusa.com	0.761	0.504	0.461	0.575
clipfasthd.com	0.739	0.521	0.460	0.573
notrefamille.com	0.753	0.477	0.490	0.573
prenoms.com	0.714	0.535	0.462	0.570
nbc-2.com	0.781	0.366	0.564	0.570
corvetteforum.com	0.844	0.450	0.414	0.569
x17online.com	0.847	0.378	0.476	0.567
abc12.com	0.854	0.323	0.517	0.565
amusingplanet.com	0.817	0.441	0.435	0.564
awsubs.co	0.680	0.500	0.506	0.562
photographyblog.com	0.787	0.380	0.515	0.561
wmcactionnews5.com	0.879	0.318	0.477	0.558
hallyukstar.com	0.736	0.404	0.532	0.557
news30over.com	0.793	0.433	0.438	0.555
techeblog.com	0.749	0.443	0.464	0.552
newsok.com	0.817	0.369	0.465	0.550
totalprosports.com	0.718	0.455	0.475	0.549
toledoblade.com	0.809	0.422	0.416	0.549
lemonidol.com	0.722	0.452	0.470	0.548
belasmensagens.com.br	0.789	0.368	0.485	0.547
walb.com	0.849	0.331	0.459	0.546
superpride.com.br	0.821	0.379	0.438	0.546
gimmedelicious.com	0.651	0.484	0.502	0.546
wbtv.com	0.850	0.323	0.459	0.544
hawtcelebs.com	0.820	0.336	0.475	0.544
livingrichwithcoupons.com	0.807	0.439	0.383	0.543
blacklistednews.com	0.779	0.442	0.408	0.543

Table B.3: Top 50 most risky websites according to the estimated total risk computed in Chapter 5

Bibliography

[1] Adam Lerner et al. «Internet Jones and the Raiders of the Lost Trackers: An Archaeological Study of Web Tracking from 1996 to 2016». In: *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/lerner.

[2] *PIMCity project*. URL: https://pimcity.com/.

[3] *HTTPArchive*. URL: https://httparchive.org/.

[4] Balachander Krishnamurthy and Craig Wills. «Privacy diffusion on the web: A longitudinal perspective». In: Jan. 2009, pp. 541–550. DOI: 10.1145/1526709.1526782.

[5] Gunes Acar et al. «The Web Never Forgets: Persistent Tracking Mechanisms in the Wild». In: *Proceedings of the ACM Conference on Computer and Communications Security* (Nov. 2014), pp. 674–689. DOI: 10.1145/2660267.2660347.

[6] Steven Englehardt and Arvind Narayanan. «Online Tracking: A 1-million-site Measurement and Analysis». In: Oct. 2016, pp. 1388–1401. DOI: 10.1145/2976749.2978313.

[7] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. «Detecting and Defending Against Third-Party Tracking on the Web». In: *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. San Jose, CA: USENIX Association, Apr. 2012, pp. 155–168. ISBN: 978-931971-92-8. URL: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/roesner.

[8] *Adblock Plus*. URL: https://adblockplus.org/en/.

[9] Imane Fouad et al. «Missed by Filter Lists: Detecting Unknown Third-Party Trackers with Invisible Pixels». In: *Proceedings on Privacy Enhancing Technologies* 2020 (Apr. 2020), pp. 499–518. DOI: 10.2478/popets-2020-0038.

[10] *Disconnect trackers list*. URL: https://disconnect.me/trackerprotection.

[11] *EasyList*. URL: https://easylist.to/.

[12] *Adblock Plus filters explained.* URL: https://adblockplus.org/filter-cheatsheet.

[13] Hanchuan Peng, Fuhui Long, and C. Ding. «Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy». en. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.8 (Aug. 2005). ISSN: 0162-8828. DOI: 10.1109/TPAMI.2005.159. URL: http://ieeexplore.ieee.org/document/1453511/.

[14] *Scikit-learn documentation.* URL: https://scikit-learn.org/stable/index.html.

[15] Tobias Urban et al. *The Unwanted Sharing Economy: An Analysis of Cookie Syncing and User Transparency under GDPR.* 2018. arXiv: 1811.08660 [cs.CR].