Instituto Tecnológico y de Estudios Superiores de Monterrey

Estado de México Campus

Escuela de Ingeniería y Ciencias

Visualization and Machine Learning Techniques to Support Web Traffic Analysis

A thesis presented by

Fernando Gómez Herrera

Submitted to the
Escuela de Ingeniería y Ciencias
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

Atizapán de Zaragoza, Estado de México, Dec, 2018

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Estado de México

The committee members, hereby, certify that have read the thesis presented by Fernando Gómez Herrera and that it is fully adequate in scope and quality as a partial requirement for the degree of Master of Science in Computer Sciences.

Raúl Monroy Borja
Tecnológico de Monterrey
Principal Advisor

Eduardo Morales Manzanares
Instituto Nacional de Astrofísica, Óptica y Electrónica
Committee Member

Luis Angel Trejo Rodríguez
Tecnológico de Monterrey
Committee Member

Raúl Monroy Borja
Associate Dean of Graduate Studies
Escuela de Ingeniería y Ciencias

Atizapán de Zaragoza, Estado de México, Dec, 2018

Declaration of Authorship

I, Fernando Gómez Herrera, declare that this thesis titled, Visualization and Machine Learning Techniques to Support Web Traffic Analysis and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Fernando Gómez Herrera
Atizapán, Estado de México, Dec, 2018

Dedication

Este trabajo es fruto de mucho esfuerzo y apoyo, tanto moral como económico, de la persona que me ayudó a llegar a donde estoy: mi mamá. Estoy, y siempre estaré, agradecido por lo que me has brindado; te admiro mucho mamá, me gustaría poder ser igual de trabajador y lograr lo mismo que tú.

Acknowledgements

Agradezco infinitamente a mi pareja Aliosha por todo el apoyo incondicional en estos difíciles años de maestría. Eres una gran inspiración para mí y si pude llegar aquí en estas condiciones, fue en gran medida debido a ti. No podría haber encontrado mejor compañera en la vida; soy muy afortunado de tenerte a mi lado. Gracias, en verdad, por todo; siempre estuviste ahí. Mamá, no encuentro cómo pudiste tú terminar tus múltiples maestrías teniendo que estar cuidando a tus hijos tú sola. Estoy seguro que mi estrés, preocupaciones y dificultades no se comparan nada a lo que tuviste que superar. Estoy más que impresionado y eres a quien más admiro en la vida. Gracias por todo tu apoyo. Te quiero mucho. Estoy muy orgulloso de que seas mi mamá. También agradezco a todas las personas que estuvieron ahí para mí; compañeros, profesores, familia y amigos. Con grata mención especial a mi compañero de lucha, Rodolfo; con quien pasé muchas risas en estos dos años y de quien también aprendí mucho. Y por supuesto no podría dejar de mencionar directamente al gran amigo que a pesar de ya no coincidir, me sigue dichando con su amistad, Luis.

Por último, pero no menos importante, doy el debido reconocimiento al Tecnológico de Monterrey, mi alma máter, por haberme otorgado la oportunidad (con beca del 100%) de cursar este posgrado. Agradezco enormemente a CONACyT por el apoyo económico sin el cual, obviamente, no podría estar aquí. A los profesores del posgrado que son una inspiración para mí; Edgar Emanuel Vallejo, Miguel Medina y mención especial para Salvador Venegas. A mi asesor de tesis Raúl Monroy, por recibirme en el proyecto como su asesorado, por el apoyo en la investigación y todo el tiempo invertido en las múltiples revisiones de la tesis, además de haberme brindado una experiencia internacional. Claramente, ligado a lo anterior, agradezco a José Emanuel Ramírez Márquez por haberme recibido en Stevens y por toda la atención que me brindó en su momento.

Gracias a todas las personas que compartieron conmigo su conocimiento, tiempo y afecto en estos dos años; compañeros de clase y amigos.

Visualization and Machine Learning Techniques to Support Web Traffic Analysis

by

Fernando Gómez Herrera

Abstract

Web Analytics (WA) services are one of the main tools that marketing experts use to measure the success of an online business. Thus, it is extremely important to have tools that support WA analysis. Nevertheless, we observed that there has not been much change in how services display traffic reports. Regarding the trustworthiness of the information, *Web Analytics Services (WAS)* are facing the problem that more than half of Internet traffic is *Non-Human Traffic (NHT)* [78]. Misleading online reports and marketing budget could be wasted because of that. Some research has been done [65, 3, 71, 40, 64], yet, most of the work involves intrusive methods and do not take advantage of information provided by current WAS. In the present work, we provide tools that can help the marketing expert to get better reports, to have useful visualizations, and to ensure the trustworthiness of the traffic. First, we propose a new *Visualization Tool*. It helps to show the website performance in terms of a preferred metric and enable us to identify potential online strategies upon that. Second, we use *Machine Learning Binary Classification (BC)* and *One-Class Classification (OCC)* to get more reliable information by identifying NHT and abnormal traffic. Then, marketing analysts could contrast NHT against their current reports. Third, we show how Pattern Extraction algorithms (like PBC4cip's miner [46]) could help to conduct traffic analysis (once visitor segmentation is done), and to propose new strategies that may improve the online business. Later on, the patterns can be used in the Visualization Tool to analyze the traffic in detail. We confirmed the usefulness of the Visualization Tool by using it to analyze bot traffic we generated. NHT traffic shared a very similar *linear* navigation path, contrasted with the more complex human path. Furthermore, BC and OCC (BaggingTPMiner [50]) worked successfully in the detection of well-known bots and abnormal traffic. We achieved a ROC AUC of 0.844 and 0.982 for each approach, respectively.

List of Figures

List of Tables

Contents

Chapter 1

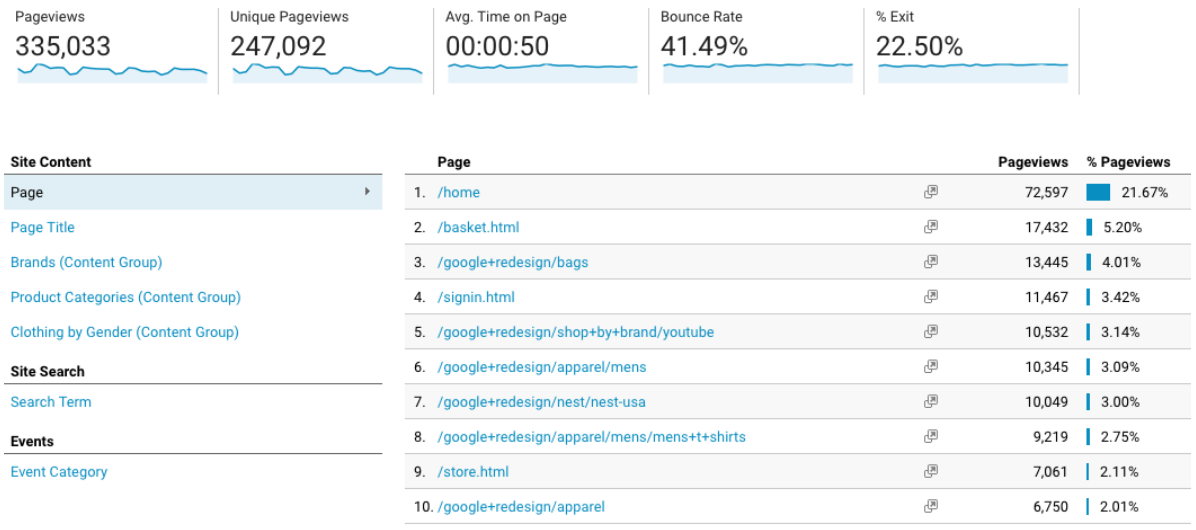
Introduction

Web Analytics services are one of the principal tools that marketing analysts use to measure the success of online businesses. Therefore, it is extremely important to have powerful tools; tools that allow to analyze and visualize the traffic in a better way, and also tools that can provide reliable information that depicts the real situation of the online business. With powerful tools, business owners may be able to make better decisions faster and distribute better the budget on online marketing resources.

We believe that the current state of products and techniques for Web Analytics analysis could be improved with the help of *Machine Learning (ML)* algorithms and new visualization techniques. This research presents some of the current problems in the Web Analytics industry where inherently marketing experts get affected. In order to support the marketing expert with those problems, we introduce a new visualization tool, a way to ensure the reliability of traffic reports by identifying *Non-Human Traffic (NHT)*, and a method of traffic analysis by using ML Pattern Extraction Algorithms.

1.1 The Need For Visualization and Analytics Tools

Data analysis is a very important task for any business analyst or marketing expert in order to get useful insights that could lead to decision making. Visualization tools come very handy for such tasks and therefore is very important to count with some of them in the everyday toolbox. In the context of Web Analytics, we observed that there has not been much change in how services display traffic reports.



Pageviews	Unique Pageviews	Avg. Time on Page	Bounce Rate	% Exit
335,033	247,092	00:00:50	41.49%	22.50%

Site Content

- Page
- Page Title
- Brands (Content Group)
- Product Categories (Content Group)
- Clothing by Gender (Content Group)

Site Search

- Search Term

Events

- Event Category

	Page	Pageviews	% Pageviews
1.	/home	72,597	21.67%
2.	/basket.html	17,432	5.20%
3.	/google+redesign/bags	13,445	4.01%
4.	/signin.html	11,467	3.42%
5.	/google+redesign/shop+by+brand/youtube	10,532	3.14%
6.	/google+redesign/apparel/mens	10,345	3.09%
7.	/google+redesign/nest/nest-usa	10,049	3.00%
8.	/google+redesign/apparel/mens/mens+t+shirts	9,219	2.75%
9.	/store.html	7,061	2.11%
10.	/google+redesign/apparel	6,750	2.01%

Figure 1.1: Website page views. It shows the Google Analytics report for the Google Online Store. The table on the right-hand side, is the common way to report the number of views for the website pages. On the top-side, it displays overall results for the whole website.

Figure 1.1 is an example of the standard report for visit metrics. Such data do not provide, for example, how visits interact with the website pages individually; more often, results are aggregated metrics spread in multiple reports. In the case of visitor's navigation, the common report is a table of sequential actions (very similar to Figure 1.1). However, there is the case where marketers would need to explore individual traces of navigation from their users; such option is not available in some of the current platforms or if exists, the way to represent such information is through simple text reports.

To improve current tools, we introduce a new visualization tool that could help analysts to get insights from data. We propose a graph-based visualization that allows combining multiple metric reports (i.e. page views, visits, bounce rate...) into a single one. It also displays how visits interact with website pages, and displays a metric associated for each one.

In order to validate the visualization tool, we consulted a Mexican e-commerce business which gave us positive feedback indicating the tool has good potential, most of all, in the analysis of individual visit navigation path.

1.2 A problem with Internet Bot Traffic

The Internet is no longer dominated in web browsing by human people, instead, bots are taking control of many interactions within websites. Around 52% of web traffic reported in latest Incapsula paper is coming from bots [78]. Such visits to websites by bots are denominated *Non-Human Traffic (NHT)*. NHT is then categorized in two: *General Invalid Traffic (GIVT) and Sophisticated Invalid Traffic (SIVT)*. Although a great percentage of GIVT has the purpose to provide a service (for example web indexers —GoogleBot, or feed fetchers), SIVT becomes a problem when the intention of such bots are malicious.

One specific problem related to SIVT arises in the field of Web Analytics. It would be very easy for a malicious attacker to launch a series of bots targeting a specific online marketing campaign. Then, the marketing expert, not being aware of that, may think that a certain product or campaign is having a marvelous success. In such case, time and money of the business could be wasted.

Taking as an example the field of online advertising, which is measured by Web Analytics tools, nowadays, it is a must for companies to invest in advertising campaigns with the purpose of increasing sells, get more customers, and expand their presence in the market. Usually, companies that provide such ad services charge a specific amount for every click made in an ad (named *CPC, Cost per Click*) or for every thousand of impressions (*CPM*) —ads displayed and viewed by a user. So, what happens when there are bots, not humans, that are clicking or viewing (actually, not even viewing!) such ads? Companies lose money. comScore reports a lose between $650 million and $4.7 billion in the U.S. [13], in addition, *WhiteOps/ANA (Association of National Advertisers)* report estimated losses of more than $7 billions of dollars [7]. NHT is responsible for such waste of money, more specifically SIVT, because 80% of NHT is sophisticated [48]. The *Media Rating Council (MRC)* is the organization that dictates standards for digital metrics and providers must complain to those standards. Since NHT is a current problem, MRC recently updated their *Invalid Traffic (IVT)* detection guidelines addressing NHT [52] but not many of the big providers like Google, Youtube nor Facebook are up to such guidelines. Because of that, organizations like Havas group (O2, Royal Mail, BBC...) and many others [49, 74] are stopping giving money to those providers until they provide reliable measures.

Given the previously stated situation, marketing experts need more reliability in the reports given by Web Analytics tools. There are some ways to fight against that problem and this work provides a way to support in there.

1.3 Working towards a better visualization tool and SIVT detection

We started and continued this research project under the hypothesis that *if we use Machine Learning and better visualization tools, then we could support better decision making for the marketing experts or business analysts*. In order to do so, we defined three main objectives.

First, the development of a new Visualization Tool which could be able to help by showing the website performance in terms of a preferred metric and enable the identification of potential online strategies upon that.

Second, use Machine Learning Classification and identify Non-Human Traffic (or at least abnormal traffic) to provide more reliable traffic reports. By doing so, we attempt to ensure the trustworthiness of the information and obtain cleaner traffic reports.

Third, support the task of visitor segmentation analysis by using Pattern Extraction algorithms to identify common patterns in such segments. Consequently, insights from those patterns could lead to propose new strategies that may improve the online business.

This written thesis attempts to accomplish the previously described objectives and it is structured as follows. First, we are going to explain how we collected web traffic (Chapter 2) and use it as a base for experimentation. Second, in Chapter 3, we introduce the new visualization tool to explore web traffic in more detail, providing new tools that could improve the way marketing experts perform traffic analysis. Third, in Chapter 4, a bot traffic detection system is created with two different Machine Learning approaches; two-class classification, and one-class classification. Fourth, Chapter 5 presets how pattern mining from the dataset can be used to analyze the website traffic, leading us to potential online strategies that could improve the conversion rate. Finally, we conclude in Chapter 6 with some of the key insights of this research and propose future work that could be worth to explore.

Chapter 2

Data Collection and Exploration

Data collection usually is the first step in the whole Machine Learning *pipeline* and our case was not the exception. Since we are going to analyze and visualize web analytics, gathering such data from scratch would have been very challenging and it may have taken too much time; luckily, there are tools that provide the basic functionality for doing so and we took advantage from them.

In this chapter, we are going to explain how we collected website traffic data by mounting sites on servers and using tracking scripts (Section 2.1); then, we are going to look into such data by performing an *Exploratory Data Analysis (EDA)* (Section 2.2), describe the attributes and some of their distributions, and decide which ones will be useful for further analysis. We will relate the information found with human and bot behavior and show visualizations of how the visits are distributed around the world. Finally, we will draw some conclusions (Section 2.3) from the data and decide how it is going to be used for the Machine Learning process.

2.1 Experimentation

2.1.1 Server and websites

Three websites were mounted on a single server which we owned at the moment. The server was located at Mexico City, Mexico, and all incoming traffic to it were collected. In the following section, Section 2.1.2, we will explain how it was done.

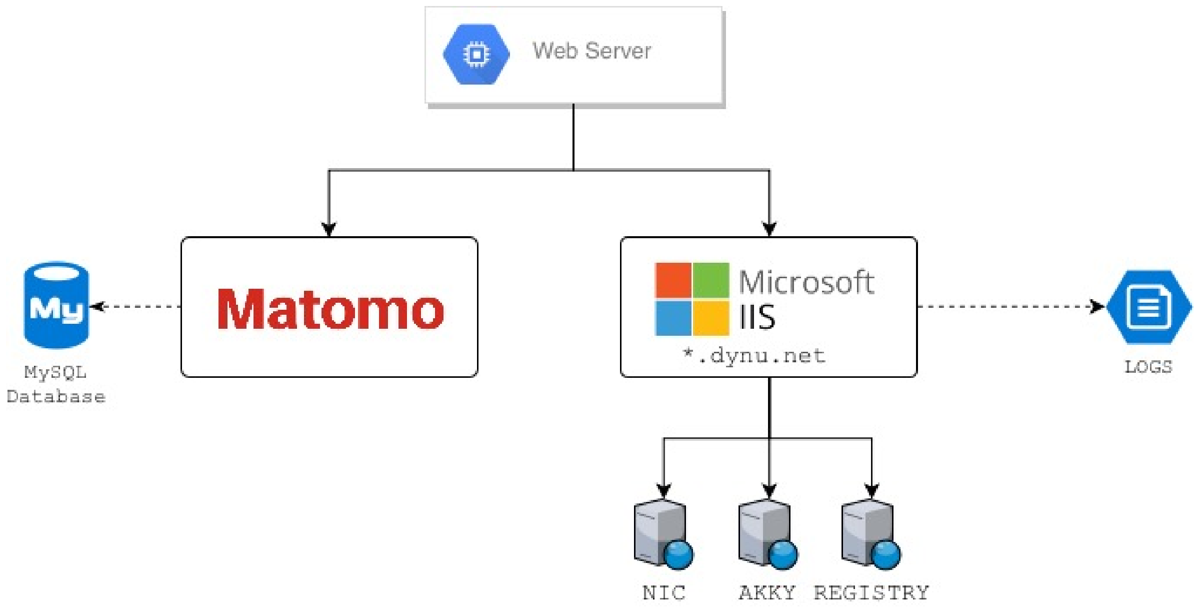


Figure 2.1: Server architecture and client interaction. Three websites mounted on the single server, plus Matomo's server where traffic logs were sent.

Figure 2.1 shows the architecture of how the websites were mounted on the server. By using a free DNS server named DynDNS, we mapped each website to unique *Unified* Resource Locations (URLs), which were the entry point for the users browsing the site.

2.1.2 Tracking scripts

It is quite common in websites to include several scripts which serve to specific purposes. Web analytics tools often provide an easy way to *install* their services by incorporating such scripts into website pages. Google Analytics and Matomo are examples of that. We took advantage of these existing platforms to collect the visitor's information and thus get the data we desired. Listing 1 shows an example of how the scripts look like; every web page had inserted the Matomo's script.

```
<!-- Matomo -->
<script type="text/javascript">
  var _paq = _paq || [];
  _paq.push(['trackPageView']);
  _paq.push(['enableLinkTracking']);
  (function() {
    var u="//{$PIWIK_URL}/";
    _paq.push(['setTrackerUrl', u+'piwik.php']);
    _paq.push(['setSiteId', {$IDSITE}]);
    var d=document, g=d.createElement('script');
    var s=d.getElementsByTagName('script')[0];
    g.type='text/javascript'; g.async=true;
    g.defer=true; g.src=u+'piwik.js'; s.parentNode.insertBefore(g,s);
  })();
</script>
<!-- End Matomo Code -->
```

Listing 1: Matomo's JavaScript tracking code. It must be included in every single page wanted for collecting traffic. Usually it gets inserted at the end of the *body* tag or inside the *head*.

We chose Matomo as the platform in charge to collect the data because it is open-source, but most of all, because it gives 100% data ownership, whereas other platforms do not allow that (e.g. Google Analytics); meaning that we were able to setup the platform in our own server (as seen in Figure 2.1 and having complete control over it. If we used Google Analytics, we would be constrained by the fact they do not allow you to download all the visits' /raw/ information. Once the scripts were installed, every time a user visited the website, Matomo collected information from the browser, device information, IP address, country of origin, language and many other information. In Section 2.2.1 we look further into what information is collected.

2.1.3 Traffic generation

Because one of our objectives was to detect *Non-Human Traffic (NHT)* we decided to run certain bot programs to generate *fake* traffic. We shall clarify that by saying *fake traffic* does not means that there was no real traffic at all, the websites indeed received requests but *automated programs* were the ones visiting, clicking and browsing through the pages. Three main bot traffic generators were used: *Traffic Spirit, Jingling Traffic Bot and w3af*; and for the human traffic we requested the participation of people from Mexico City

(Mexico) and New Jersey (USA). Table 2.1 shows how we conducted the experiments. The first week we ran the bot traffic generators, and the rest of the weeks people visited the website; bot traffic were stopped at such moment.

Traffic Type	**Start Date time**	**End Date time**
Jingling Traffic Bot	June 06, 2017 04:16:39 CDT	June 06, 2017 05:10:22 CDT
Traffic Spirit	June 06, 2017 05:11:12 CDT	June 06, 2017 05:57:07 CDT
Traffic Spirit	June 07, 2017 05:21:50 EDT	June 07, 2017 09:33:11 EDT
Traffic Spirit	June 12, 2017 05:10:22 EDT	June 12, 2017 13:20:02 EDT
w3af	June 13, 2017 05:30:43 EDT	June 13, 2017 09:48:54 EDT
Jingling Traffic Bot	June 14, 2017 05:05:16 EDT	June 14, 2017 10:04:31 EDT
Human Traffic	June 16, 2017	July 06, 2017

Table 2.1: Traffic generation dates. First, we generated bot traffic June 6 to June 14, 2017; then, we asked the participation of people to generate human traffic from June 16 to July 6, 2017. Start and End date columns indicate the date and time in which traffic started to be generated.

We are going to analyze the results of this experiment in the following section.

2.2 Exploratory Data Analysis

2.2.1 Dataset General Information

The constructed dataset has a total of *1,809 observations* and 68 attributes; 35 are numerical attributes and 33 categorical. Each observation represents a *visit to a website* and has information like the browser the user used to visit the site, geographical location information in most of the cases (user's continent, country, city, etc.), in some cases the user's device information (device brand e.g. Apple, Samsung, etc.) and also aggregated information of the visit like how many pages the user viewed (*pageviews*) and even custom information we gathered like the amount of *clicks and keypresses* the used did during the visit.

Bot traffic dates from June 6, 2017 to June 14, 2017, whereas human traffic dates from June 16, 2017 to July 3, 2017. 96% of the data belongs to bot traffic and the remaining to human traffic (4%); we can see that we are against a class-imbalance problem.

Detailed information of the dataset is described in Table 2.2. Attributes with a $\star$ indicate that new *statistical* attributes were generated from them, specifically the *mean*, *median* and *a total sum*.

Attribute	**Description**
actions	Number of actions in the visit. It contains both events and pageviews
browserCode	Coded version of browserName. Examples: CH, FF, IE (for Chrome, Firefox and Internet Explorer, respectively)
browserFamily	Rendering engine. Example: Trident, Gecko, Blink
browserName	Name of the visitor web browser e.g. Chrome, Firefox, Internet Explorer, etc.
browserVersion	Web browser application's version. e.g. 10, 8
city	Name of the identified visitor's city. e.g. Sydney, Sao Paolo, Rome
clicks $\star$	# of clicks made during the whole visit
continentCode	Visitor's continent. e.g. eur, asi, amc, afr
countryCode	Example: de, us, fr
cpu_cores	# of CPU Cores of the visitor's device. This value cannot always be obtained. e.g. 2, 4, 8
daysSince FirstVisit	Days past since the first time the user visited the site
daysSince LastVisit	Days past since the last time the user visited the site
deviceBrand	Visitor's device brand. e.g. 3Q, Acer, Airness, Alcatel, Altech UEC, Arnova, Amazon, Apple, Archos, Asus, Avvio
deviceModel	Model of the visitor's device
deviceType	Type of visitor's device. e.g. desktop, smartphone, tablet, feature phone, console, tv, car browser, smart display, camera, portable media player, phablet
events	Number of events triggered in the visit
fingerprint	A 32-bit integer representing the browser's fingerprint (more details at [67]). A fingerprint is information collected about a remote computing device for the purpose of identification. Fingerprints can fully or partially identify individual users or devices even when cookies are turned off. e.g. 1167364286

firstAction Timestamp	Timestamp of the first action in a user's visit
generationTime⋆	Time the page took to render in seconds (Matomo uses seconds, but IIS uses milliseconds). Because this value is per pageview, we compute summarized data for it.
keys⋆	# of keys pressed during the whole visit
languageCode	Language of the visitor's browser. e.g. de, fr, en-gb, es
lastAction Timestamp	Timestamp of the last action in a user's visit
latitude	Geo coordinate
longitude	Geo coordinate
operatingSystem Code	Coded version of the visitor's device operating system. e.g. WIN, MAC, LIN (Windows, macOS, Linux, respectively)
operatingSystem Version	Operating system version. e.g. XP, 7, 2.3, 5.1, ...
pageviews	# of page views i.e. how many pages the visitor viewed
plugin_cookie	Boolean attribute that indicates if the browser has enabled cookies
plugin_director	Boolean attribute that indicates if the browser has enabled the Director plugin
plugin_flash	Boolean attribute that indicates if the browser has enabled the Flash Player plugin
plugin_java	Boolean attribute that indicates if the browser has enabled the Java plugin
plugin_list	Idem
plugin_pdf	Idem
plugin_silverlight	Idem
plugins	# of the previous plugin attributes enabled
referrerName	Name of the Website where the visitor comes from
referrerSearch EngineUrl	URL of the search engine provider that the user used to reach the website
referrerType	Referrer type (where the visit comes from) Example: direct, search, website, campaign
referrerUrl	URL of the website where the user comes from
regionCode	Example: 01, P8, 02
resolution	Screen resolution of the user. Example: 1280x1024, 800x600

screen_info	An electronic visual display for computers. The screen monitor comprises the display device, circuitry and an enclosure.
serverTimestamp	Timestamp of the visit (unix epoch in seconds, not ms)
timeSpent⋆	Amount of time the user spent in the page in seconds
visitCount	Count of visits for this visitor
visitDuration	Visit duration in seconds
visitIp	Visitor's public IP
visitLocalHour	Local hour related to the visitor's country
visitLocalTime	Local time of the visitor. This field is more descriptive than the serverDate
visitServerHour	Server local hour (where the website is mounted on)
visitorId	ID for identifying unique visitors, i.e. the same ID can have multiple page views
visitorType	Describes if the user has previously visited the website. Can take the values *new* or *returning*.
class	Class label, either *bot* or *human*.

Table 2.2: Dataset information. Description of each attribute.

2.2.2 Redundant Attributes

The original dataset contained several attributes that were redundant, most of them were in the form of *attrCode, attrName*; for example, *browserCode and browserName*. Both attributes represent the same; **Name* attributes just contain detailed info. For example, to represent the Google Chrome's browser, the attributes have the following values: *browserCode = CH*, *browserName = Google Chrome*.

We removed the attributes that ended in *Name* and left only the *coded* ones (e.g. *browserCode*). That decision was taken because the **Code* version use less *memory space*.

2.2.3 Variable correlation

Before going deeper into some of the attributes we have, we performed a correlation analysis of the numerical variables. We have 35 numerical attributes, however, some of them are statistical attributes derived from others (the ones with a $\star$), so we are going to use only those we consider are the most meaningful.

Variable correlation is useful because it can let us better understand the relationships between two variables. It is performed as a statistical method and it is represented as a numerical value between -1.0 to 1.0. The correlation is positive if the value is greater than 0, and negative otherwise.

If the correlation is positive, it means that both of the variables change in the same direction, and if it is negative, their values change in opposite ways. For example, if we are predicting house prices, it would be very probable that the price will be positively correlated to the size of the house, i.e. the price increases as the size increases.

Figure 2.2 shows the variable correlation analysis for the dataset. It is a matrix that intersects an attribute in a row to another in a column. Each intersection indicates the correlation coefficient, which belongs to $[-1.0, 1.0]$. It is said that a variable is highly correlated to another if its correlation coefficient is close to 1.0; and conversely, a pair of variables is *negatively correlated* if the value gets close to -1.0; we can say that there is no correlation if the value gets close to 0.0.

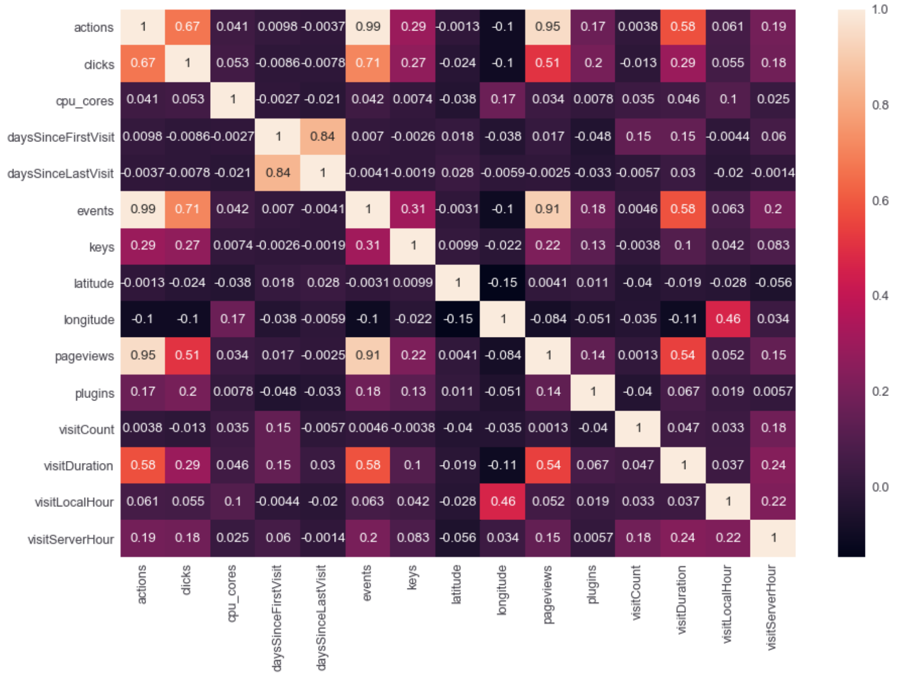


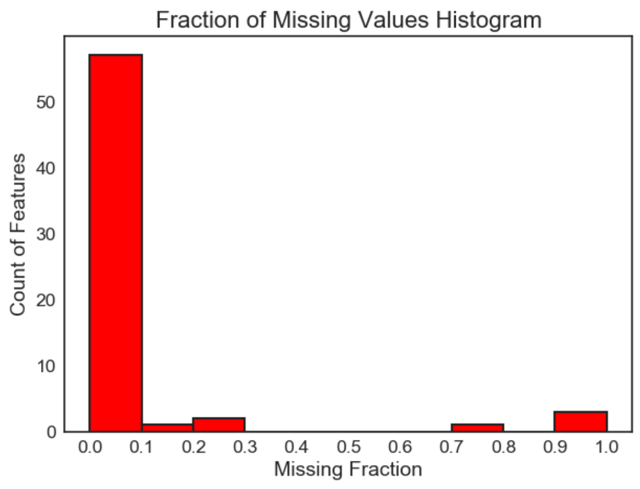
Figure 2.2: Heatmap Matrix of the variable correlation using Pearson's method. A value close to 1.0 means a positive correlation, whereas a value close to -1.0 is a negative correlation; values close to 0.0 indicates no correlation.

As we expected, there are some highly positive correlated variables but there are other correlations we did not expect. We expected to see a high correlation between *clicks, keys, pageviews and actions/events* because the first ones are derived attributes from the latter, and so it happens. However, we not expected a lower correlation coefficient between *clicks* and *pageviews*, and they have a value of 0.51. Also, we expected to see a higher value for the *visit duration* and the *pageviews*, however, it only has a correlation coefficient of 0.54. In this case it may indicate that users *possibly spent more time reading or viewing the pages, rather than browsing a lot.*

Because of the high correlation between the variables of *actions/events* with *clicks/pageviews/keys*, we decided not to use them for the Machine Learning process, i.e. we are going to use only the derived attributes of *clicks*, *pageviews* and *keys*, instead of *actions* and *events*. Such variables we are keeping provide an improved contextual representation of the visit; *pageviews, keys and clicks* gives us a better understanding of what the user was doing, e.g. it is more convenient to have the following values: *pageviews* = 10, *clicks* = 4, *keys* = 2 rather than actions = 10 and *events* = 16 because then we know that there were 10 page views during the visit, and also the user made 4 clicks and pressed a total of 2 keyboard keys.

2.2.4 Missing Values

With the purpose of cleaning the dataset and leaving only those attributes that provide enough information, we analyzed the fraction of *missing values* for each attribute. The results are shown in Figure 2.3. We can see that only few attributes have a high percentage of missing values. We dropped out those with a $fraction \geq 0.75$. Such attributes are: *deviceModel, referrerName, referrerSearchEngineUrl* and *referrerUrl*. We decided to use such *threshold* because it was the recommended value by the toolset we used to pre-process the data. Moreover, even if we had to choose a more strict value (> 0.3), we would end up with the same attributes removed.



(a) Histogram of missing fraction

Attribute	MF
RSE URL	0.993919
deviceModel	0.993367
referrerName	0.988944
referrerUrl	0.788834
OS Version	0.254837
city	0.200663

(b) Attributes and their missing fraction (*MF*) value (second column). RSE: Referrer Search Engine. OS: Operating System.

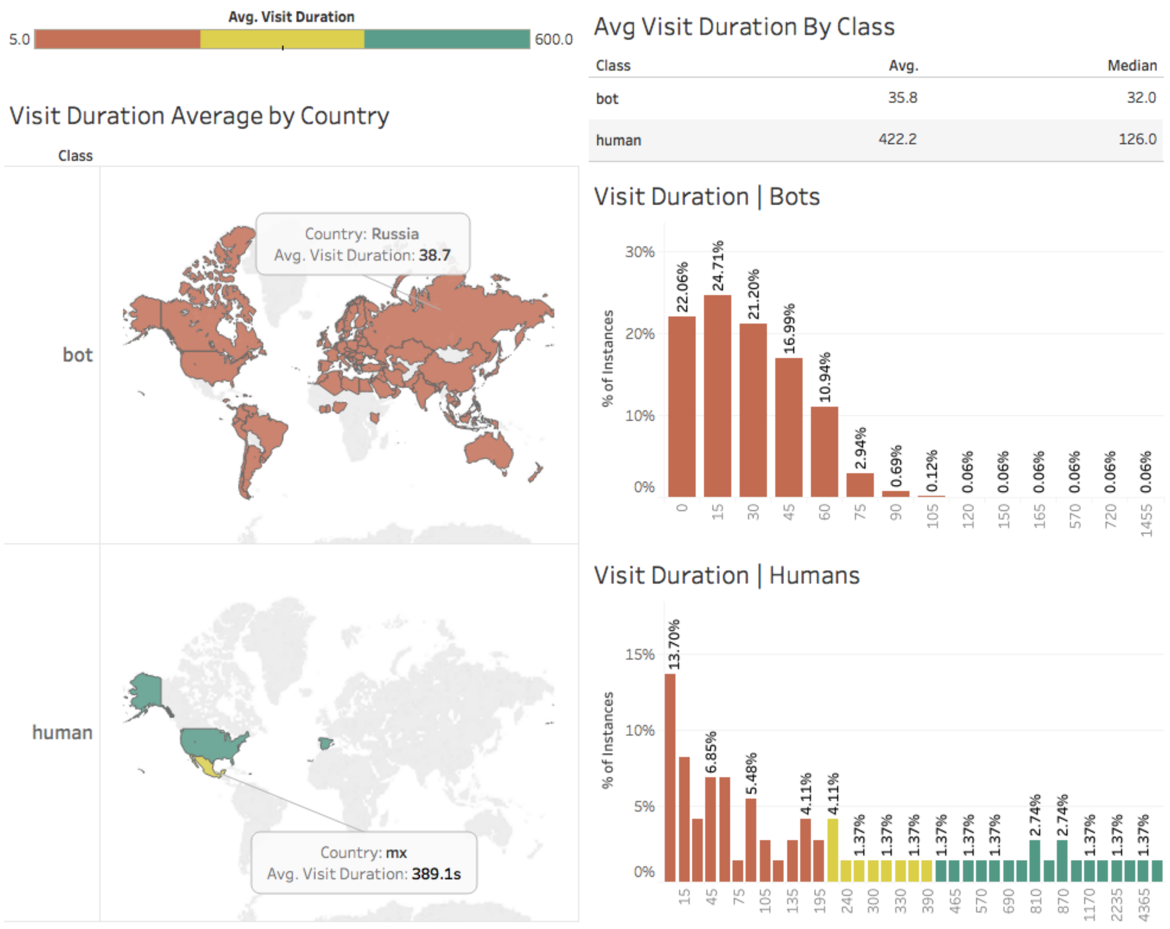
Figure 2.3: Fraction of Missing Values. On the histogram of the left-hand side, the x-axis represents the fraction of missing values; the y-axis counts how many features fall into that bucket. On the right-hand side, the table shows how many values are missing for each attribute as a percentage.

Surprisingly, the meaning that the *referrerUrl* presents a high percentage of missing values indicates that many of the visits were *direct visits*. For the human traffic we predicted that, because it was given a direct URL to the participants in the experiments. But, for the bot traffic we did not know exactly what would be the behavior, even when we launched directly the bots. It seems that the bots we used, visited directly the website without any intermediary. Is this good? It depends. From the point of view of advertising, *it is very good* because it means that bots are not reaching the website through search engines and so, they are not wasting the spend on ads. However, from the point of view of detecting sources of invalid traffic it is a bad, because we can not build a blacklist of referrers.

2.2.5 Individual Attribute Inspection

In the following subsections, we present a more detailed exploration of some individual attributes. We explore only the most common attributes presented by Web Analytics services, such as the *visitor's duration*, the number of *page views*, the visit's *geographical location* and the web *browser's usage*. We included two more attributes that are not commonly reported by Web Analytics platforms: the number of clicks and keys the visitor pressed during the visit, with the hope that they provided useful information to further analysis.

2.2.6 Visit Duration



Avg Visit Duration By Class

Class	Avg.	Median
bot	35.8	32.0
human	422.2	126.0

Figure 2.4: Visit Duration Information. Left-hand side: Colored World Heatmap by class, with ranks depending on the visit duration. Right-hand side: Histograms of the visit duration by class.

In Figure 2.4 we can see a summary of the visit duration attribute. It shows two World maps, that contain information about the bot and human classes, respectively. The countries are colored as a heatmap, where the country with lowest visit duration is colored as red, the ones with a high value are colored as green and yellow as an intermediate value. As an example, we show the average visit duration of Mexican visitors: 389.1 seconds, and the same value for a bot visitor country (Russia): 38.7 seconds.

On the right hand side we can also see two histograms, each one depicting the visit duration attribute by class. The x-axis represents the different buckets of the histogram (in steps of 15 seconds) and the y-axis represents the percentage of instances that fills that bucket.

Figure 2.4 shows information we expected: almost every country from the bot class is colored in red and for human visitors, the countries are colored in yellow-green. This means that if we split the visitors in three main buckets, let say Low-Middle-High, many of the bots visitors would be placed in the Low bucket and the humans in the Middle-High buckets, *which would mean that the bots we used can be characterized to spend low time visiting the website, where as humans are the opposite*.

Regarding to the histograms, we can see that many of the humans fall into the first interval $0 - 15$ but its distribution is more sparse than the bots histogram. Moreover, bots are browsing the website at most 75-90 seconds; bins further than such values contain very few instances (0.66%).

2.2.7 Visitors Clicks

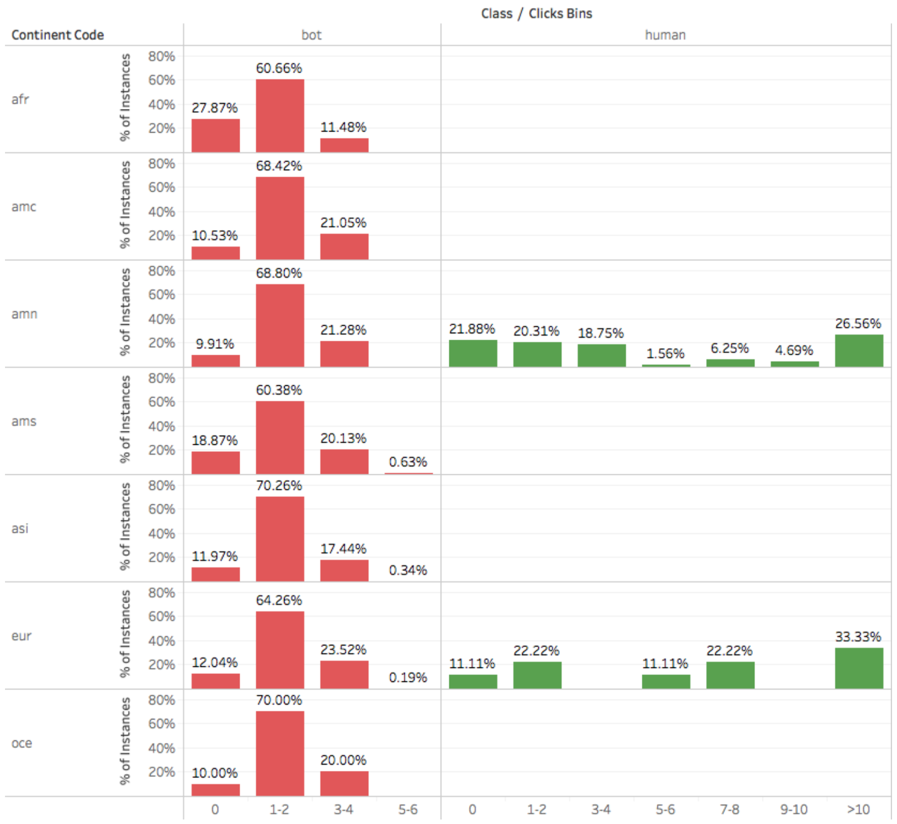


Figure 2.5: Histograms of clicks by continent and class. The left-hand side shows information for bots and the right-hand side for humans. Each row represents the continent code, and for each continent there is a histogram by class. X-axis represents the amount of clicks made by a visit. Y-axis shows the percentage of instances that fall into a specific bucket.

Figure 2.5 shows a distribution of the clicks variable. It displays a histogram for each continent and for each class, i.e. we analyze the visitors from America and generate two histograms: each one belonging to a different class (human and bot). We repeat such visualization for each of the continents. This process is because in marketing analysis it is common to analyze customer segments and not as a hole. One example of segmentation is by demographics.

We can find some similarity in the behavior of the bots. By looking at Figure 2.5 we can see that the click distribution is very similar for each continent: 60-70% of the time they click 1-2 times, around 20% of the time clicks 3-4 times and finally around 10% click zero times. The distribution seems to be equal for every continent. In contrast, the human click distribution seems to be more sparse.

2.2.8 Operating System and Browser Usage

We are interested in some characteristics of the visitors like the Operating System they are using, for example to know which are the most vulnerable to be infected by bots; the same applies to web browsers. Fortunately, we have this related information. Figure 2.6 shows the Operating System and Web Browser that the visitors were using during their visit to the websites.

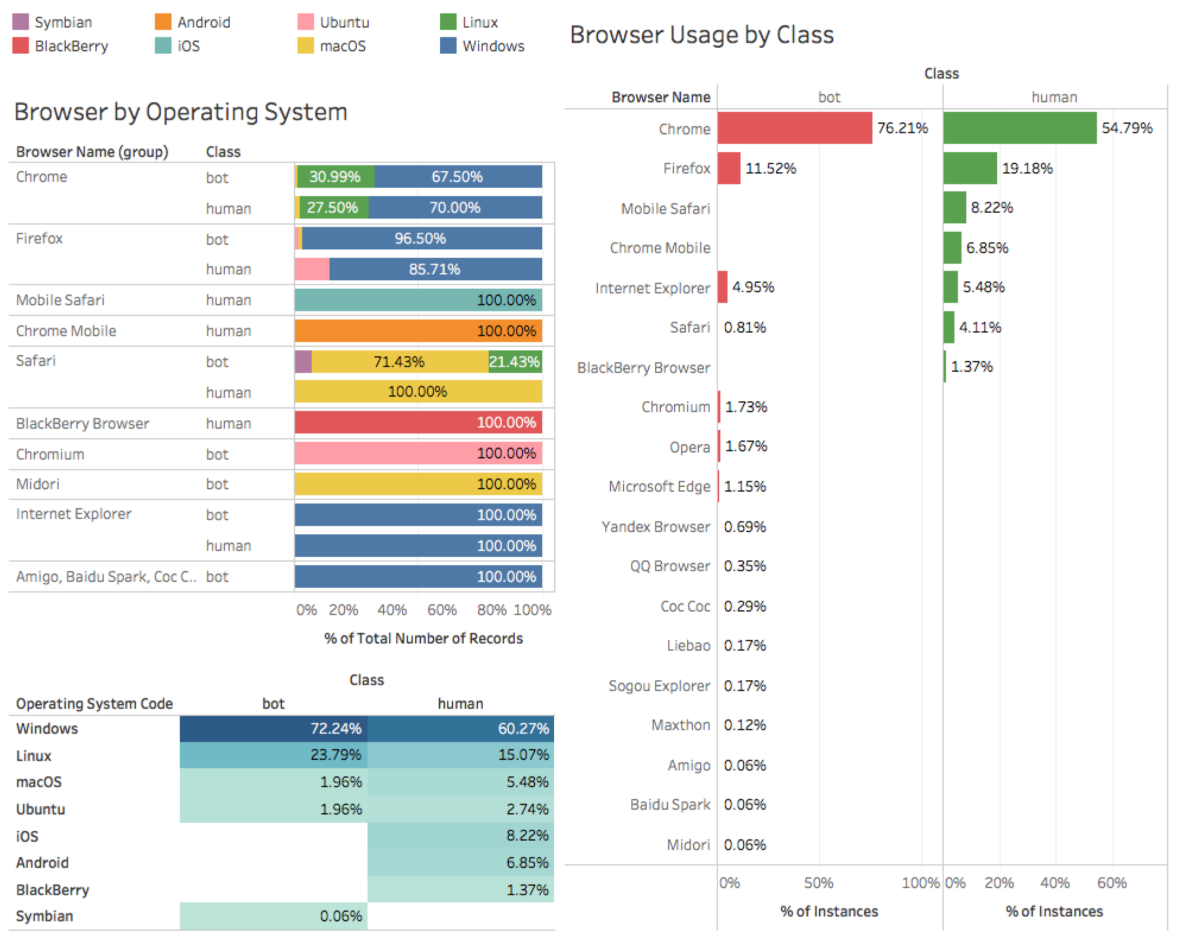


Figure 2.6: Operating System and Web Browser Usage. Bar charts on the right-hand side show the distribution of web browsers separated by class to see which are the most common for humans and bots. On the upper-left side we see information about the browser and the operating system it was used. On the bottom-left corner we see the percentage of OS usage by class.

It is easy to spot that *Google Chrome* is the most used Web Browser and *Windows* the most used Operating System, both for the two classes. However, we can see other useful information. For example, all the visits registered for mobile browsers belong to the human class. Could this mean that mobile devices are less probable to be infected by web bots?

By analyzing the information of the *Browser by Operating System* chart we can get more insight in how important the OS is for the browser and the class it belongs. For example, in most of the cases, for the browser when the visit is classified as bot, Windows was the operating system the user was using. Regarding to the *popularity of the browser*, no human traffic belongs to an unpopular browser (Chrome, Firefox, Safari, Opera). This could lead us to implement a business rule to keep an eye on uncommon web browsers.

2.3 Conclusions

We can extract a few ideas from the previous analysis, some of them as a hypothesis and others as facts. We shall remember that all of this information is related to this specific scenario and may not represent the whole universe, nevertheless, it is a good starting point and still provides useful insights to marketing people. Let us enumerate some conclusions from the previous sections:

1. We are not going to use the attributes of *events* and *actions* for the Machine Learning process because they are highly correlated to the derived attributes of *clicks,keys* and *pageviews*. Such three attributes provide us a better contextual explanation and therefore are better to use.

2. We are going to drop the attributes *deviceModel*, *referrerName*, *referrerSearchEngineUrl* and *referrerUrl*, because they have a fraction of missing values ≥ 0.75.

3. The analysis of missing values gave us information related to the referrers. It seems that most of the traffic were direct visits, i.e. they did not reach the website through an intermediary (search engine, social media). This behavior was expected for human traffic and unknown for bot traffic. This is good for advertising, because the spend on ads is not wasted in search engines by invalid traffic and also good for human traffic because it indicates a kind of loyalty to the brand by remembering the site URL or at least entering directly to it.

4. Visit duration and pageviews were not highly correlated as we expected. It might indicate that users *possibly spent more time reading or viewing the pages, rather than browsing a lot.*

5. Bots tend to spend less time than humans. The median value of visit duration for bots was 32 seconds versus 126 seconds of humans.

6. Bots click behavior seems to be shared among their visits; the distribution of its clicks among the different visits from the global continents look very similar. In contrast, human click distribution seems to be more sparse.

7. Google Chrome and Windows were the most used by the users for visiting the websites.

8. Browsers running under Windows tends to be more present in the bot class.

We have collected the information and explored it, now, we are going to use it for:

1. Create a visualization tool for exploring and query visit information.

2. Take different approaches for building Machine Learning classifiers and detect Non-Human Traffic (NHT).

3. Get insights from the data by using Machine Learning Pattern Extraction algorithms and use patterns to query existing traffic or create custom visitor segments.

2.4 Acknowledgements

Later on during this research, NIC México, the Mexican company which supported part of this project, shared with us a new dataset which contributed to this research. We thank the company for giving us access to their information. Such data did not contain any confidential information about their clients and some attributes were encrypted or omitted in order to protect the privacy of both NIC and its clients.

Chapter 3

Visualization Model

Getting insights from data is a very important skill for any business analyst, marketer or any decision maker. However, it is almost impossible to get knowledge looking directly at raw data, so it must be presented in a way that is easy to understand and analyze. There are many options to visualize data; specifically, for web analytics, companies have developed platforms for doing such tasks, and by the use of geographic maps, pivot tables, heatmaps and other visualization techniques, finding valuable information becomes easier. Nevertheless, the trend of creating new ways to display web analytics has not been changing much in the past few years; goal reports, conversions and site performance usually are still displayed as tables, big score counters or line plots. Such data do not provide, for example, how visits interact with the website pages individually; most often, results are summarized and spread in multiple reports. In the case of visitor's navigation, the common report is a table of sequential actions or as aggregated traffic flow (funnels). However, there is the case where marketers would need to explore individual traces of navigation from their users; such option is not available in some of the current platforms or if it exists, the way to represent such information is through simple text reports. We believe this could be improved; our proposal addresses the problem by using a visual representation inspired in network diagrams.

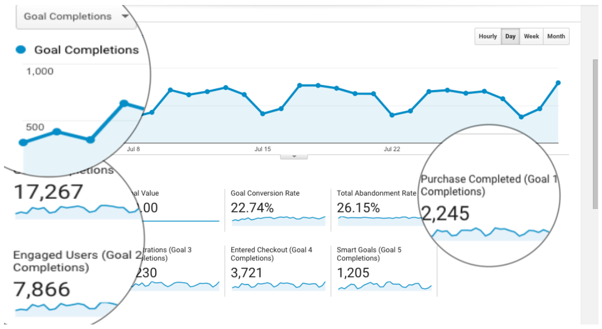
Most of the work done in this area has been developed by private companies and offered as a paid service; CrazyEgg, Kiss Metrics, Mixpanel, Imperva or comScore's software are just a few examples of them, and of course, the public Google Analytics. As the Internet is a global network, with traffic coming from any part of the world, it is very common to see new visualizations inspired in world maps or network graphs. [5, 39, 44, 17, 29, 77, 54, 11] are some examples.

Although the previous work was used to analyze network traffic, many of them uses either an intrusive client-approach or a low-level packet capturing method, in order to create a visualization. We aimed for a strategy closer to Web Analytics that is the use of server logs and browsing information, generated directly from a client's request. In this way we do not have to install software in each client's computer, we just need a single HTTP server to gather multiple client's information.

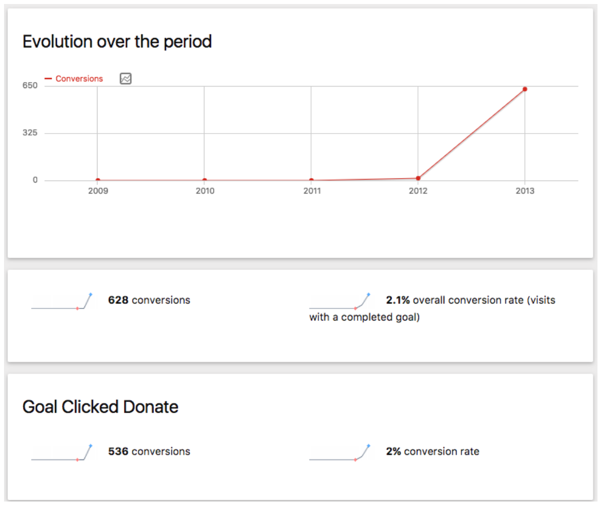
The latter was a motivation for us to propose a new visualization tool that could help analysts to get insights from data and to support other reporting tools. We propose a graph-based visualization that allows to combine multiple metric reports (i.e. page views, visits, bounce rate...) into a single one, it also displays how visits interact with the pages of a website and a metric associated to each one. Regarding to visit details, we believe to have a better visual representation of a navigation path which enabled us to find visual patterns of human and bot traffic, whereas using table reports we would not be able to find them that easily. This was also possible thanks to a query system embedded into the tool, where Machine Learning Patterns can be introduced to segment website visits. Furthermore, we implemented the previous ideas to multiple sites at once, providing a visualization that shows how internal traffic flows and also where external traffic is coming from. This is a new feature that, as far as we know, no one has implemented in their tools.

The contributions we aimed, enable marketing people to explore in more detail web site visits, it allows them to combine multiple reports into a single visualization, and also it can be used to assess an ad-campaign performance or to audit reports provided by advertisers and compare how reliable the information is, contrasted with the one reported by our tool. We had very positive feedback from the marketing team we consulted for validation and they see this work as a powerful resource.

3.1 Problem description



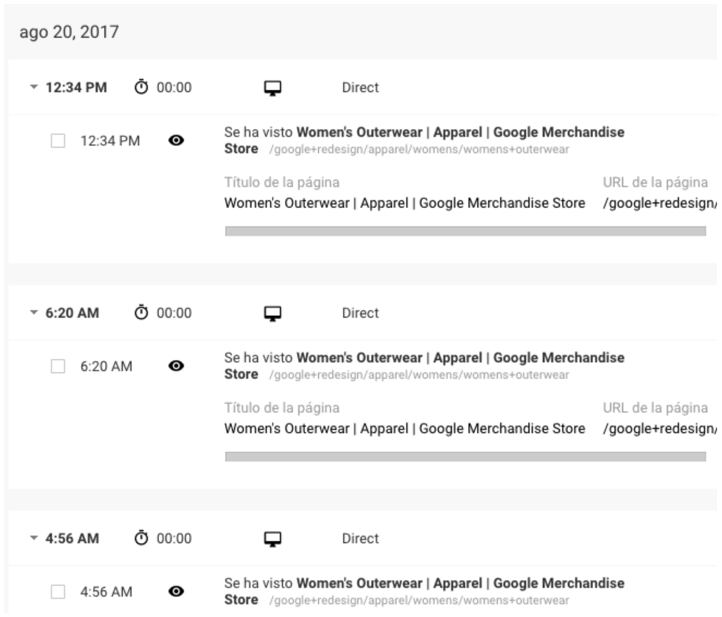
(a) Google Analytics. Example of a Day view report for some defined goals on the Google Online Store.



(b) Matomo. Example of a Year view report for some defined goals on an example website provided by the Matomo Platform.

Figure 3.1: Goal reports by Google Analytics (left) and Matomo (right). These are a common kind of report to display the performance of *user pre-defined goals* such as *number of sign ups, purchases, etc.* through time.

As described before, some of the web analytics reports have not changed in the past few years. For example, Figure 3.1 shows a common report to display goal performance and conversion counters. Although these kinds of reports provide a quick way to compare *time series*, there could be ways to improve them. For example, such report does not provide information about how visitors interact with the website pages; the information is usually just presented as summarized results spread in multiple reports.


ago 20, 2017

12:34 PM 00:00 Direct

12:34 PM Se ha visto **Women's Outerwear | Apparel | Google Merchandise Store** /google+redesign/apparel/womens/womens+outerwear

Título de la página: Women's Outerwear | Apparel | Google Merchandise Store
URL de la página: /google+redesign/

6:20 AM 00:00 Direct

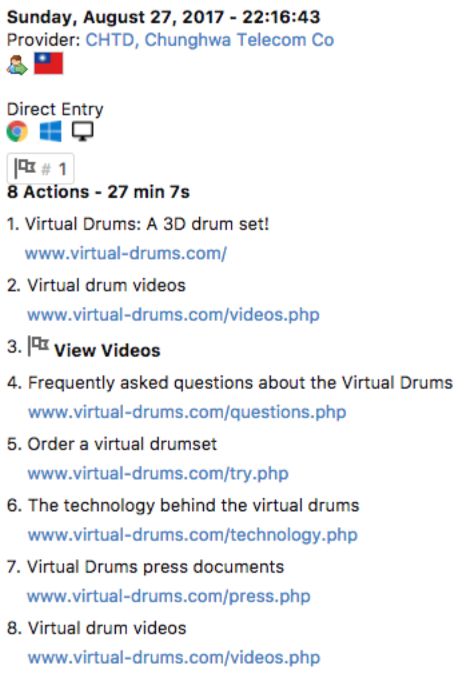
6:20 AM Se ha visto **Women's Outerwear | Apparel | Google Merchandise Store** /google+redesign/apparel/womens/womens+outerwear

Título de la página: Women's Outerwear | Apparel | Google Merchandise Store
URL de la página: /google+redesign/

4:56 AM 00:00 Direct

4:56 AM Se ha visto **Women's Outerwear | Apparel | Google Merchandise Store** /google+redesign/apparel/womens/womens+outerwear

(a) Google Analytics: user's browsing history


Sunday, August 27, 2017 - 22:16:43
Provider: CHTD, Chunghwa Telecom Co

Direct Entry

1
8 Actions - 27 min 7s

1. Virtual Drums: A 3D drum set!
   www.virtual-drums.com/
2. Virtual drum videos
   www.virtual-drums.com/videos.php
3. **View Videos**
4. Frequently asked questions about the Virtual Drums
   www.virtual-drums.com/questions.php
5. Order a virtual drumset
   www.virtual-drums.com/try.php
6. The technology behind the virtual drums
   www.virtual-drums.com/technology.php
7. Virtual Drums press documents
   www.virtual-drums.com/press.php
8. Virtual drum videos
   www.virtual-drums.com/videos.php

(b) Piwik: user's browsing history

Figure 3.2: Google Analytics' (left) and Piwik's (right) Click Path view . Both are tables of the sequence in which a single visitor was browsing the site.

On the other hand, for reporting visitor's navigation, the common report is represented as tables of sequential actions (as shown in Figure 3.2) or as *aggregated* traffic flow by the use of *funnels*. However, there is the case where marketers would need to explore individual traces of navigation from their users; such option is not available in some of the current platforms or if it is, the way of representing such information is through simple text reports as we can see on Figure 3.2.

Regarding customer segmentation, few platforms have implemented features for doing it automatically. The common way to do it is by an expert manually creating filters based on arbitrary parameters such as the *visitor's country, user type, visitor's language, etc.*. Although this does not represent any problem, there is a huge chance to *mislead the segment*; we believe that Machine Learning could improve this process by *suggesting such filtering parameters* through the use of Pattern Mining [23, 26, 30]. Such patterns represent *true segments* found from the data itself.

Having described the previous situation, this research proposes new ways to display website traffic by using an interactive tool that provides several ways to arrange visits, conversions, user behavior, *click path*, page views and filtering options, which combined with Machine Learning Pattern Recognition [23] (specifically, with the pattern extractor of PBC4Cip [46]) could help to find clusters for new market niches, discover unknown visitor segments or improve segment analysis by the use of the *patterns* that can be found on the data. A pattern contains a series of conditions that are true for a certain amount of data instances. "*country = Mexico* $\wedge$ *pageviews* $>$ *10*" is an example of a pattern. We have interest in patterns that cover as much data as possible, because web traffic can be segmented by them. A complete chapter of patterns and its application is discussed in Chapter 5.

We integrated into the visualization tool a way to introduce a pattern and using it as a query to filter the visits. This allows the expert to create segments automatically (after introducing the pattern) or at least give some insight on which group of visitors shares common properties or behavior. As an example, we use the tool to visualize *Non-Human Traffic (NHT)* present in the data.

In the case of *click path navigation*, we believe current visualizations can be improved. We will describe our proposal in Section 3.3, it addresses the problem by introducing a new visual representation of click path navigation inspired in network diagrams [51].

The motivation of developing this visualization tool is to help analysts to get insights from both the site traffic and the performance of the online marketing goals. Then, better business decisions like improve page content or move ads to certain pages can be taken with the support of the tool.

3.2 Related work

There are many tools available for measuring digital content and because this is such an important topic in the marketing industry, some companies have created their own tracking tools and build a business around it. We will briefly analyze such tools dividing them into three categories: *enterprise solutions, open source tools, and research proposals*. Many of the enterprise solutions (with the exception of Google products) are paid services, whereas the open source tools are free to use; we did not find implementations as a product of any of the research proposals we looked into. Furthermore, we see the problem discussed in the previous section, of non-evolving text reports without explicit visitor-page interaction, (Section 3.1) present in every one of the mentioned tools.

3.2.1 Enterprise solutions

Google Services. As the biggest company in the Internet, Google has three main products focused on web advertising (Google AdWords [27] and DoubleClick [1]) and web analytics (Google Analytics). Combining those products, you can have powerful insights of the users that are visiting your website. DoubleClick is a paid service, but AdWords and Analytics are not.

Google Analytics (GA) is the main product for getting reports and analyze the traffic to a website. It can be configured to import and track ad campaigns from Ad Words and Double Click. It allows to segment the traffic from many sources and apply several kinds of filters to build them. Also, it has the advantage of being widely known by marketing experts and people from other domain areas, so it has become a standard in a certain way.

comScore [19]. It is an American company founded in Virginia which now has a big presence not only on the Internet but also providing services to other kinds of media companies like the TV industry, newspapers, health care and others as well. Nevertheless, we could not get a further analysis of their tools because they are paid services and usually oriented to big sized companies with a large amount of data. Despite this fact, comScore has been very open with their current research and has been publishing some reports in a periodic way [20, 13].

Kiss Metrics [41]. It is also a paid service but the company behind it provides a more tailored experience, focusing on consulting and teaching their customers on how to implement the tool and interpret the results. Popular stories about the use of Kiss Metrics are the one of Lucid Chart [2], a tool for creating digital diagrams, which after upgrading their product, needed to measure the performance of the new design and by using Kiss Metrics, they got an increase of 30% in conversions. Another case is the one of the e-commerce site Manillo, an Amazon-like service from Denmark, which increased their *Return on Investment (ROI)* by 50% by understanding better their audience [3].

The disadvantage of the previous tools is that they can only be used as a hosted service. For some companies that could be problematic if they need to obey certain law regulations about storing customer's data, or for companies that need an on-premise solution. Another big concern about these kinds of services is that, usually, the user does not own the data, instead, the only way to access the information is through a third-party. To avoid the mentioned issues,

[1]`https://www.doubleclickbygoogle.com/`

[2]`https://www.kissmetrics.com/lucidchart/`

[3]`https://www.kissmetrics.com/manillo/`

the tool we present can be installed on any computer, either a web server or the business analyst's desktop.

3.2.2 Open Source tools

Matomo [1]. Formerly named Piwik, it is one of the most popular and robust tools that could be either *self-hosted* or as *Software as a Service (SaaS)* in the cloud. Matomo is a company (named the same as their main product) that focuses on giving their users a complete control of everything, meaning that the user gets full reports (no data sampling, in contrast with GA) and also is the owner of the 100% of the data; it is also completely open source, which means that can be customized as needed. Matomo is developed in PHP and also provides an HTTP API for consulting several kinds of reports such as the visitor's information, goals, and pages performance, user segments, live visits information, etc.

For the sake of this work, in addition to *raw server logs*, *we used this tool to gather and manage web traffic data in our experiments*. This decision was taken because it would be very time-consuming (and out of the scope of this research) implementing from scratch the scripts for the visitor data collection. Thus, we used the on-premise version of Matomo and installed it on a custom web server. Then, we activated the *tracking script* the tool provides to start collecting the visitors' information.

Open Web Analytics (OWA) [60]. Although the OWA project has not published any new version since 2014, it is still popular in legacy websites. It was integrated into former versions of Content Management Systems (CMS) like Wordpress or Media Wiki. It can be tested only by installing it on a personal server. One of the greatest features included *out-of-the-box* is the *click heatmap* that shows the *hottest* sections of a website page; then, it can be used to optimize the placement of page information.

3.2.3 Research and others

As the Internet is a global network with traffic coming from any part of the world, it is very common to see new visualizations inspired in world maps or network graphs. For example, Akamai [5] provides an interactive map of web attacks in real time. Kaspersky offers a similar tool but using a 3D perspective and has a little bit more features integrated [39]. Logstalgia [44] is another interesting tool to visualize HTTP server logs, it is inspired by *Atari's pong game* and when you get requests, it renders a swarm of pong balls.

In the field of credit fraud, a new method is being employed: the use of graph-based databases. Neo4j[4] and IBM Graph[5] are examples of tools for such purposes. The motivation is to find cycles inside graphs, which commonly represents a kind of fraud [47, 55].

Neural Networks can also be used as visualization tools; Self Organized Maps (SOM) has been employed to find web traffic attacks [8].

[17] surveys a couple of visualization tools developed at the User Interface Research Group at Xerox PARC. The mentioned work used such tools to improve web usability, find and predict browsing patterns and to show web structure and its evolution. Like us, such work also implemented a graph inspired visualization.

Another graph inspired visualization is Hviz [29], which was used successfully to explore and summarize HTTP requests in order to find common malware like Zeus [34] and also as a tool for forensic analysis [21]. Hviz deserves a mention for its versatile use cases and also for creating a heuristic to aggregate HTTP requests by using Frequent Item Mining. Hviz is related to ReSurf [77], using it as a benchmark to improve browsing reconstruction. Another work in this field is ClickMiner [54], which also reconstructs browsing paths and provides a tool to visualize it; this is analogous to the *Click Path* feature we propose, but the context is different: they analyze traffic from a single machine, client by client, whereas ours is server-based and we don't need access to each individual computer.

NetGrok [11] uses a combination of graphs and Treemaps to display bandwidth usage from IP hosts in real-time. Although they used the tool successfully to detect anomalies, the scope of their analysis does not match with ours; they use low-level packet capturing, whereas we use server logs, more close to the Web Analytics resources available.

3.3 Description of the Visualization Tool

Inspired on the motivation described before, we created a visualization tool to analyze web traffic. It is intended for helping analysts to get insights from both the site traffic and the performance of the online marketing goals. Then,

[4]`https://neo4j.com/`

[5]`https://www.ibm.com/us-en/marketplace/graph`

as we will see, certain marketing strategies can be proposed to improve page content, take advantage of the most visited pages like moving ads to them or adding *appealing* elements.

The tool itself has the intention to provide the following useful features:

S.1 Explore the pages, visits and goals of a website, plus their metrics (e.g. page views, bounce rate, etc.).

S.2 Explore the relationship of user visits and the pages they visit by a metric (e.g. visit duration, page views, etc.).

S.3 Combine **S.1** and **S.2** in a single report.

S.4 Explore individual visits and the *navigation path* taken by a user to a *objective page*.

S.5 Combine Machine Learning Pattern Extraction techniques to highlight visit patterns and find traffic segments, which could belong, for example, to *bot traffic* or new market niches.

S.6 By using **S.5**, help audit reports provided by advertisers and compare how reliable is the information.

Let us describe each specification...

S.3. Combine S.1 and S.2 in a single report.

Items **S.1** and **S.2** are common features found in any web analytics platform, more often, those reports are provided as tables or line plots (as shown in Figure 3.1). However, by using a different visualization we can combine a couple of those into a single one (**S.3**). A contribution of this work is the new visualization model, which we explain below.

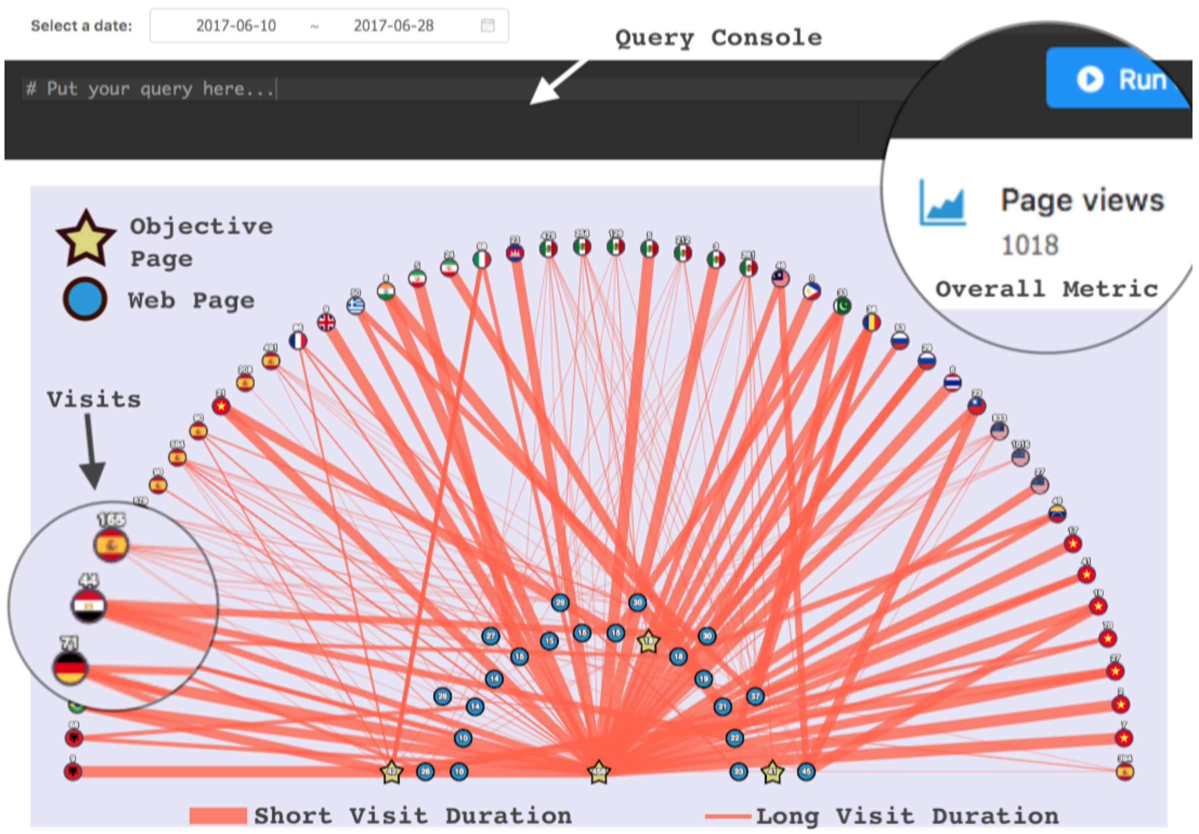


Figure 3.3: Visits View. Blue nodes represent web pages; green stars are objective pages; nodes with country flags are visits of users from that country. *Page views* was used as metric and inverse *visit duration* for the edge width. Thin connections represent longer visit duration; thicker otherwise.

Figure 3.3 shows the visualization model; it is based on several *concentric circles* and reports a specific period of time.

Visit nodes. *Outermost concentric semi-circle*; the ones containing country flags. Each node represents a *visit* to the website. The flag indicates from which country the visit is coming. A *connection* between a visit and a page node, indicates that such page was visited. The *width* of the *connection* represents a metric, in this case, the inverse of the *visit duration*; meaning that the width is thicker if the visit has a low visit duration, and a thinner width otherwise. On top of the node, the metric used for the *connection width* is displayed; *e.g.* it indicates how many seconds the visit lasted.

Page nodes. *Several concentric semi-circles of blue nodes.* Each node represents a unique web page of the website. For each inner semi-circle, nodes are ordered in ascendant from left to right by the metric used, in this case, *page views*; *i.e.* pages with fewer page views are positioned at the leftmost side and increasingly positioned to the right. In the center of the node, the metric is displayed; e.g. it indicates how many *page views* the page has.

Node distribution. Each concentric semi-circle is ordered into *k-levels*, in this case *3 levels*. Each level is ordered from the center to the outermost level in ascending order by the metric used (page views); i.e. pages on the 1^{th} *level* (closest radius) are going to have the *fewer page views*, whereas pages on the k^{st} *level* have the *higher page views*.

The selection of the *k-levels* can reflect a certain *Key Performance Indicator (KPI)*, for example, the business could decide to use $k = 5$ levels and the goal KPI could be that at least, all the objective pages should be at the fourth level; a different situation could indicate a bad performance of the business goals.

Once the parameter k is established, the elements are distributed into the k levels in the following way. *k-segments* of size $\frac{max(metric)}{k}$ are selected, starting from $min(metric)$. Then, we put all the pages depending on their specific metric into the respective level. The index page is excluded from all computations because its position is fixed into the center of the view.

For example, in the pictures we show, we established $k = 3$ and metric as page views. Therefore $max(pageviews) = 45$, $min(pageviews) = 10$, $SegmentSize = 45/3 = 15$. Thus, we end-up with three segments (starting at 10): $[min(pageviews), 25], (25, 41], (41, max(pageviews)]$.

The strategy of distribution can, of course, be changed. We used it as a simple example. An alternative could be dividing into quartiles.

Objective nodes. *Green stars.* Web pages that are considered goals of the business, e.g. *sign-up pages, landing pages, checkout pages, etc.* The same *positioning rules* as the *page nodes* are applied.

The contribution of the visualization in Figure 3.3 shows how, by using a single report, we can observe *how many views is getting each page, including the performance of goal pages, plus the visit duration and from which country the visit comes from.*

For the sake of this research, the use case we have given to the tool was to visualize *Non-Human Traffic (NHT)* and how it impacts the overall metrics of a website. As mentioned before, this is important for many consumers of analytic tools because bot traffic makes companies waste their investments in advertising. For example, comScore reports a lose between \$650 million and \$4.7 billion in the U.S. [13], in addition, *WhiteOps/ANA (Association of National Advertisers)* reports estimated losses of more than \$7 billion of dollars

[7]. NHT is responsible for such waste of money, more specifically *Sophisticated Invalid Traffic (SIVT)*, because 80% of NHT is sophisticated [48].

S.4. Explore individual visits and the *navigation path* taken by a user to a *objective page*.

Regarding the *user's navigation path*, in many platforms it is commonly represented as tables of sequential actions or as *aggregated* traffic flow (Figure 3.2). We believe this could be improved and we propose using a visual representation inspired in network diagrams.

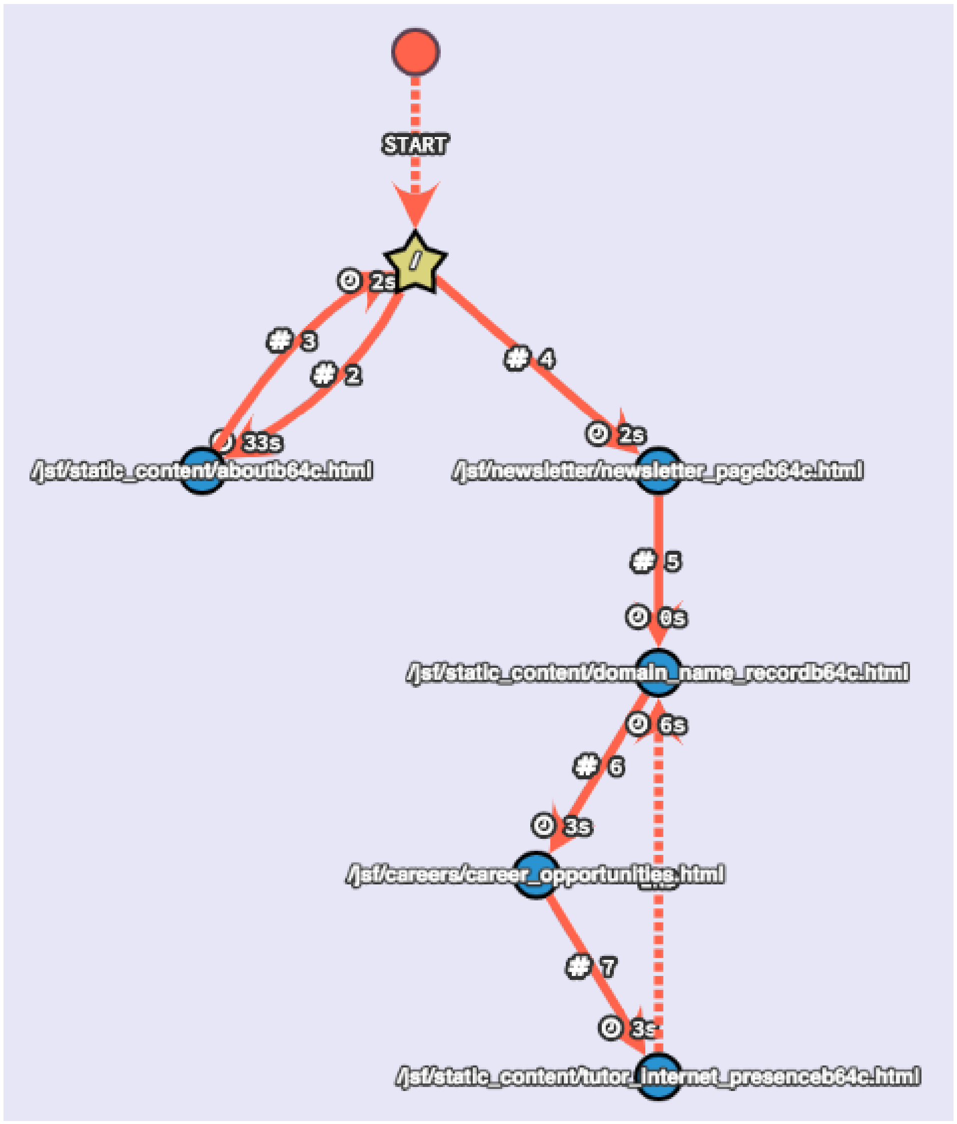

START
/
2s
# 3
# 2
33s
/jsf/static_content/aboutb64c.html
# 4
2s
/jsf/newsletter/newsletter_pageb64c.html
# 5
0s
/jsf/static_content/domain_name_recordb64c.html
6s
# 6
3s
/jsf/careers/career_opportunities.html
# 7
3s
/jsf/static_content/tutor_internet_presenceb64c.html

Figure 3.4: Click Path. Navigation graph of a visit. The number at the edge's center indicates the view step, and the one with a clock icon shows how long the previous page was viewed.

Figure 3.4 shows the proposed visualization for the navigation path. It includes a series of *page nodes* connected one another; we use *node and page* interchangeably. Each node represents a page the user visited. In the center of each node, the *Universal Resource Identifier (URI)* is shown. Each *connection* has *two items* of information; at the center of the *link* a counter shows the number in which the page was visited, i.e. the order of the page view. The second element is at the end of the connection and shows the *time spent* by the user in the previous page. If any of the pages is an *objective page*, then it is shown as a *star node*.

Contrasted with Figure 3.2, our visualization gives quickly understanding of *non-linear navigation*, like the presence of recurrent pages viewed, identify in which point of the visit the user landed to a certain objective page, or if in fact, the user never touched an objective.

S.5 Combine Machine Learning Pattern Extraction techniques to highlight visit patterns and find traffic segments, which could belong, for example, to *bot traffic* or new market niches.

S.6 By using S.5, help audit reports provided by advertisers and compare how reliable is the information.

One of the first tasks for analyzing an audience is to create segments, which is no more than performing a filter process given a series of conditions: e.g. *young people* (`18 < age < 24`); men living in New York City (`gender = male AND location = NYC`); people coming from social networks (`referrer IN [Facebook, Twitter, Snapchat]`), etc. So, it is very important to have an easy way to perform such filters and because of that, there is a *Query Console* incorporated in the tool (Item **S.5**, Figure 3.8). The *query console* interfaces with another *open-source* resource called *Cytoscape* [25], which is responsible for the actual query expression evaluation.

The following are examples of queries that can be performed using the tool, supporting the base logic operators: $\wedge, \vee, \neg$ *(and, or, not)*, *relational operators* $(=, >, <, \leq, \geq)$*, string matching* and others. The patterns follow a very similar syntax than the *Disjunctive Normal Form* [9]. The full capabilities are provided by Cytoscape and the full specification can be found in the documentation [6]. The aim of this section is to provide examples of how to use such query system on our advantage to create segments.

[6]`http://js.cytoscape.org/#selectors/data`

3.3.1 Query format

Depending on how data is structured and the available attributes, it is possible to perform queries matching such attributes. We are using data coming from Matomo's so we are able to query visits and pages based on their database structure[7]. The format follows the next grammar:

$$group[attr\ OP\ value]$$

Where: $group \in \{node, edge, .className\}$

- $className$ represents a custom class assigned to a particular data. In our case we have three classes: `visit, page, objective`. So instead of using *node or edge*, we use such classes which are less abstract. For example, `.page[attr OP value]` will target only web page attributes, similarly `.visit[attr OP value]` will query for the visit information.

- $attr$ means the attribute used for filtering. Such attributes should correspond to the context represented by the *group*. So, for example, for the *visits* group, you could have attributes like `duration, country, browser, events, etc.`; on other hand, in the case of the *page* group you could have attributes like `url, visits, pageviews, bounceRate, exitRate, avgTimeSpent, etc.`. A complete list of attributes is shown in Section 2.2.1.

- OP represents any binary operator. Depending on the *data type*, certain operators can be used over the others. The available operators are: `=, !=, >, <, >=, <=, *=, ^=, $=`. The last three operators are used for string comparison.

- $value$ is the value used for matching $attr$ by the selected operator (OP). Depending on the *data type* of *attr* you would put strings, numbers or booleans.

Logical operators

To implement the logical operators $\wedge, \vee$ you have to follow the next format:

[7] `https://developer.piwik.org/guides/persistence-and-the-mysql-backend`

$\wedge$ (**AND**):

$$group[attr_1\ OP_1\ value_1]group_2[attr_2\ OP_2\ value_2]\ \ldots\ group_n[attr_n\ OP_n\ value_n]$$

Notice the join of groups after the square brackets.

$\vee$ (**OR**):

$$group[attr_1\ OP\ value_1], group_2[attr_2\ OP\ value_2]\ \ldots\ group_n[attr_n\ OP\ value_n]$$

Notice the use of the *comma (,)* for concatenating conditions.

Examples

- **Simple query with the .visit *className***

  Visitors from the United States:

  ```
  .visit[countryCode = "us"]
  ```

- **Using the AND operator**

  Visitors using Google Chrome *and* with a visit duration greater than 10 seconds *and* coming South America.

  ```
  .visit[browserCode='CH'][visitDuration>10][continentCode='ams']
  ```

- **Using the OR operator**

  Visitors coming from Facebook or Instagram:

  ```
  .visit[referrerName='Facebook'],.visit[referrerName='Instagram']
  ```

- **Using both operators AND, OR**

  Visitors from Mexico or USA that have visited the site more than once.

  ```
  .visit[visitCount>1].visit[countryCode='us'],.visit[countryCode='mx']
  ```

The previous querying system enables us to use it with Machine Learning algorithms, specifically with decision trees-based techniques that create patterns splitting the data by attribute values (Item **S.5**). More use cases of patterns that can be used in the query system are discussed in a complete chapter, Chapter 5.

S.7 Visualize how multiple sites interact one another.

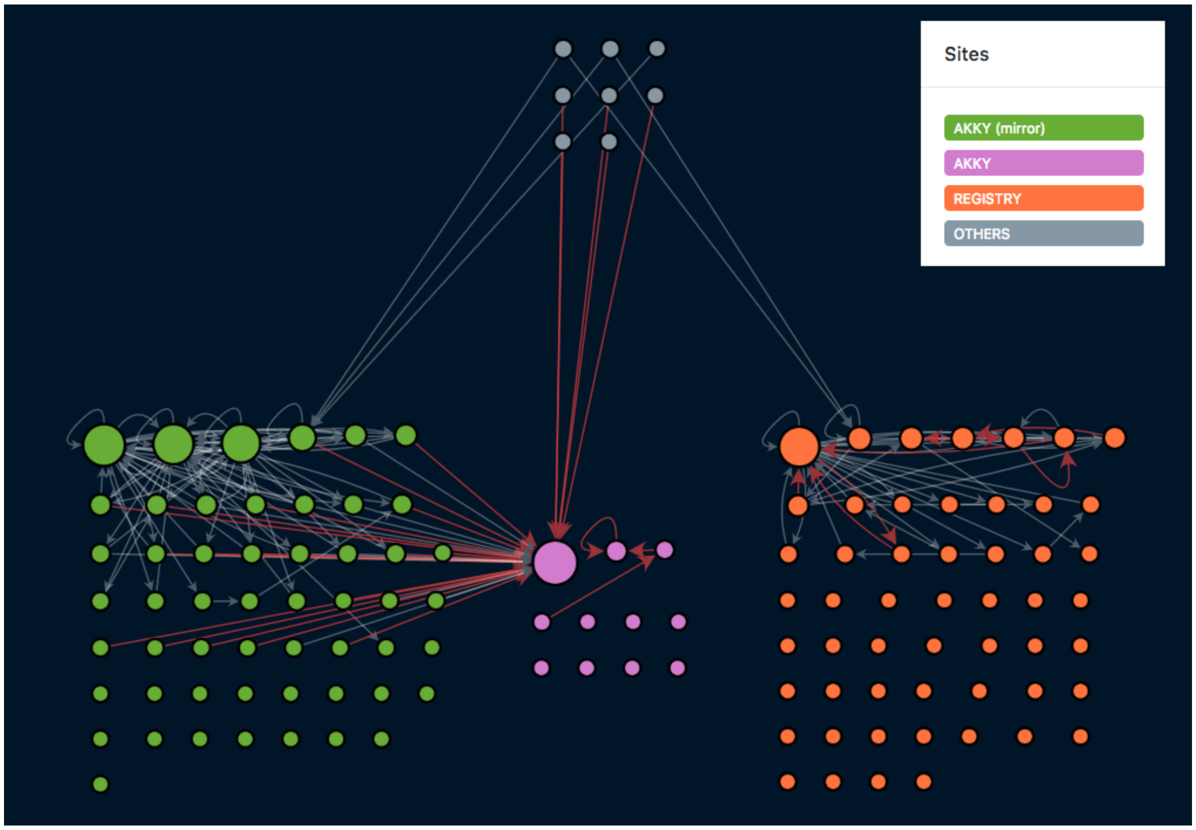


Figure 3.5: Multi-site View. We are analyzing three sites: each color represents a different domain. Grey nodes are external referrers (e.g. Google, Facebook, another website, etc.). Red connections are visits from bots we identified. The right sidebar shows information of the selected page.

We used the previous concepts of representing visits (Item **S.3**), to create a new visualization that includes the interaction between multiple websites. Figure 3.5 shows how multiple sites interact with each other and where the traffic is coming from. Red connections represent visits from bots; as we can see, many visits for the selected page (*www.akky.mx*) are coming from the referrers and from the green website.

3.4 Analysis of web traffic using the tool

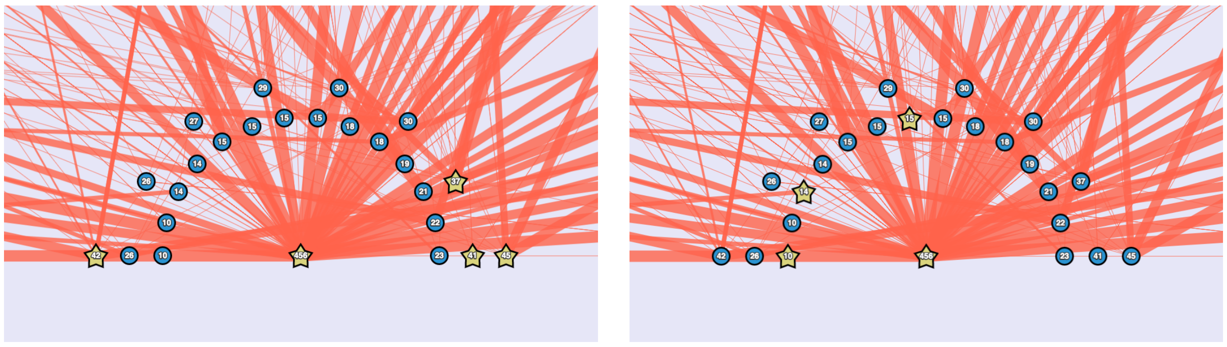
Observing the performance of objective pages and spot new opportunities by designing a business strategy for the most visited web pages

Figure 3.3 displays the traffic received on the website from a two weeks period of time. The metric we used for page nodes is the number of *page views*; whereas the *visit duration* was used for the connection between a visit (country flag node) and the page. The metric is also displayed above each node. Then, we are going to refer as the best or worst page, to that one with the greatest or lowest metric, respectively. We can notice several things from this visualization. If we focus only on the pages, we can see how many page views each one has. We can define levels for how many semi-circles we want and use it as a *Key Performance Indicator (KPI).*As an example, let say to define a goal as follows: *objective pages should be at least on k-1 outermost level*, and then measure the performance using that KPI as a basis.

In the examples shown in every figure, we have 3 levels. The farther from the center (close to the visit nodes), the better. From Figure 3.3, the worst page is the one on the left side (with 10 *page views*), within the first concentric circle. The best page is at the right (with 45 *page views*), in the outermost page nodes semi-circle.

From this example we can also spot the performance of the objective pages (star nodes). We would want that such pages to be on the outermost level, if not, they may not be reaching well the visitors. Nevertheless, we can spot *new opportunities* looking at the last semi-circle. There are nodes not considered as objectives, but they are having a lot of views. Thus, we can implement some call-to-action elements inside there, such as advertising, banners, promotions, etc. As an example, Figure 3.6 shows a comparison of how good (left-hand side) and bad (right-hand side) performance looks like.

If we look at the entire picture (Figure 3.3), we can see how visits (country flag nodes) interact with the pages. The first insight we can get is the countries that have more visits, i.e. how many nodes they have. We see a lot of visits coming from Mexico, Spain, and Vietnam. By selecting a visit, we can get more information like: which pages were viewed exactly, how many views does it represent contrasted with the overall site (red and green numbers at the right-sidebar), the visitor IP, where it was located in a map, its visit duration and many more details (Figure 3.7).



(a) Case 1. Good performance: star nodes in the first level (healthy).

(b) Case 2. Bad performance: star nodes mixed in the closest levels.

Figure 3.6: Objective performance visualization through a concentric layout.

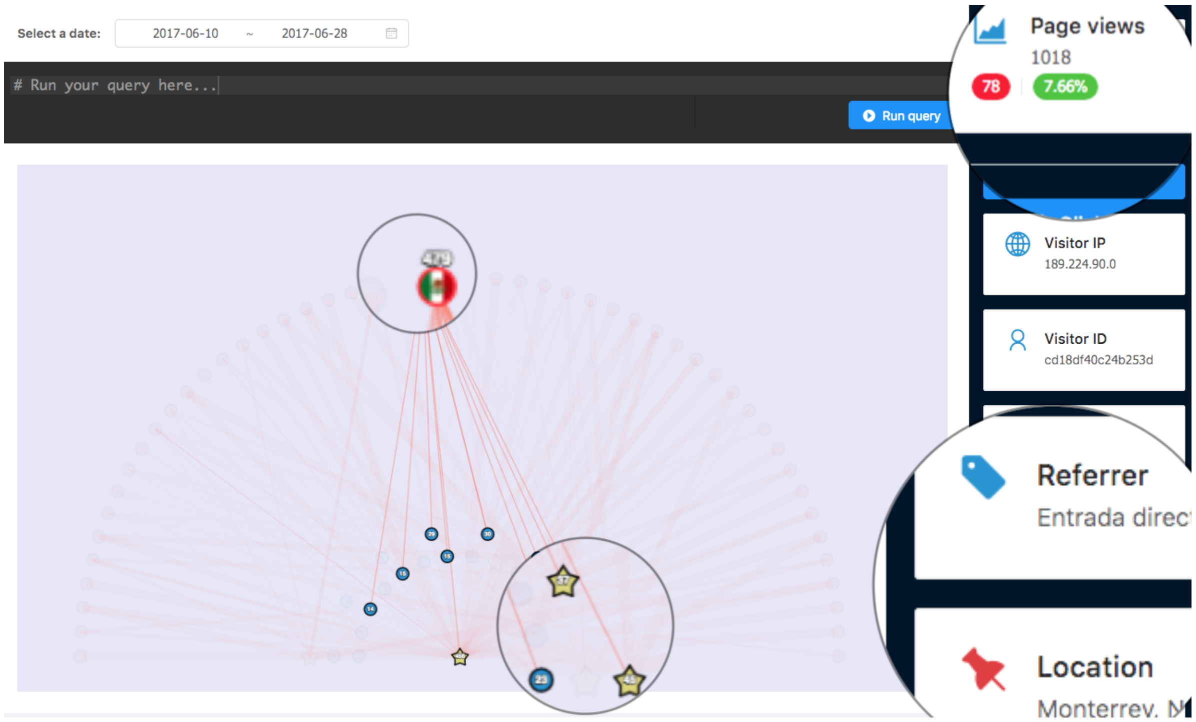
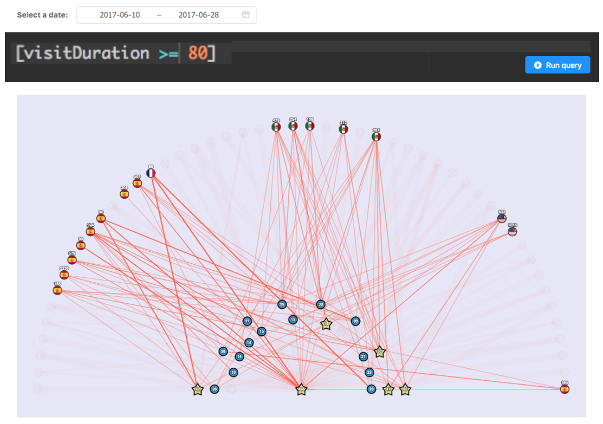
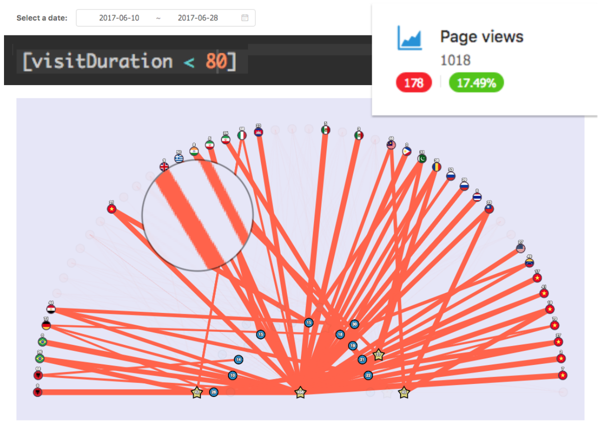


Figure 3.7: Visits View. Selection of a visit.

Furthermore, the connection between the countries and the pages represent (in this example) the *inverse visit duration*; i.e. the less time spent visiting the website, the thicker the connection. By doing so, we can see a clear pattern: visits with thicker edges, short visit duration; and visits with thinner connections, longer visit duration. This clearly resembles what we were looking for; bot traffic, those with thicker edges; and human traffic, the thinner ones.



(a) Human Traffic. Query: $visitDuration >= 80$



(b) Bot Traffic. Query: $visitDuration < 80$

Figure 3.8: Query functionality implementing ML Patterns.

As you can see in Figure 3.8, we used the *query* feature of our tool in combination to the ML Patterns extracted to find such visits (Item **S.5**). The amount of bot traffic according to the pattern, represents around 17.5% of the overall website traffic.

Clicking into the *Click Path* button, we can explore in detail how the user was navigating through the website (Figures 3.4 and 3.9). This view also allows to see if the user passed through an objective page and how he/she reached it. It also tells you for how many times the page was viewed. The exploration of this information could help to improve the site structure and lead more users to the business' goals, for example, by observing at which point of the path the user gets stuck and cannot reach to a certain objective page; even more, it can be used to extract the critical pages which are in the middle of common paths and therefore place a *call-to-action* element (or even ads) at such page.

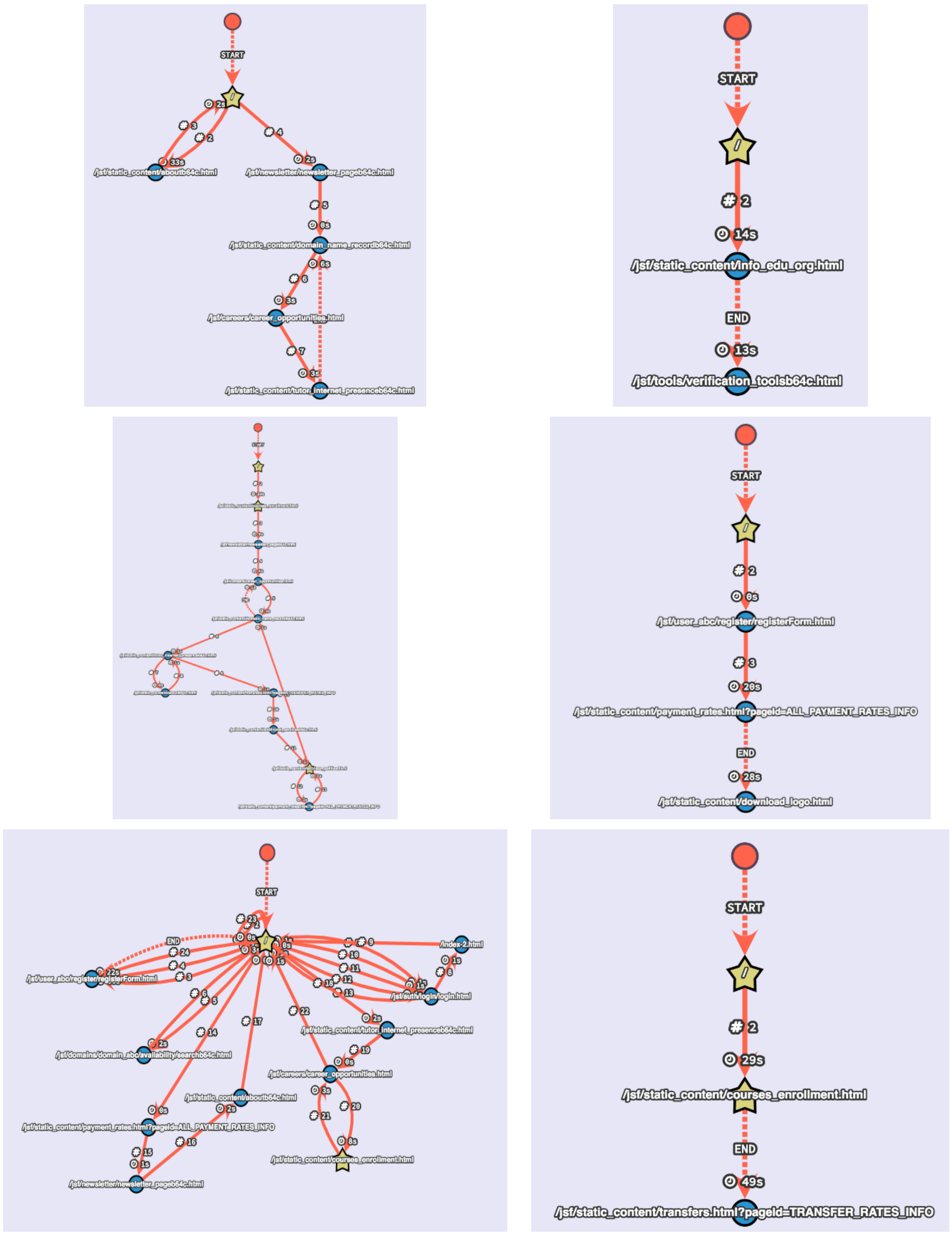

START
/
END
/jsf/static_content/aboutb64c.html
/jsf/newsletter/newsletter_pageb64c.html
/jsf/static_content/domain_name_recordb64c.html
/jsf/careers/career_opportunities.html
/jsf/static_content/tutor_internet_presenceb64c.html
/jsf/static_content/info_edu_org.html
/jsf/tools/verification_toolsb64c.html
/jsf/user_abc/register/registerForm.html
/jsf/static_content/payment_rates.html?pageId=ALL_PAYMENT_RATES_INFO
/jsf/static_content/download_logo.html
/index-2.html
/jsf/auth/login/login.html
/jsf/domains/domain_abc/availability/searchb64c.html
/jsf/static_content/courses_enrollment.html
/jsf/static_content/transfers.html?pageId=TRANSFER_RATES_INFO

Figure 3.9: Navigation Patterns. Left side: human traffic; Right side: bot traffic.

Another example would be to detect navigation patterns. Figure 3.9 shows a comparison of *human navigation* (left-hand side) versus *bot navigation* (right-hand side). In this example, we can clearly see some visual patterns: *bot navigation path is more linear*, whereas *human navigation involves a more complex browsing, with loops and more page views*.

As we validated the usefulness of the tool with certain domain experts, we confirmed the good potential it has for performing visit analysis and thus endorsing that by using this tool, we can have a better visual representation of a navigation path which. Such visualization enabled us to find *visual patterns* of human and bot traffic, whereas if using table reports (contrast it with Figure 3.2) we would not be able to find them that neatly. Figure 3.9 shows an example of such navigation patterns for humans and bots.

3.5 Conclusions

The motivation for developing this visualization tool was to help analysts to get insights from data and to support other reporting tools. We had very positive feedback from the marketing team we worked along with and they see this work as a powerful resource. Of course, the proposed work is a *tool* and the best use of it is given by the person who will be using it. For example, in our case, the before mentioned marketing team belonged to the Mexican company NIC México, and we carried multiple meetings with them to present the project. In such reunions, for example, the online experts were able to find irregularities on their website by using the visualization tool. One specific case was that they found multiple web pages with a lot of incoming traffic, similar to the index page. Suddenly, by using the tool, they found that the site presented multiple index pages (which they not recognized) where traffic was coming. That was an issue for user retention since the index page is one of the most important pages on a website. We believe that findings like that, and several others, could be found with the proposed visualization tool.

The tool itself is good to generate some insights of a website traffic and can be used to take decisions in the business. The contributions we aimed for in Section 3.3 can improve the way to analyze web traffic, by combining multiple reports into a single visualization, which can be used to assess an ad-campaign performance or even, by using Item **S.5** to audit reports provided by advertisers and compare how reliable the information is. Regarding visit details, we have achieved a better visual representation of a navigation path (Figure 3.4 compared to Figure 3.2) which enabled us to find *visual patterns* of human and bot traffic, whereas using table reports we would not be able to find them that easily. The tool was presented to a couple of domain experts from a Mexican company with an e-commerce website. They gave us positive feedback mentioning the useful potential it may have for their marketing analysis. Also, new suggestions to improve the tool were received, for example, integrating it with funnel analysis; another suggestion from the experts was to incorporate aggregated analysis (not just for one visit, but for all of them according to certain filters) in the Click Path visualization (Item **S.4**). Analyzing web traffic is a day-to-day task for some people like marketers, security teams, product owners or even researchers; as confirmed to us by some domain experts, this tool can be very helpful for such tasks and along with other visualization tools, it could provide powerful support for decision making.

Chapter 4

A Machine Learning Approach to detect Non-Human Traffic

Web Analytic services are one of the principal tools that marketing analysts use to measure the performance of some online business. Thus, it is extremely important to have reliable information about how many visits the website receives, where do the visits come from, how much time the users spend browsing and, most important, how many of them are making a conversion (a conversion is the completeness of a predefined business goal, e.g. visit certain web page). Knowing the fact that around 52% of Internet traffic is coming from Non-Human interactions [78], and that almost 80% of such traffic is considered sophisticated [48], identifying such invalid traffic becomes critical to provide reliable web analytics information. It would not make sense and may produce financial loses determine the success of an online campaign if visits, or page views, are used as a metric and most of the traffic is coming from bots.

A marketing analyst could be fooled that the business is having a success, then invest more into a product or campaign, and then ending up with an unexpected breakdown guided by the unreliable information about website traffic. Since *Non-Human Traffic (NHT)* is a current problem, the *Media Rating Council (MRC)* – the organization that dictates standards for digital metrics, recently updated their traffic detection guidelines addressing NHT [52]. However, not many of the big providers like Google, Youtube nor Facebook are up to such guidelines. Because of that, organizations like Havas group (O2, Royal Mail, BBC...) and many others [49, 74] are stopping giving money to those providers until they provide reliable measures.

The previous situation arises the need for better services to detect NHT. This work provides a tool to help the marketing expert to detect NHT; then, decisions and analysis of website traffic can be supported with more reliable information.

In this chapter, we are going to explore two different approaches to create a Machine Learning classifier that will help us to identify well-known bots and also anomalies in website traffic, which may belong to NHT.

We will start with a review of related work on bot traffic detection (Section 4.1) and compare such approaches with ours. Then, in Section 4.3, we will take a first approach to build a two-class classifier and use Decision Trees (DT) to have an interpretable set of classification rules, which will give us insights on what is going on with bot traffic.

Subsequently, because a two/multi-class approach may not be always feasible, in Section 4.4, we are going the explore an anomaly detector, i.e. a one-class classifier that will be able to detect traffic as human or as an anomaly. For doing such task, we are going to use well-known one class classifiers in the literature, like Isolation Forest [43], Weka's integrated One Class Classifier [33], Cost Sensitive Classification [70], Bagging TPMiner [50], Bagging Random Miner [14] and Local Outlier Factor [12].

Finally, in Section 4.6, we are going to draw conclusions from both experiments, evaluate which classifiers had the best results, and evaluate when would it be better to use the first approach or the second.

4.1 Related work

Bot recognition has applications in several domain areas, but the focus has been mostly in security, online advertising, and ad-fraud detection. The latter two areas are the ones with most interest to us because it complements perfectly with the Web Analytics seen in Chapters 2 and 3.

There are tools in the industry to detect Non-Human Traffic, but many of them are private technologies owned by companies like comScore, distil Networks, Imperva [15, 20, 22]. Fortunately, there is strong public research on botnet detection. In the beginning, it was focused on IRC/P2P bots but it extended to many other protocols such as HTTP [65, 3]. Despite the existing related work, many of them are focused on a low-level analysis of network packets and only a few employs less intrusive methods, such as the analysis of higher Network layers, like the Application Network Layer, which is the focus of this investigation.

Specifically for mobile advertising, click fraud is very common [42]. Given that situation, there exists a high risk of wasting money in ad-campaigns provided by publishers that cannot have reliable bot traffic filtering. Past research has focused on some known datasets; the first one is the so-called *FDMA*, which stands for *Fraud Detection in Mobile Advertising*. It was released as part of a Workshop in 2012 [57], with three main datasets for training, validation, and testing. The common protocol to follow with this dataset is to train a model using the training set (*09feb12*), choose the best model using the validation set (*23feb12*), and then test it with the testing dataset (*08mar12*) using *PRC Area* [63] as a metric. The best PRC in the competition was 59.39% and 51.55% for the validation and test datasets, respectively. Later, Taneja [69] created new features based on time granularity statistics, incrementing the accuracy to 97.23% for validation and 64.07% for testing. The best model was a *Hellinger Decision Tree (HDT)* [18] with *Recursive Feature Elimination (RFE)* [28]. His score is the best PRC reported in the literature so far [59, 10, 69].

[76] used a similar approach to [4] and created time-based features to detect abnormal traffic. The dataset used was privately collected from an ad exchange company and it contained a 1-day log with around 0.13 billion impression records, 8.5 million click records, and about 1.85 million conversion records. The model with the best performance was a Random Forest with *Gradient Boosting Decision Tree (GBDT)* [53].

On the other hand, [36] followed an unsupervised approach and used clustering analysis to detect crowd-sourcing click-fraud in search advertising. Again, they worked with a private dataset (not available to the public); their main strategy was to analyze the search query and group clicks with the same IP and the same advertiser, as non-fraud.

Recently, Kaggle launched the competition "*TalkingData AdTracking Fraud Detection Challenge*" [68], where the Talking Data Chinese company released a dataset very similar to the FDMA. The company handles over 3 billion of clicks per day, of which 90% are potentially fraudulent, so clearly they provide a real-world big-data problem, with a huge opportunity for further research. As today, the best result evaluated by the ROC Curve is 0.9797.

An extensive research of benign URL ad detection can be found in [79], they created a taxonomy of the methods employed by previous works and proposed new lexical-based features to detect malicious-ad URLs. Most of the features are focused on preprocessing URL properties such as the hostname, URI length, resource extension, etc. They collected URLs from three main sources [2, 73, 6].

HTTP CSIC 2010 [71] is a dataset commonly used by some investigations involving *Artificial Neural Networks (ANNs)*, fuzzy logic, and other machine learning methods [72] but even that this is an already labeled dataset, they focus on detecting anomalous traffic oriented to web attacks like *Cross-Site Request Forgery (CSRF)*, SQL injection, and malicious payloads in general. Botnets used for click fraud do not necessarily involve web attacks, so addressing the problem with such techniques do not provide us with a full covering of the problem.

There has been also further research on advertising but focused on CTR prediction. Criteo, a global technology company specialized in the performance of display advertising, launched in 2014 a challenge in order to answer the question of "Given a user and the page he is visiting, what is the probability that he will click on a given ad?", i.e. click-through prediction [16]. The winner of the competition used Field-aware Factorization machines (FFM), getting the best LogLoss of 0.44463. The report of their methodology and more information about FFM can be found in [37]. A similar dataset named Open Advertising Dataset was collected by the University College London (UCL) [73]; it includes ads from May 2012 to February 2013, covering US and UK markets. A brief analysis of such dataset is done by [58]; they focus on present time-based patterns and also on CTR prediction. [75] studied feature selection on a Microsoft AdCenter dataset; they generated new features based on time intervals and also sequential information of the clicks. Their conclusion was to select a subset of features and use *Conditional Random Fields (CRF)*, which was the best model in their experiments using F-Score as a metric.

We can observe some aspects that previous works share in common; first, the best models were based on decision trees and a boosting mechanism; second, generating time-based features like the number of clicks in a certain time-window, absolutely helps to improve the accuracy of the model. The latter also reduces the dataset dimensions in terms of instances, creating a balance between the number of class instances, helping to the imbalance problem.

4.2 Motivation

Because botnets used for click-fraud do not necessarily involve web attacks, addressing the problem by banning IPs, back-listing User Agents or inspecting low-level network packets [65, 3, 40, 64], do not provide a full coverage of the problem. Such methods may not always be possible to implement and perhaps they will require a high amount of work. Thus, implementing an autonomous anomaly detector surely is a good option to use in this case. Our current research is addressing the problem through a novel methodology involving the analysis of user behavior with machine learning algorithms and visual models.

4.3 A first approach to a Two-Class Classification for Bot Detection

Building a two-class model can help to detect *well-known bots* which are commonly used in the community. The current state of bot tools available on the internet do not include any kind of structured collection of resources, it is often the case that bot applications are published on web forums and then downloaded by people; that situation leads to a series of outdated bot programs currently running on the Internet, thus, they are not often updated by their third=party owners (people do not created the bot, just using it). That situation assembles to how virus and anti-virus community works; common viruses are identified by the community and then counter-measures are implemented. Usually, anti-virus companies create indexes of such well-known viruses and a software *patch* to detect them.

A similar solution could be implemented in the case of bot applications. Although it may not be as easy to detect a bot as a virus, Machine Learning classifiers can be used to provide such identification. Two-class classifiers (and maybe multi-class) could be implemented as pre-trained models to detect well *known-bots*. It may also be the case to use *One-Class Classifiers (OCC)* but they usually are trained with *owned* data, and using pre-trained may not always be possible because they learn from *specific distributions considered as normal*, which are case-independent. i.e. if you use an OCC to detect *abnormal* traffic, the very concept of abnormal is tightly coupled with what is considered as *normal*, and thus to build the classifier, it must be trained with your own data. To avoid such situation, a pre-trained two/multi-class classifier may be used as a plug & play solution, which can be further grown by identifying more and more well-known bots. Nevertheless, both approaches, two-class, and one-class, can be complemented one another and it may be better to have both of them than just one. In this section, we are going to explore the two-class approach, and in the next section, we will explore the one-class option, which, as we will see, may not be perfect and that is why both approaches can be complemented.

Before we start, we shall say that the collected dataset used in this experiment (binary classification) suffers from a high bias. We ran the experiments in a controlled environment, which means that some of the information we have, could not be reliable to make a generic classifier. For example, we consider that the attributes of date, time and geographical location should not be used in the classification process. It may be obvious why; bot traffic were generated at specific dates and at a specific time. Precisely, from June 6 to June 14, before the afternoon of each day. So, a classifier could easily choose those date ranges to classify bot instances and it would have the maximum precision. On the other hand, we have human traffic from June 16, 2017 to July 3, 2017; the same problem as before applies and in this case, most of the human traffic comes from Mexico, so again a classifier that chooses visits with $countryCode = mx$ could easily achieve the maximum precision for human classification.

The previous situation implies that, for this experiment and only for building a classifier with this dataset, we throw away all the attributes related to date, time, and geographical location, with the exception of `continentCode`, which we hypothesize will not affect that much as the specific countries. It is an expensive trade-off, however, we believe that the rest of attributes that represent browsing behavior are as important as the location attributes, e.g. `pageviews, visitCount, visitDuration, clicks.` With this hypothesis, we expect to mitigate the trade-off of removing such attributes for classification.

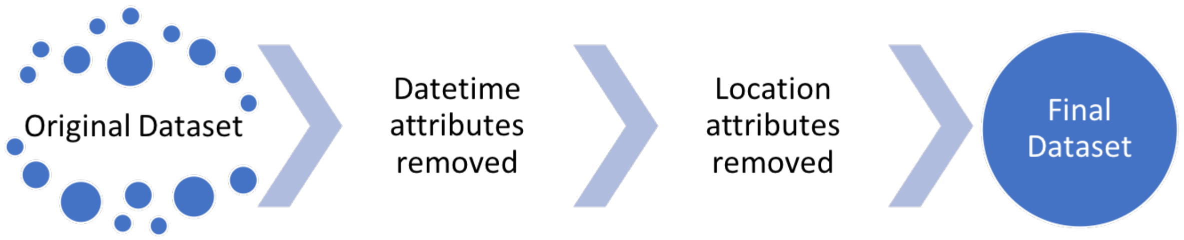


Figure 4.1: Dataset modifications for testing performance. The first one does not have any attribute modifications at all; the second one has removed date and time attributes; the third one has both date, time and location attributes; the final dataset includes the removal of the six attributes mentioned in Section 2.3.

Figure 4.1 shows how we are going to pre-process the dataset, removing the information previously mentioned. Because we are going to use Decision Tree based models, we will run a *J48* classifier for each dataset variation and test how it well performs against the modified dataset. The metric to measure performance is the AUC under the ROC Curve because it is useful to find an *optimal threshold* used as discriminant for classification. Once we have the final dataset, we will test more classifiers against it and see which one performs better. Also, this is the dataset we use for the *Pattern Extraction* process (Chapter 5).

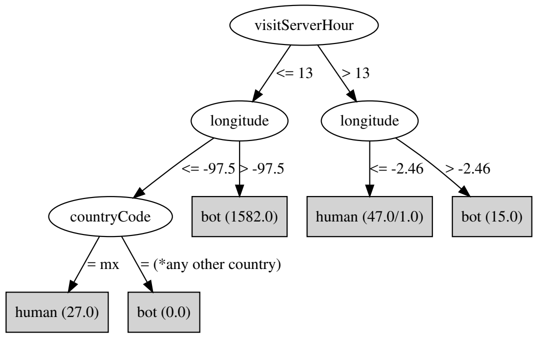
4.3.1 Dataset variations and Decision Trees

The classification process was done using Weka. The selected classifier was J48, with the default parameters and using a 5-fold Cross Validation. For reproduction purposes, we should mention that the seed used for all the experiment was the number 1, and also that the visit IDs and site IDs were removed.

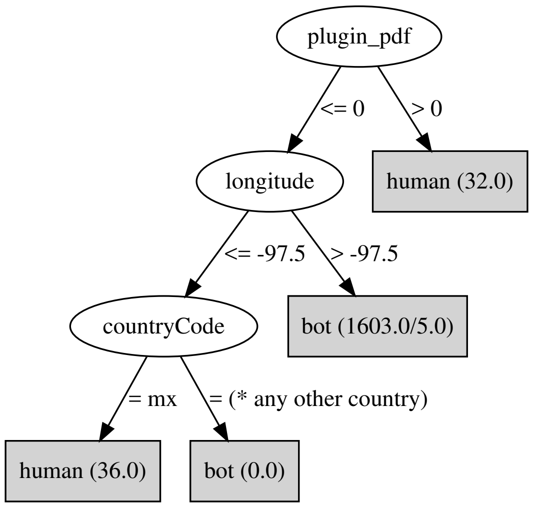
Dataset	**TP Rate**	**FP Rate**	**Prec.**	**Rec.**	**F-M.**	**ROC Area**	**PRC Area**
Original	0.996	0.092	0.996	0.996	0.995	**0.964**	0.994
No D.Time	0.995	0.079	0.995	0.995	0.995	0.958	0.992
No D.Time-Loc.	0.985	0.316	0.985	0.985	0.984	**0.844**	0.973
Final	0.985	0.316	0.985	0.985	0.984	**0.844**	0.973

Table 4.1: J48 5-fold Classification results for the four different dataset versions. Decision of which one to keep was done by taking into account the ROC Area metric. *D.Time = No Datetime; D.Time-Loc. = Datetime-Location; Prec. = Precision; Rec. = Recall; F-M. = F-Measure*

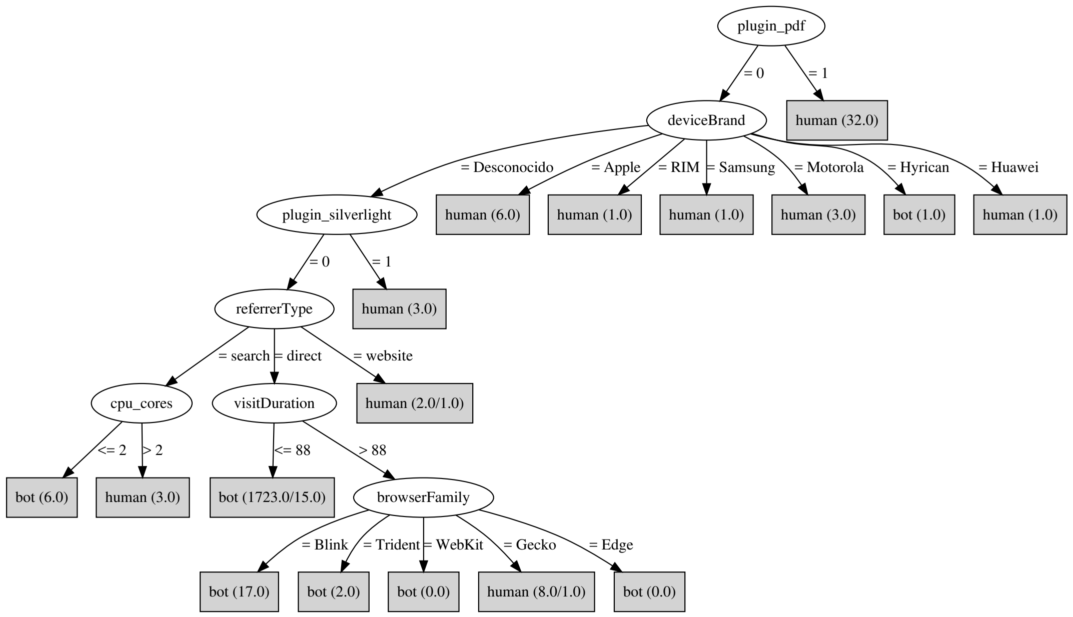
As shown in Table 4.1, the AUC for the original dataset was 0.964 and as we started to remove attributes, the performance decreased to a final value of 0.844. We lose around 12%, however, it is still a good metric value considering we are dropping many important information such as the visitor's country and the time they accessed the website. By observing Figure 4.2, it seems that behavior and device information are very useful by itself: browser's plugins information, device brand, visit duration and referrer type, are the chosen attributes by the classifier as important attributes for the decision.



(a) Original.



(b) No Datetime.



(c) No Datetime-Location and Final.

Figure 4.2: Datasets and their J48 Decision Trees. Each tree represents a variation on the dataset. The *No Datetime-Location*'s tree resulted in the same as the final's. The note *(* any other country)* represents a series of nodes, each one being a country (e.g. USA, Germany, Russia...) and all of them ended into a bot leaf. Such nodes were removed and resumed in this general node in order to reduce the tree size.

Figure 4.2 shows the built decision trees by the J48 algorithm. As we expected, the DT (*Decision Tree*) generated from the original dataset is very concise because the biased attributes (location and datetime) are basically all it needed to perform a good classification. The `visitServerHour` is the root of that tree, followed by the location attributes of `longitude`, `latitude`, `countryCode`. This is not useful for us to build a good classifier because we may not be able to have a controlled environment in the future. Nor does it considers behavior attributes like the duration of the visits or the number of pageviews made by the visitor.

As we proceeded to eliminate the datetime attributes, `visitServerHour` is no longer the root node (Figure 4.2 (b)), however, the location attributes still remain to be very relevant. With this attribute removal, the ROC Area dropped only 0.6%, which is not that bad. Proceeding with the removal of the location attributes, we ended with the tree Figure 4.2 (c). In this case, we see a more complex tree, and although we lose around 10% of AUC, it includes extra relevant features like the visit duration, the plugins installed on the visitor's browser, information about their device and even the kind of visit referrer. The *final* dataset includes the removal of the *six* attributes mentioned in Section 2.3 because of their high *missing values fraction* and high correlation coefficient.

4.3.2 Conclusions

Taking into account the Decision Tree built using the final dataset (Figure 4.2c), and by performing the classification experiments, there are several insights we can extract from the results,

- Date and time locations may not be always very important for classification processes. By removing them from the dataset, the AUC decreased at most 12%.
- Behavior and device information attributes such as: *device brand, plugins, visit duration, device's cpu cores and browser family*, are useful attributes chosen by a J48 DT to build a classifier.
- It is possible to create a two-class classifier to identify Non-Human Traffic for common bot tools like Traffic Spirit, Jingling Bot and w3af.
- In Figure 4.2c we have a lot of *pure nodes*, which is desirable. From 17 leafs we have, only 3 are impure, so around 82% are pure leafs.

- Most of the bot instances fall into the leaf led by the path:

  plugin_pdf = 0 $\wedge$ *deviceBrand = "Desconocido"* $\wedge$ *plugin_silverlight = 0* $\wedge$ *referrerType = "direct"* $\wedge$ *visitDuration* ≤ 88.

  This implies that many of the bot visitors came directly to the website, their device brand could not be found, they do not have a PDF plugin installed in their browser and the duration of their visits is less or equal than 88 seconds.

- By following the branch `plugin_pdf = 1`, we get a pure leaf classifying humans. This implies there was not the case, not even once, were the bots' browser had installed a PDF plugin. It may be intuitive, because for us, as humans, usually it is often to open multimedia files in the browser, one of them are documents are PDFs, so it is high probably that we have a PDF plugin installed. However, for bots this may not be the case, because they may not even care about such things, they just open web browsers pragmatically in order to accomplish some task.

- We confirmed our hypothesis that the selected attributes for removal in Section 2.3 were not useful, because as shown in the results, there is no change in the *Area under the ROC Curve* between the final version and the previous one (No Datetime-Location).

- Two-class classification of bots is useful when there exists already known information about bots; it can be used to construct iteratively a multi-class classifier. For example, as anti-virus companies do, one can construct a reference of known-bots and keep it growing once new bots are found. Then, a simple multi-class model can be built and deployed to detect common bots. This approach may not need to be trained with personal data and thus implement it as a plug & play method, whereas using, for example, a one-class approach, would be needed to learn from a current data distribution.

4.4 Building a One-Class Classifier for anomaly detection

The Internet is evolving every second, and thus software. Malicious tools and bots may not take too long to get updated or emerge new ones; therefore, we may not be able, in time and resources, to search and identify every possible bot, create a classifier (either two-class or multi-class) and use it to be protected. However, there is another approach we can take: building a *one-class classifier* to detect anomalies. If we count with a good number of samples of what we presume is *normal traffic*, we can build an anomaly detector and whenever traffic gets unusual, we could have something to pop-up an alert and investigate further on the matter. Thus, building a one-class classifier seems a good option to protect a website from Non-Human Traffic. We are going to explore this approach in the rest of this chapter and analyze the results obtained.

4.4.1 Experimentation details

To build the one-class classifier we used a different dataset. Later on, the research we were provided with *web server log* files from a company which has an e-commerce site in Mexico. The company's logs include information of traffic to their website for a period of *five months*. It contains over *7.4 million requests*, which were imported and pre-processed in Matomo. By default, Matomo ignores requests to static files (such as CSS, images or JavaScript files) and malformed requests. The import process ended up with around *740k requests* imported and it was able to re-create *116k visits*. The number of visits is smaller because a single visit could include many requests.

We trained and tested several configurations of one-class classifiers:

1. Weka's One Class classifier [33]
2. Weka's Cost Sensitive Classification [24]
3. Local Outlier Factor (LOF) [12]
4. Bagging TP Miner [50]
5. Bagging Random Miner [14]
6. Isolation Forest [43]

All the experiments were performed in Weka, and for the case of Bagging TP Miner [50] we used C# .NET Programs. Every configuration starts with the default values provided by Weka, nevertheless, it is worthy to mention for reproduction purposes, that the seed used on algorithms has the default value of 1. Results are going to be evaluated by the ROC Area metric.

Since we had a large amount of information, which was not very unbalanced, an *inverse* kind of 5-Fold Cross-Validation [53] method was used. We divided the dataset into five subgroups using the *stratified k-folding* method. Then, we built five different models. In the first one, we used the first fold to train the model and the four remaining to test it. Then, the ROC AUC was computed. Consequently, in the second model, we used the second fold to train and the remaining four to test, and so on. In the end, we reported the mean of all the ROC AUCs computed.

The purpose of using this method was to avoid overfitting in the models, which was validated by using the four folds. Then, we could select the best model from the five we trained.

4.4.2 Results

Weka's One-Class Classifier

An OCC already included in Weka. According to the package documentation, it learns the data without using any information from other classes. So, during build time only one class is considered. Nevertheless, in order to calculate the *outlier pass rate* the dataset must contain information from more than one class. The testing stage will classify an instance as *target* or *outlier*, where *target* is the name of the class we are training for, in our case, the *human* class.

By default, this algorithm just wraps a process for performing one-class classification. On the inside, it uses bagging plus a specific classifier (e.g. J48, Naive Bayes, Random Forest, etc.); the default configuration is Bagging plus REPTree [38]. We chose to modify the classifier (REPTree) as hyper-parameter and see how it well performs.

Table 4.2 shows the results of the experimentation. The SMO configuration suffered from a time explosion and never ended. Although many configurations have a ROC Area around 0.83, we may be fooled if we think it is performing well. If we look further into the Confusion Matrix of each model, Table 4.3, we can see that *it is classifying everything as human*. That is why we do not have a computation for the F-Measure.

Given such results, this classifier will not be useful for us at all.

#	Params	ROC	F-M
1	**Default**	**0.834**	**?**
2	(NaiveBayes)	0.827	?
3	(SMO)	-	-
4	(J48)	0.562	?

Table 4.2: One-Class Classification: Weka's One Class Classifier experiments and results. F-M (F-Measure) could not be computed because of the complete misclassification of the bot class.

a	**b**	**← classified as**
76,771	0	**a = human**
5,601	0	**b = bot**

Table 4.3: Weka's OCC Confusion Matrix. All the instances are being classified as human. 76,771 human instances were classified correctly, but all the 5,601 bot instances were classified *incorrectly*.

Weka's Cost Sensitive Classifier

Because previous results were showing a tendency to classify everything as the *human* class, we decided to implement a *cost sensitive classification* [24]. Although commonly used for class imbalance problems, the intuition behind its objective works the same for use: penalizing mistakes more for one class than the other.

According to Weka's package documentation, this algorithm is meta-classifier that makes its base classifier cost-sensitive. Two methods can be used to introduce cost-sensitivity: re-weighting training instances according to the total cost assigned to each class; or predicting the class with minimum expected classification cost (rather than the most likely class). *We opted for the latter*.

In order to perform the cost-sensitive classification, a requirement is to have an associated cost-matrix. As applied in the literature [56], we used the heuristic of cost computation as $C_{ij} = N_i/(N_i+N_j)$. Where C_{ij} is the cost for mistaking the i class, and N_i, N_j are the number of instances in the respective class i or j.

#	**Parameters**	**ROC**	**F-Measure**
1	**(NaiveBayes)**	**0.858**	**0.879**
2	(J48, true, 1.0 / 2.85)	0.573	0.503
3	(RandomTree)	0.532	0.663
4	(BayesNet)	0.729	0.833

Table 4.4: One-Class Classification: Cost Sensitive Classifier experiments and results. Base learner is Weka's OCC and we just changed its classifier hyper-parameter as in Section 4.4.2.

Table 4.4 shows the results of the experiments. For the base learner we used Weka's One Class Classifier and just changed its classifier hyper-parameter, as done in Section 4.4.2. The *parameters* column indicate such modification.

As we presumed, Cost-Sensitive Classification performs better than using alone the Weka's OCC Section 4.4.2, and we get rid of the problem of *classifying everything as human*. We even achieved a better ROC Area than Section 4.4.2 and F-Measure can be computed with not such a bad value. The best configuration was with Naive Bayes and we achieved a ROC Area of 0.858.

Isolation Forest

According to Weka's algorithm documentation, in order Isolation Forest to work, the data is expected to have a class attribute with one or two values, which is ignored at training time. It returns two values, the first one is (1 - anomaly score), the second element is the anomaly score. Normal cases should correspond to the first value of the class attribute, anomalies to the second one. For this experiment, we arranged the dataset to fulfill such requirement; the class attribute had the following values in order: human, bot.

Five different configurations were tested for this algorithm. Table 4.5 shows the results of training and evaluating the model. Each row represents a different model configuration. The *parameters* column follows the syntax: (*numTrees*, *subSampleSize*), which are hyper-parameters of the classifier. The first one indicates the number of trees to use in the forest; the second one is the size of the sub-sample used to build each tree.

#	Parameters	ROC	F-Measure
1	Default	0.753	0.624
2	(100, 25)	0.857	0.770
3	(100, 10)	0.898	0.816
4	(100, 512)	0.761	0.625
5	**(500, 10)**	**0.920**	**0.845**

Table 4.5: One-Class Classification: Isolation Forest experiments and results. Each row represents a different model configuration. Parameters column follows the syntax: (numTrees, subSampleSize), which are hyper-parameters of the classifier. *Default = (100, 256).*

By observing Table 4.5, we can tell that the parameter with a higher impact on the classifier's performance is the *sub-sample size*. The smaller the better. We shall highlight that Isolation Forest gives us a decent ROC Area, but most of all, *it runs very, very fast*, which could be something we would want for larger datasets. The best result obtained was a ROC Area of 0.920 by using 500 trees and a sub-sample size of 10.

Bagging TP Miner and Bagging Random Miner

Bagging TP Miner (BTPMiner) [50] is classifier built with the purpose of a masquerade detection system in the context of computer systems intrusion. On the other hand, Bagging Random Miner (BRMiner) [14] is a further exploration of the same problem, but with an improved training time. BTPMiner has a training time complexity of $O(n^2)$, whereas BRMiner has $O(n)$. Nevertheless, BRMiner has a much higher testing time complexity than BTPMiner.

Both classifiers have proven to perform well on anomaly detection [50, 14, 45], that is why we decided to use them.

Classifier	Parameters	ROC
Bagging TPMiner	**(10, Mahalanobis)**	**0.9823**
Random Bagging Miner	(10, Mahalanobis)	0.9820

Table 4.6: One-Class Classification: Bagging TPMiner and Random Bagging Miner experiments and results. The *parameters* column has the following syntax: *(number of classifiers, distance function).*

The best result we obtained was with BTPMiner, using 10 classifiers and Mahalanobis [62] as the distance function. Table 4.6 shows the results of the experimentation. The *parameters* column includes the configuration of the classifier and has the following syntax: *(number of classifiers, distance function).*

It is worth to mention that even when both classifiers have similar results, experiments with BTPMiner had better time complexity than the enormous amount of time taken by BRMiner. Consequently, we decided it would be best to use BTPMiner.

Overall Best Results

Table 4.7 exhibits a summary of the best configuration for each model and their results. The best classifier was Bagging TPMiner with a ROC Area of 0.9823. We would like to say that the second best is *Isolation Forest* with a ROC Area of 0.92. We excluded *Bagging Random Miner* due to its excessive amount of time for testing.

Classifier	**ROC Area**
Bagging TP Miner	**0.9823**
Isolation Forest	0.9200
Weka's CSC	0.8580
Weka's OCC	0.8270
Local Outlier Factor	0.4980

Table 4.7: Best results for one class classification. CSC = Cost-Sensitive Classification (Section 4.4.2); OCC: One-Class Classification (Section 4.4.2). Classifiers were evaluated by ROC Area.

4.5 A further exploration of data using the OCC

Once we built the One-Class Classifier, then it can be used to detect anomalies present in the traffic. We had the hypothesis that *there may be some instances within the human traffic that we did not know they were bots, i.e. some hiding bots which cannot be noticed a-priori*. Thus, a possible way to detect them is with the one-class classifier.

To test such hypothesis, we ran the classification for all the human instances. If no bots present, all the instances should be classified as *human*; otherwise, if there are some instances classified as bots, they may be indeed *Non-Human Traffic (NHT)* which we missed before.

The classifier used was *Bagging TP Miner (BTPM)* since it produced better results in previous experiments. As a result, BTPM returns a *measure of similarity*, a value $\in [0.0, 1.0]$. A value closest to 1.0 indicates that *there is great chance* to belong to the *human* class, *otherwise*, as it gets close to 0.0, it may belong to the bot class. For each instance, we compute such similarity value. Then, *if $value > Threshold$, it is classified as human, and as bot otherwise.*

The *Threshold* value is usually selected from a ROC analysis that provides what value would be better to use. Intuitively, the analysis involves selecting multiple values for the threshold, and then testing how it well performs for such threshold.

We selected as $Threshold$ the value correspondingly to the $ZeroFMR = 0.60699$. *False Matching Rate (FMR)* is the percent of invalid inputs that are incorrectly accepted. Consequently, *ZeroFMR* is the threshold value such that it produces a value close to zero for the *FMR*. Thus, *ZeroFMR* is the threshold for which we expect to not have any bot instances classified as human.

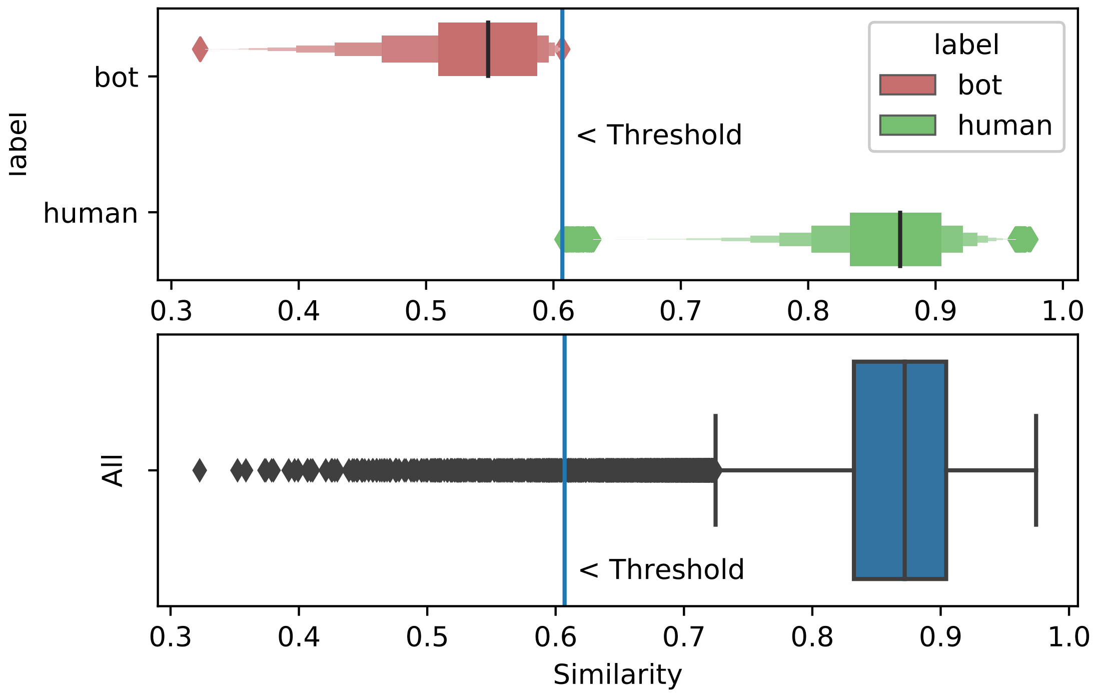


Figure 4.3: Similarity Distribution. Boxplot on the top for each class, global box-plot on the bottom-side. Instances were classified as *human* if $Threshold > 0.60699$, *bot* otherwise.

Figure 4.3 shows a distribution for the classification results (the similarity value). The figure contains two box-plots, the one on the top is for each class, and the one on the bottom is for the whole data. The horizontal axis represents the similarity computed for the instances by BTPM. The *Threshold* line indicates the value we described before for performing the classification.

Comparing both box-plots, we can see that almost every instance has a high value of similarity (blue rectangle); great percentage are concentrated in a range between $[0.83, 0.90]$ of similarity, which is what we expected since all instances are from the human class. Nevertheless, there are few cases for which the similarity value is *less that the threshold*. Specifically, from 85099 instances, *276* were classified as bots. For such cases, we looked further into those instances and then validated the classification result.

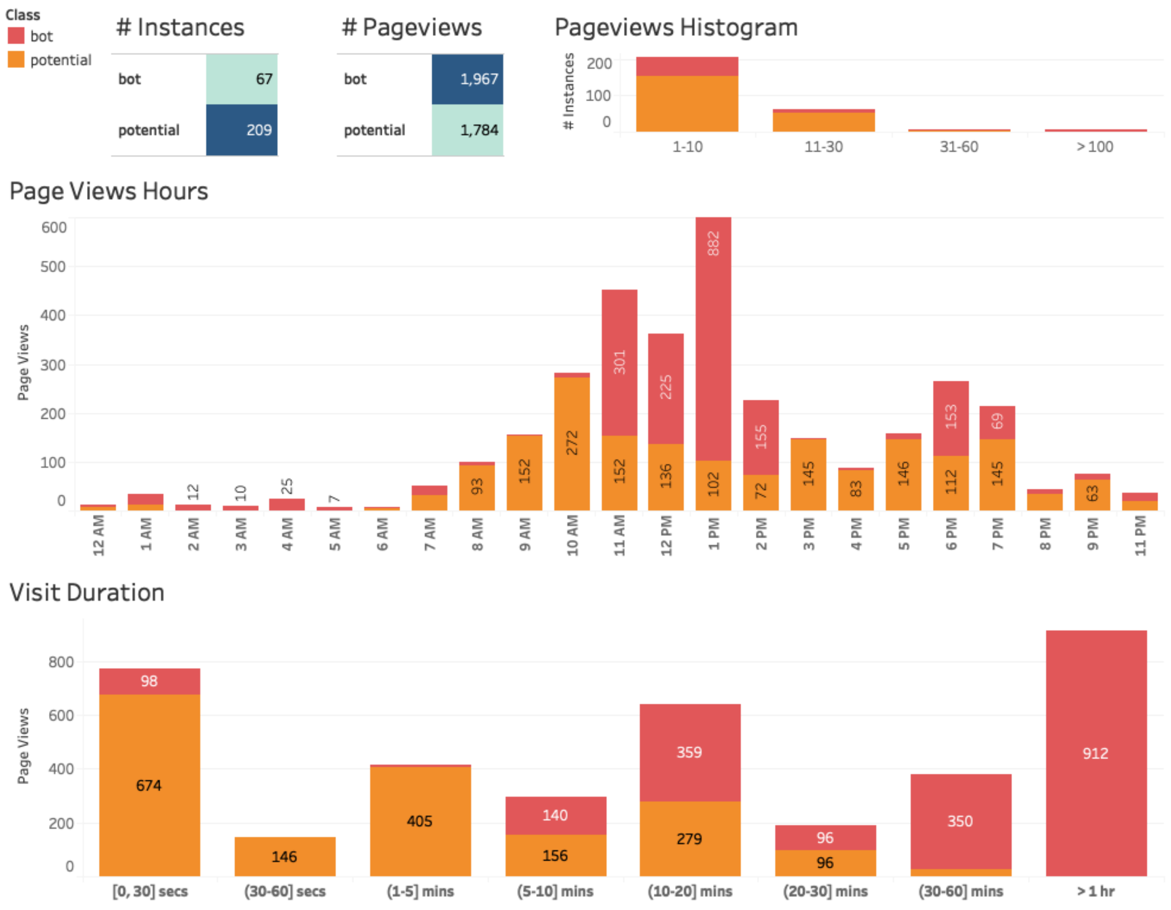


Figure 4.4: Information about the 276 bot instances. 67 are bots indeed, the rest may also be, there is not enough information to ensure that.

Indeed, we found some hidden bots inside the traffic. Figure 4.4 shows an exploratory data analysis for the visits. From the figure we can see several things:

- Almost every visit with a *visit duration* $>$ *30 min.*, is indeed a bot.
- From the 276 visits, 67 were indeed bots, and the rest 209 may also be.
- The bots were responsible for around *2k page views*, and another *1.8k* may also be from bots. Both sum a total of *3,751* page views.
- Bots were more active around 11-AM to 2 PM.

Some of those visits were from *search engines and crawlers* like *Ahrefs*

Bot [1], *Bing Bot* [2], *Semrushbot* [3], *Proximic Crawler* [4], *a Chinese Search Engine* [5], *Qwant Search Engine* [6], *Hyperspin Service* [7] *and Flipboard Proxy Service* [8]. From those 276 visits, we left the bot class for those we were sure they were bots, and we assigned them the *potential* class for the rest.

4.6 Conclusions

When exploring a Two-Class Classification:

- We showed it is possible to create a two-class classifier to detect Non-Human Traffic generated by common tools like Traffic Spirit, Jingling Bot, w3af. For example, such classifier can be used to detect common well-known bots.
- Date, time and location attributes may not be always very important for the classification. By removing them from the dataset, the AUC decreased at most 12%.
- By building Decision Trees (DT), we were able to get insight from the website traffic. Concluding that many of the bot visitors came directly to the website, they did not have identification on their browsers device, they did not have a PDF plugin installed on the browser and their visit duration was less or equal than 88 seconds.
- Again, DT visual representation (Figure 4.2) give us a hint on what patterns do bots have in common. It is interesting to mention the fact of the inclusion of the browser's plugin information. For example, there was not the case, not even once, were the bots' browsers had installed a PDF plugin. It may be intuitive, because for us, as humans, it is common to open multimedia files in the browser (like PDFs), so it is highly probable that we all have a PDF plugin installed. However, for bots this may not be the case, because they might not even care about such things, they

[1] `https://ahrefs.com/robot`

[2] `http://www.bing.com/bingbot.htm`

[3] `https://www.semrush.com/bot/`

[4] `http://www.proximic.com/info/spider.php`

[5] `https://www.so.com/index.htm`

[6] `https://www.qwant.com`

[7] `http://www.hyperspin.com/esar/`

[8] `http://flipboard.com/browserproxy`

just open web browsers programmatically in order to accomplish some task.

- Two-class classification models can be built with good precision; this may be the start point to build iteratively multi-class classifiers and create datasets of well-known bots, similar to what anti-virus companies do when creating references for well-known viruses.

When exploring the One-Class Classification approach, we achieved very good results with certain anomaly detectors.

- Weka's One Class Classifier and Local Outlier Factor (LOF) did not perform well in our context because they showed a tendency to classify everything as the *human* class.

- As we presumed, Cost-Sensitive Classification performs better than the Weka's OCC Section 4.4.2 alone, getting rid of the problem stated in the previous point. We even achieved a better ROC Area. The best configuration was with Naive Bayes having a ROC Area of 0.858.

- The classifier with best ROC Area was Bagging TPMiner with a value of 0.9823. The second-best was Bagging Random Miner with 0.9820, and the third place was Isolation Forest with 0.920.

- A pragmatic decision for the anomaly detection on web traffic would be to use either Bagging TPMiner or Isolation Forest. Depending on the situation we would opt for one or the other; if speed is desired, then Isolation Forest would be the best option. On the other hand, if precision and recall are wanted, Bagging TPMiner would be the best choice.

- As mentioned before, although Bagging Random Miner had better results than Isolation Forest, it consumes an excessive amount of time for testing. Therefore, we consider it may be better to use Bagging TP-Miner, which provides very close results.

- By exploring further with a re-classification of the *human* visits, we found *67 hidden visits from bots*, and *209 that may also belong to them*. Such visits were responsible for *3,751* page views of the site. Some of those bots are crawlers from several search engines, but we could not identify the rest of them. However, they seem to have unusual numbers of page views and visit duration times, which may indicate the presence of bots indeed.

- Using a One-Class approach would be better in the cases when we do not have enough information about the bot class. It may be capable to evolve as new bots come to the site but it depends on the data we give the model for training.

Chapter 5

Mining Patterns To Support Web Traffic Analysis

Web Analytics reports provide useful information that reflects the success of any online business. Analyzing such data for a high volume of visits may become an overwhelming task, and therefore, marketing experts are in the need for better ways to support their data exploration. For instance, a common use case of web traffic analysis is visitor segmentation, i.e. apply a series of conditions in order to filter and segregate common groups of visitors. Some examples of common segments are: by demographics, segmentation by conversions (i.e. if a visitor accomplished some pre-defined business goal like purchasing or visiting certain web page) or even visitor segmentation by a custom series of properties (like the Operating System or Web Browser they are using), it depends on the needs of the business. The common way to do that segmentation task is manually, meaning that the marketing expert is the one that decides which attributes are going to be used to create the segment. Sometimes the segment target is known *a priori*, but there could also be the case that the expert could not see at a first-glance which attributes, combined together, may introduce a large visitor grouping, and therefore useful to design marketing strategies for such public. The previous situation is a great opportunity where Machine Learning algorithms can support such critical online industry tasks.

As part of an integral work on providing means to help the marketing expert, in this chapter, we introduce some examples of how Machine Learning, specifically Pattern Extraction algorithms like PBC4CIP [46], could help to conduct visitor segmentation and find common patterns present in website traffic. Contrasted with the time it could take to the marketing expert analyzing raw data by himself, the Pattern Extraction approach provides an automated way of doing so. Then, insights from those patterns will enable us to propose some new business strategies in order to, for example, try to improve user retention or increase online conversions. The patterns will also be useful to know possible problems the visitors are facing (e.g. slow performance of the website) or just to know the common properties they share in common, which later can be used to plan new business strategies.

Moreover, those patterns can be further used in the Visualization Tool we previously introduced in Chapter 3, in order to filter the website traffic and analyze it in detail.

The following sections will be distributed as follows. Section 5.1 describes some common concepts related to patterns which are going to be used in the rest of the chapter. Then, Section 5.2 explains how anyone could prepare a dataset in order to be able to perform pattern extraction on it. Finally, in Section 5.3 we show some use cases of using patterns to analyze visitor segments.

5.1 Description of a pattern

All patterns are in *Disjunctive Normal Form (DNF)* [32]:

$$Pattern_k = \bigwedge_{i=1}^{n} C_i$$

Where n will be at most the cardinality of the attributes set, and C_i has the form:

$$C_i = attribute\ OP\ value$$

Where $OP \in \{>, <, =, !=, \geq, \leq\}$.

Support	**Pattern**
0.38	$visitDuration > 175.50 \wedge clicks\_median > 0$

Table 5.1: Example of a series of patterns for the *human* class. It has a support of 0.38, meaning that 38% of the *human instances* share that common pattern, whereas *zero percent* of the opposite class (*bot*) is covered for that pattern. This pattern indicates that 38% of the *human* visitors were browsing the website for at least 175.50 seconds ($\sim$ 2.9 mins.), and their clicks median is greater than 0.

Support. It indicates how many items are covered by a pattern in proportion to the total amount of instances. It can be computed as:

$$support(Pattern_k, N) = \frac{count(Pattern_k)}{N_{class}}$$

Where $\boldsymbol{count(Pattern_k)}$ is the total amount of items covered by such pattern, and $\boldsymbol{N_{class}}$ is the total amount of items that belong to the *class* that the pattern is supporting.

Table 5.1 gives an example of a pattern for the *human class*. In this case, 38% of the objects belonging to the *human* class share the same pattern, and it does not include any object from the opposite class (bot). The pattern can be interpreted as follows: *38% of the human visitors browsed the website for at least 175.50 seconds ($\sim$ 2.9 mins.), and clicked at least once within each page.*

Because there are so many patterns mined for each use case, we present only the most relevant during a period of observation, i.e. the ones with the highest support, and that include relevant attributes, which will depend from case to case.

5.2 Dataset preparation

Let us describe how anyone could prepare a dataset in order to extract patterns of interest. Because we are going to use PBC4cip's miner [46], the requirement is to have a dataset labeled *into two classes*.

Visit	Country	Duration	Type	Referrer
1	mx	80	returning	direct
2	ru	20	new	google
3	mx	30	new	direct
4	usa	40	returning	twitter

Table 5.2: Dataset example before labeling it into two classes. *Duration: visit duration* in seconds. *Type: returning* if the user has already visited the website; *new* if it is a new visit.

Visit	Country	Duration	Type	Referrer	Label
1	mx	80	returning	direct	**high**
2	ru	20	new	google	**low**
3	mx	30	new	direct	**low**
4	usa	50	returning	direct	**high**

Table 5.3: Dataset labeled into two classes: *high* and *low*. *high* represents visits with a duration ≥ 40, and *low* visits with duration < 40.

Visit	Country	Duration	Type	Referrer	Label
1	mx	80	returning	direct	**direct**
2	ru	20	new	google	**social**
3	mx	30	new	direct	**direct**
4	usa	50	returning	direct	**social**

Table 5.4: Referrer labeling example. The *referrer* column was transformed into two-classes: *direct*, and *social* that groups *google* and *twitter* together.

Visit	Country	Duration	Type	Referrer	Label
1	mx	80	returning	direct	**mx**
2	ru	20	new	google	**ru**
3	mx	30	new	direct	**mx**

Table 5.5: Country comparison labeling. Mexican (*mx*) visitors against Russians (*ru*). The remaining instances (Visit 4) were deleted.

Table 5.2 shows a dataset which we are going to use as an example of how to transform it in order to be able to perform the pattern extraction.

The first step is to decide what we want to contrast. For example, let say we want to extract patterns of interest from visitors who spend more time than others browsing in the website. So, we must define two possible labels: *low, and high*. Then, we shall decide how to assign the labels to the dataset instances. The marketing expert would be responsible to take that decision depending on the current business metrics. As an example, we are going to decide that all instances with a *visit duration* < 40 belong to *low*, and to *high* those with the *visit duration* ≥ 40.

Once we have defined the labeling procedure, we shall create a new column called *Label* and assign the respective instance values, thus, converting everything into a two-class dataset. Table 5.3 shows how the dataset would end up after performing such change.

To mention a few more examples, if we would like to contrast Mexican visitors against the rest of the world, in that case we must have $Label =$

$\{mx, world\}$, or if we want to contrast Mexican visitors against another country, Russia for example, we shall delete the instances that does not belong to those countries and just assign the country value as label.

Tables 5.4 and 5.5 are other examples of how the original dataset could be modified for different situations of interest. As it can be seen, the main requirement is that the *Label* column must have only two possible values. If we consider $Label$ as a set, then $|Label| = 2$ must hold.

5.3 Use Cases

Let us describe some use cases of pattern extraction on web traffic data. All the patterns were mined from the data we personally collected as described in Chapter 2. For each one of the elaborated use cases, we contrast a specific attribute against another and extract patterns upon that. For example, we are going to contrast *bot visits versus human visits*; *visitors based on country of origin; visitors who completed certain business goal versus the ones who did not, and finally recurrent visitors versus new ones*. We already talked about the importance of Non-Human Traffic (NHT) and how it affects web analytics reports, that is why we used it as the first example. The remaining use cases are extracted from common analysis done in web analytics.

In the following examples, the pattern lists are sorted by the class support. Each case contains a series of patterns for a specific class where the support for the opposite class is zero. That means that every single pattern covers a certain percentage of a specific class and covers zero percent of the opposite class.

5.3.1 Case 1: Humans vs Bots

In this example, we make a comparison of the two classes we have: *human and bot*. Such classes were known *a priori* by the experiments we already discussed in Chapter 2. We can see some similarities with the J48 trees seen in Chapter 4, which are an example of how also Decision Trees can be used for data interpretation. Nevertheless, in this case with *pattern extraction* we can get more detailed and useful information. Let us analyze the extracted patterns from this use case.

#	Support	Patterns
1	0.44	plugin_pdf = “1”
2	0.42	visitDuration $>$ 101.00 $\wedge$ operatingSystemVersion $\neq$ “7” $\wedge$ visitIp $\neq$ “103.65.30.0” $\wedge$ resolution $\neq$ “1280x720”
3	0.40	clicks $>$ 6.00
4	0.38	visitDuration $>$ 175.50 $\wedge$ clicks_median $>$ 0.25
5	0.37	browserVersion $>$ 57.50 $\wedge$ browserName $\neq$ “Coc Coc” $\wedge$ cpu_cores $>$ 3.00
6	0.36	plugin_flash = “0” $\wedge$ visitDuration $>$ 80.50 $\wedge$ browserName $\neq$ “Internet Explorer”
7	0.21	plugin_flash = “0” $\wedge$ timeSpent_mean $>$ 12.54
8	0.16	deviceBrand $\neq$ “Desconocido” $\wedge$ browserCode $\neq$ “IE”

Table 5.6: Patterns for the *human* class. The first column is a counter to reference a pattern within this table. The second column indicates the support for the *human* class. Third column describes the pattern in DNF.

Pattern #1. For the human class, we have that 44% of the instances are covered by the pattern `plugin_pdf = 1` (Pattern $\#1$). This fact may be intuitive because for us, as humans, usually it is often common to open multimedia files in the browser, for example PDF documents, so it is highly probable that we have a PDF plugin installed. However, for bots that may not be the case, because they might not even care about such things, they just open web browsers programmatically in order to accomplish some task.

Patterns #3 and #4. There are another two patterns of interest, $\#3$ and $\#4$. The third pattern is telling us that 40% of the human visitors are clicking at least six times during their visits. This is very useful because it can help to the page builder and the marketing team to include some kind of interactive content, ensuring them that at least 40% of the visitors are going to make around that number of clicks.

On the other hand, the fourth pattern is telling us that 38% of the human traffic are browsing the site for at least two minutes and their median of clicks is above 0.25. Surprisingly, it is a very high value for the visit duration, which means that the content may be good at engaging the users. Regarding the clicks median, it does not give us much information but, because the clicks attribute is a discrete value, we can interpret it as that at least one click is made.

#	Support	Patterns
1	0.62	plugin_pdf = "0" $\wedge$ visitIp $\neq$ "148.241.2.200" $\wedge$ deviceBrand = "Desconocido" $\wedge$ pageviews $\leq$ 11.00 $\wedge$ generationTime_median $>$ 293.25 $\wedge$ visitDuration $\leq$ 566.00 $\wedge$ visitDuration $>$ 4.50
2	0.62	plugin_pdf = "0" $\wedge$ visitIp $\neq$ "148.241.2.200" $\wedge$ visitIp $\neq$ "10.18.1.0" $\wedge$ browserName $\neq$ "Chrome Mobile" $\wedge$ pageviews $\leq$ 12.50 $\wedge$ visitIp $\neq$ "189.210.60.0" $\wedge$ generationTime_median $>$ 293.25 $\wedge$ generationTime_mean $>$ 335.17 $\wedge$ visitIp $\neq$ "187.198.154.0"
3	0.62	pageviews $\leq$ 10.50 $\wedge$ browserVersion $\leq$ 57.50 $\wedge$ plugin_pdf = "0" $\wedge$ visitIp $\neq$ "148.241.2.200" $\wedge$ plugins $>$ 1.50 $\wedge$ visitIp $\neq$ "187.200.192.0" $\wedge$ deviceBrand $\neq$ "Samsung" $\wedge$ generationTime_median $>$ 293.25 $\wedge$ visitDuration $>$ 0.50
4	0.60	plugin_pdf = "0" $\wedge$ visitIp $\neq$ "148.241.2.200" $\wedge$ deviceBrand = "Desconocido" $\wedge$ generationTime_mean $>$ 335.17 $\wedge$ visitIp $\neq$ "10.18.1.0" $\wedge$ visitIp $\neq$ "187.198.154.0" $\wedge$ resolution $\neq$ "1920x1080"

Table 5.7: Patterns for the *bots* class. The first column is a counter to reference a pattern within this table. The second column indicates the support for the *bots* class. Third column describes the pattern in DNF.

Looking at the patterns for the bot class (Table 5.7) it is clear that they seem more *complex*, meaning that they include more attributes in every pattern. The first insight we can notice is that there are many patterns with higher support than the human ones; we have patterns with support greater than 50%, whereas for the human class the maximum support was 44%. Also, if we look carefully, there are items in the patterns that repeat across them, for example `plugin_pdf = 0`, `generationTime_median > 293.25`, `visitIp` $\neq$ `'148.241.2.200'` or `resolution != '1920x1080'`. As we discussed before, it was expected that they did not have a PDF plugin installed. However, for the *generationTime* attribute, which represents the amount of time in seconds that the visitor's browser took to load a web page, it seems that they are taking too much time to load a page. Referring to that IP address (148.241.2.200), we white-listed such value because we know from *a priori* that it belongs to a computer we used for testing, so it makes sense to appear in the pattern as an inequality comparison.

5.3.2 Case 2: Business KPI (conversion achieved)

Let us define two groups of users: the ones that have completed a business goal we previously defined, and the ones that did not complete the goal. In marketing, these concepts of *goal or objective* are commonly used interchangeably and often used as a Key Performance Indicators (KPI) of how well the business is performing. An example of an objective could be: when a user fills out a form in the contact page, when the visitor purchases something, or even when a simple page is visited.

The event of a visitor accomplishing a goal is named a *conversion*. Thus, conversion analysis is very important in any business strategy because it provides metrics and mechanisms to measure the success of a campaign, such as getting a *conversion rate* value to optimize and track the *Return on Investment (ROI)* and, in the use case of advertising, create a better bidding strategy. The *Conversion Rate* is one of the most used metrics and is defined as a percentage of $\frac{goals}{total\ visits}$, where *goals* is the number of visitors that completed the goal.

In this use case, we took as goal visiting a specific web page [1] which, in the context of the data we are analyzing, is the one that provides the functionality for buying new internet domain names. So, if a user visited such page, we classified that instance with the class `completed` and `non-completed` otherwise. From 619 visits to that website, only 49 visited such web page, thus we have a *Conversion Rate* of 7.92%. Depending on the industry, there are indicators and studies [35] of what could be a good conversion rate, nevertheless, a common characteristic is that all use to be low values, around $1-5\%$. For example, in e-commerce, a good conversion rate would be around 1.9% or for technology services around 3%. Given this situation, the conversion rate we have for that goal looks promising.

[1]URI: domain_name_recordb.jsf

#	**Support**	**Patterns**
1	0.22	clicks $>$ 4.50 $\wedge$ operatingSystemCode = "WIN" $\wedge$ visitDuration $\leq$ 586.50 $\wedge$ generationTime_mean $\leq$ 1053.25
2	0.22	operatingSystemVersion $\neq$ "9.3" $\wedge$ kind = "human" $\wedge$ clicks $>$ 3.50 $\wedge$ timeSpent_mean $\leq$ 26.93 $\wedge$ plugin_director = "0" $\wedge$ pageviews $\leq$ 20.50
3	0.20	deviceType = "Escritorio" $\wedge$ kind = "human" $\wedge$ clicks $>$ 2.50 $\wedge$ visitDuration $\leq$ 586.50 $\wedge$ timeSpent_mean $\leq$ 13.01
4	0.16	visitDuration $>$ 32.50 $\wedge$ kind = "human" $\wedge$ timeSpent_mean $\leq$ 8.06 $\wedge$ generationTime_median $>$ 98.25
5	0.16	generationTime_median $\leq$ 407.75 $\wedge$ timeSpent_mean $\leq$ 8.57 $\wedge$ plugins $\leq$ 3.50 $\wedge$ clicks $>$ 1.50 $\wedge$ visitDuration $>$ 75.50 $\wedge$ pageviews $\leq$ 20.50
6	0.14	timeSpent_mean $>$ 2.38 $\wedge$ kind = "bot" $\wedge$ timeSpent_mean $\leq$ 9.82 $\wedge$ resolution = "1366x768" $\wedge$ continentCode $\neq$ "ams" $\wedge$ browserCode $\neq$ "SE" $\wedge$ clicks $>$ 2.50 $\wedge$ operatingSystemVersion $\neq$ "Vista" $\wedge$ visitDuration $\leq$ 48.50 $\wedge$ pageviews $>$ 2.50

Table 5.8: Patterns for the *completed* class. The first column is a counter to reference a pattern within this table. The second column indicates the support for the *completed* class. Third column describes the pattern in DNF.

Let us now analyze the patterns found from the visitors that accomplished the objective and the ones that did not. Tables 5.8 and 5.9 shows some of the patterns extracted by the miner.

#	Support	Patterns
1	0.48	**pageviews $\leq$ 2.50** $\wedge$ **visitDuration $\leq$ 35.50** $\wedge$ generationTime_median $>$ 184.75
2	0.45	pageviews $\leq$ 2.50 $\wedge$ resolution $\neq$ "1366x768"
3	0.43	plugins $\leq$ 2.50 $\wedge$ plugin_pdf = "0" $\wedge$ pageviews $\leq$ 2.50 $\wedge$ visitDuration $\leq$ 35.50 $\wedge$ continentCode $\neq$ "amn"
4	0.42	clicks $\leq$ 1.50 $\wedge$ **generationTime_mean** $\leq$ 2018.50 $\wedge$ visitDuration $\leq$ 48.50 $\wedge$ continentCode $\neq$ "amn"
5	0.40	resolution $\neq$ "1366x768" $\wedge$ clicks $\leq$ 1.50
6	0.39	plugin_silverlight = "0" $\wedge$ clicks $\leq$ 1.50 $\wedge$ **generationTime_median** $\leq$ 2018.50 $\wedge$ visitDuration $\leq$ 48.50 $\wedge$ timeSpent_mean $>$ 3.29
7	0.39	pageviews $\leq$ 2.50 $\wedge$ continentCode $\neq$ "eur" $\wedge$ **generationTime_mean $>$ 184.75**
8	0.36	pageviews $\leq$ 2.50 $\wedge$ timeSpent_mean $\leq$ 7.18 $\wedge$ **generationTime_median $>$ 184.75**
9	0.36	visitDuration $\leq$ 32.50 $\wedge$ timeSpent_mean $>$ 3.29
10	0.35	keys $\leq$ 0.50 $\wedge$ visitDuration $\leq$ 35.50 $\wedge$ timeSpent_mean $>$ 4.06
11	0.24	clicks $\leq$ 1.50 $\wedge$ pageviews $>$ 1.50 $\wedge$ **generationTime_median $\leq$ 890.00** $\wedge$ visitDuration $>$ 14.50
12	0.24	pageviews $\leq$ 1.50

Table 5.9: Patterns for the *non-completed* class. The first column is a counter to reference a pattern within this table. The second column indicates the support for the class. Third column describes the pattern in DNF.

We are more interested in the *non-completed* class due to its potential in converting into a *completed* class, i.e. what can we learn from the users that do not make any conversions in order to increase the conversion rate.

Pattern $\#1$**.** The first pattern of the *no-completed* class gives us interesting information, it says that 48% of the visitors are viewing at most two pages within a visit duration less or equal than 35 seconds. Furthermore, all of them do not have installed the *silverlight* plugin and the median of time taken to render the web pages is more than three minutes. How can a business strategy be implemented with this information? For example, first of all, it is worth to highlight that this pattern covers almost the half of the instances, so we can assume that at least we are reaching a lot of the visitors. Second, because we know that those users are not going through more than two page-views within 35 seconds, we can implement *call to action* elements since the first moment the user reaches the website. The first two visited pages are critical (*pageviews* ≤ 2.50) to make a conversion.

With the previous information, we could track (using analytics services) the amount of time the user has being browsing the site, and then, for example, trigger an event at the 30th second of the visit time that launches a promotion or something that asks for the user attention, trying to keep the visitor on the website.

Another interesting insight from this pattern is an *item* shared with some other patterns: `generationTime_median > 184.75`. We can find this item and several others with higher values (352.00, 407.75) in many of the extracted patterns. A hypothesis can be made referring that the website is not performing well; it is taking too much time to load the pages (*in the order of minutes!*) and that is impacting negatively the user experience and thus the conversion rate. What can be done to test this hypothesis? This is the perfect scenario to implement something commonly used in marketing called: *A/B testing* [66]. It helps to test hypothesis and compare two scenarios, for example, if the business changes the welcome page banner, could this increase user retention? To test that, two versions of the website are created and measured independently; one with the old welcome banner; the second, with the new welcome banner. At the end of a time period results are compared to test the hypothesis rejection.

Moreover, because of the insight that the website has slow loading page times, it is the perfect opportunity to implement *Accelerated Mobile Pages (AMP)* [61], quite popular this days and business are adopting it. Then, we can implement an A/B test for these two scenarios: pages with AMP and pages without it. After a period of time, we could even compare how the *generationTime_median* is evolving and check if it was affecting variables like the *visit duration and pageviews*. The previous seems to be a very good strategy worth to try, and we got it thanks to the insights from the extracted patterns.

5.3.3 Case 3: User segmentation by country

#	Support	Patterns
1	0.56	generationTime_median $\leq$ 191.50 $\wedge$ cpu_cores $>$ 1.50 $\wedge$ visitIp $\neq$ "185.118.167.0"
2	0.56	generationTime_median $\leq$ 191.50 $\wedge$ referrerType $\neq$ "search" $\wedge$ resolution $\neq$ "1280x800" $\wedge$ visitIp $\neq$ "84.53.247.0"
3	0.50	**visitDuration $>$ 101.00**
4	0.44	browserVersion $>$ 57.50 $\wedge$ referrerType $\neq$ "search"
5	0.42	deviceBrand = "Desconocido" $\wedge$ generationTime_mean $\leq$ 205.33 $\wedge$ cpu_cores $>$ 3.00 $\wedge$ generationTime_median $\leq$ 220.00
6	0.13	**visitDuration $\leq$ 5.50**
#	**Support**	**Patterns for Russia's class**
7	0.76	pageviews $\leq$ 7.50 $\wedge$ visitDuration $>$ 5.50 $\wedge$ visitIp $\neq$ "148.241.2.200" $\wedge$ operatingSystemCode $\neq$ " $\wedge$ " $\wedge$ generationTime_mean $>$ 286.65 $\wedge$ plugin_flash = "1" $\wedge$ generationTime_median $>$ 230.00 $\wedge$ visitIp $\neq$ "189.182.124.0"
8	0.76	generationTime_mean $>$ 195.00 $\wedge$ clicks $\leq$ 3.50 $\wedge$ generationTime_median $>$ 182.00 $\wedge$ resolution $\neq$ "1080x1920" $\wedge$ timeSpent_median $\leq$ 5.75 $\wedge$ visitorType = "new" $\wedge$ referrerType = "direct" $\wedge$ visitDuration $>$ 5.00 $\wedge$ resolution $\neq$ "1920x1080" $\wedge$ visitIp $\neq$ "187.163.86.0"
9	0.60	generationTime_median $>$ 191.50 $\wedge$ browserVersion $\leq$ 57.50 $\wedge$ clicks $\leq$ 6.00 $\wedge$ deviceBrand = "Desconocido" $\wedge$ operatingSystemVersion $\neq$ "XP" $\wedge$ visitDuration $>$ 5.00 $\wedge$ resolution $\neq$ "1920x1080" $\wedge$ visitIp $\neq$ "187.163.86.0"

Table 5.10: Patterns for the *mx (Mexico)* and *ru (Russia)* classes. The first column is a counter to reference a pattern within this table. The second column indicates the support for the class. Third column describes the pattern in DNF.

In Table 5.10, we show the patterns found when contrasting Mexican visitors versus Russian visitors. Mexican's patterns seem to be easily interpreted, whereas Russian's patterns look more complex.

Patterns #3 and #6 for the Mexican class.

First, pattern #3 is telling us that 50% of the visitors have a visit duration of at least 101 seconds (∼1.6 mins.), which is great for any website. Next, pattern #6 is telling us that 13% of the visitors have a visit duration of 5.50 seconds at most. Those kind of patterns with a single element are very useful because we can get extra information from them. For example, in this case, we can conclude that the rest 37% of the visitors have a visit duration between 5.50 and 101 seconds. A possible action in the business could be to try to reduce such 13% of visitors with a visit duration of 5.50 seconds at most, as less as possible. If a product was just launched to the Mexican market, there is an initial not-insignificant gap on your audience.

Getting more information from patterns with one item...
Because we are using contrast patterns in such a way that a pattern cannot cover any instance from the opposite class, *i.e. the support is zero*, we can use it in our advantage to get more information from the data.

For example, Pattern #3 tells that 50% of Mexican visitors have a visit duration of at least *101 seconds*. Because the support for the opposite class is zero, such pattern is not covering any instance from the Russian visitors, meaning that Russians browse the website and stay there at most 101 seconds. It is like a complement to the Mexicans' pattern.

We can conclude from the previous analysis that *Mexican visitors spend more time than Russian visitors on the website*. Again, this can be actionable information, if the business is interested in the Russian market, then it can be defined the visit duration of Russians as KPI and profile it as a goal upon that.

5.3.4 Case 4: Recurrent visitors

This use case may come very handy to understand how *first-visits* behave. The experiment segments the visitors into two classes: *new* (first-visits) and *returning* (visitors who already visited the site at least once). Let us first analyze the first-visits; patterns extracted for that class are shown in Table 5.11.

#	Support	Patterns
1	0.26	timeSpent_median > 1.75 ∧ clicks > 0.50 ∧ **visitDuration > 12.50** ∧ browserFamily ≠ "Gecko"
2	0.15	generationTime_median > 233.50 ∧ visitDuration > 16.50 ∧ visitDuration ≤ 27.50
3	0.15	referrerType = "direct" ∧ clicks_median > 0.75 ∧ **generationTime_median > 86.00** ∧ continentCode = "eur"
4	0.14	**generationTime_median > 145.50** ∧ clicks_median ≤ 0.75 ∧ generationTime_median ≤ 290.50
5	0.14	**generationTime_median > 233.50** ∧ visitDuration ≤ 61.50 ∧ visitDuration > 27.50 ∧ cpu_cores ≤ 3.50
6	0.11	cpu_cores > 2.50 ∧ **generationTime_median > 146.00** ∧ visitDuration > 52.50
7	0.11	timeSpent_median ≤ 1.75 ∧ **generationTime_median > 129.00** ∧ generationTime_median ≤ 233.50
8	0.10	**generationTime_median > 145.50** ∧ timeSpent_median > 7.75
9	0.09	**generationTime_median > 233.50** ∧ deviceType = "Escritorio" ∧ visitDuration ≤ 7.50
10	0.08	referrerUrl ≠ "https://outlook.live.com/" ∧ pageviews > 4.50 ∧ **cpu_cores > 2.50**

Table 5.11: Patterns for the *new* class. The first column is a counter to reference a pattern within this table. The second column indicates the support for the class. Third column describes the pattern in DNF.

Patterns for the *new* class.

Pattern #1 . It is telling that 26% of the new incoming visitors, spend more than 12.50 seconds browsing, they click at least once during all their visit, they are not using a web browser based on Gecko (commonly Firefox), and they spend a median of 1.75 seconds on each page. What can be most useful from the previous information is the fact that as marking strategy, every call-to-action element should appear before the 2^{nd} second on the web page and that we have at least 12.50 seconds before the user leaves the website in order to make a conversion.

Pattern $\#2$**.** It is telling us that 15% of the new visitors are getting a page's generation time median greater than 233.50 seconds, which seems very long! plus, they browse for a period of 16.5 to 27.5 seconds during their visit.

What seems problematic for this and the rest of the patterns $(\#3 - \#9)$ is the large value for the *generation time* attribute, which is very long for a page to be delivered.

Pattern $\#3$**..** This pattern has relevant information for potential new visitors segments; 15% of new visits are coming from Europe, they are accessing directly to the website (without a referrer), they click at least once on the pages, and the generation time of each page is around 86 seconds or greater, which is not bad at all. With this information, we can specifically target an audience and we know in advance a little bit of their behavior on the website. A business strategy can be implemented to convert these new visitors into recurrent ones, for example, offering free shipping to European countries or promoting specific local products. Either way, the marking team would have sufficient information and plan how to keep engaging users bit by bit, finding new market niches.

Pattern $\#10$**.** Although it only has 8% of support, the complementary representation of this pattern is interesting. It implies that the returning visitors that are coming from that specific referrer (Outlook), have few *page views* (≤ 4) and their computers have at most 2 CPU Cores. That last part of the CPU cores may be insignificant; what is worth to rescue is the fact that the visitors coming from their email service (Outlook) seem to have few page views. So, what can be done? It could imply that email marketing is not engaging that much. Thus, an improvement on the email campaigns can be suggested to the business.

Patterns for the *returning* class.

#	Support	Patterns
1	0.10	pageviews $\leq$ 7.50 $\wedge$ **generationTime_median $\leq$ 29.00** $\wedge$ cpu_cores $>$ 6.00 $\wedge$ **generationTime_median $>$ 1.50**
2	0.08	visitDuration $>$ 828.00 $\wedge$ clicks $\leq$ 6.00 $\wedge$ timeSpent_median $>$ 0.50
3	0.08	referrerName $\neq$ "Yandex" $\wedge$ **generationTime_median $\leq$ 148.50** $\wedge$ visitDuration $\leq$ 191.00
4	0.08	**generationTime_median $>$ 145.50** $\wedge$ operatingSystemCode = "UBT" $\wedge$ browserCode = "FF" $\wedge$ pageviews $>$ 3.00

Table 5.12: Patterns for the *returning* class. The first column is a counter to reference a pattern within this table. The second column indicates the support for the class. Third column describes the pattern in DNF.

Table 5.12 shows few of the patterns mined for the *returning* class. Unfortunately, most of all did not have high support, compared with those of the *new* class. However, pattern #1 gives us a hint of what may be a cause for visitors returning to the website: at least 10% of the returning traffic have *generation times* between 1.5 and 29 seconds, contrasted with the higher times of the opposite class.

5.4 Conclusions

We have shown in the previous sections how mining patterns from Web Analytics data can be used to plan business strategies by observing the visitors behavior. The patterns also serve to get an insight into what users share in common or do not at all. By contrasting the patterns from one class against the other with zero support, we can extract complementary information from single item patterns (as we did in Section 5.3.3). Those kind of patterns (with a single item and high support) are very useful because they describe a wide variety of instances, letting us characterize objects in a much better interpretable fashion.

Regarding the analysis of the use cases, when we compared human versus bot visits (Section 5.3.1), we were able to find out that bots do not use to have installed plugins in their browsers which are common to many people, like the PDF plugin.

In the second use case, Section 5.3.2, we explored how *conversions* can be characterized by patterns. We were able to know that, for the visitors who did not complete any business goal, 48% of them are viewing at most two pages within a visit duration less or equal than 35 seconds. A business strategy to implement call-to-action elements was proposed given that fact. Subsequently, it was found that many patterns included higher values of web page generation times. Because of that, a site optimization strategy was proposed: the implementation of AMP (Accelerated Mobile Pages) on the site.

When analyzing patterns for the third case, a comparison between Mexican and Russian visitors (Section 5.3.3), the patterns led us to conclude that *Mexican visitors spend more time than Russian visitors on the website*.

Finally, for the use case of new visitors and returning ones (Section 5.3.4), we observed how the complement of patterns can be used to infer extra information. In that case, we used pattern $\#6$, which covers 13% of the visits, to conclude that 37% more visitors have a visit duration between 5.50 and 101 seconds. Upon that, some business actions were proposed.

Chapter 6

Conclusions and Future Work

Web Analytics has become very important in the past few years. That may be the case why new job roles have been arising on the industry for that specific domain, like Community Managers, Influencers or Social Network Experts. Being capable of understanding web traffic and explore it in more detail has become a key part of any web analytics report. Thus, the importance of having tools that help with their tasks is crucial for any online business.

In this work, we introduced a new visualization tool for Web Analytics (Chapter 3). It allows to explore in a new way: website traffic, site goal performance, individual navigation paths, and multi-site traffic connection; without the intrusion of requiring access to each individual client computer or installing TCP packet extractors. We used existing open source software for that, specifically a platform called Matomo [1], and build the visualization tool on top of that. Three main contributions from the visualization tool can be noticed. *First*, it includes a new way to visualize in a single report, visit metrics, page metrics, page goal performance, and interactions of visitors to the website pages. *Second*, the tool integrates easily with patterns mined using Machine Learning algorithms. The exploration of such patterns using the tool can lead us to implement new marketing strategies or to get insights of what is going on with the website traffic, such as detecting potential customer segments, or to contrast visitors between different countries, among other use cases. *Third*, a different approach to visualize the navigation path for a visitor is presented. All the tools in the market use simple tables to represent how a single user was browsing the website (Chapter 3, Section 3.4). We believe that could be improved. We created a visualization for such step-by-step browsing, using a network diagram which includes how much time the user spends on each page and the order of visit, plus the moment in time where the visitor completed certain business goal.

To our knowledge, there is no tool in the market that integrates all of the previous features mentioned. We have validated the tool's usefulness with domain experts from a Mexican company that provides internet services, and they validated the good potential it has for performing marketing reports and analysis.

Another important aspect that must be taken with care is the analysis of how reliable is the information that such tools are providing, given the fact that bot traffic is daily present on the Internet [78].

Facing the problem of Sophisticated Invalid Traffic (SIVT) detection, we have successfully created a one-class classifier to identify abnormal traffic (Chapter 4). We used web server logs that contained around 7.4 million of requests, to create a new dataset. Such new dataset was pre-processed in order to group multiple requests together and thus form visit instances. A visit contains many requests, and each request could mean an access to a certain web page or the request of any static file from the server. We tested several one-class classifiers that can be found in the literature [33, 24, 12, 50, 14, 43]. In this research, the best performance obtained was with *Bagging TP Miner*. By using it, we were able to perform the anomaly detection of bot traffic achieving a performance, measured by the *ROC Area*, of 0.9823. To prevent the model from *over-fitting*, experiments were conducted with a 5-fold Cross Validation, using an inverted strategy of *training* with *1-fold* and *testing* with *k-1 folds*, *i.e.* 1 fold was used for training and 4 for testing.

With the support of such bot traffic detector, we could be able to audit ad campaigns to improve the reliability of performance metrics. Thus, detecting bot traffic could enable us to compare advertising reports from commercial platforms, and in the case of metric mismatches, argue for refunds to the platform by arguing that many of the reported traffic belonged to bots that they were unable to detect.

Consequently, we showed in Chapter 5, how mining patterns from Web Analytics data can be used to plan online business strategies by observing the visitors behavior. Patterns also serve to get insight into what users share in common or do not share in common at all. By contrasting the patterns from one class against the other with zero support, we can obtain complementary information from singled item patterns. That kind of patterns, the ones with a single item and high support, are very useful because they describe a wide variety of instances, letting us characterize objects in a much more interpretable manner.

In summation, we contributed with use cases of how mining patterns from web traffic can lead us to implement online content strategies. For the first case, when we compared *humans versus bots* visits (Section 5.3.1), we were able to find out that bots do not use to have installed plugins in their browsers which are common to many people, like the PDF plugin. in addition, the second use case examined how current conversions can be characterized by patterns (Section 5.3.2). We were able to know visitors' critical browsing times and with such information propose the implementation of call-to-action elements to lead a conversion. Subsequently, it was found that many patterns included higher values of web page's *generation times*. Because of that, a site optimization strategy was proposed: the implementation of AMP (Accelerated Mobile Pages) on the site. More information was found in further patterns, like concluding that *Mexican visitors spend more time than Russian visitors on the website* (Section 5.3.3). Finally, for the use case of *new visitors and returning ones* (Section 5.3.4), we observed how complementary patterns can be used to infer extra information. We observed that there was a potential customer segment on European countries, and therefore, strategies, like offering free shipping or selling local products, could be used to lead to a conversion. Additionally, we were able to find out that an optimization to email marketing campaigns could improve user retention.

This work has presented an exploration of web analytics, a visualization tool for that kind of data, a Machine Learning approach to identify abnormal traffic, and finally, a demonstration of how mining patterns from users traffic can help to improve a website configuration and implement several marketing strategies. We believe there is a lot of opportunity in this field to keep exploring further. We address some ideas of what can be done in the next section.

6.1 Future Work

6.1.1 Graph mining from visits to web pages

We have seen in Chapter 3 how it is possible to represent as a directed graph the navigation path of a user. Within such graph there could be nodes representing conversions, i.e. pages we desired to get visited. It would be useful to know how many steps a user takes before reaching a page goal and it would be impossible to analyze visit by visit by hand. So, it may be a good idea to use graph mining [4] for getting aggregated metrics from the data. That could tell us what is the smallest number of steps that users are taking to achieve certain goal page.

Another application of graph mining could be to extract common connected-components [4]. Every connected-component could have assigned a support, indicating how many visits share it. We can give many usages to that common patterns. For example, if we know that certain number of visitors are following the same navigation path, then we can include *call-to-action elements* like *ads, buttons, banners, etc.*, in-between their paths to lead a conversion.

In the same way, an interesting application could be the use of *Markov Chains* [31] on that series of connected components in order to know *how probable would be that the user makes a conversion, given that he/she is visiting certain web pages*.

6.1.2 Aggregated traffic information

So far, among other things, the visualization tool allows to analyze independent navigation paths, i.e. explore for a single visit what pages the user visited, in which order, and if a conversion was made. However, what if we wanted to visualize common navigation paths, for example, the same as the ones extracted in the previous graph mining proposal. Visualizing that kind of information could easily help the marketing team to detect critical nodes (web pages that are common in several browsing paths) that are being visited by a large number of users. It would be also possible to see at which step of the navigation path the user starts losing interest and leave the website.

By knowing the previous information, a site restructure could be implemented or even *A/B* testing experiments can be suggested. Website with a large number of visits would be benefited from the aggregated visitor information. Precaution should be taken for aggregated metric reports, though. It is easy to be fooled by aggregated numbers. For example, it is not recommended to use averaged metrics, because they may not represent entirely your visitors. Imagine this situation, let us say we have 5 visitors with a visit duration of *200, 2, 2, 3, and 3* seconds, respectively. An *average visit duration* would indicate that in average, each visitor browses the website for around *41.8 seconds*, however, that is not *true*. The metric becomes affected by the distribution of the attribute we are measuring, which is often skewed. A *correct* metric for the *visit duration* should be that most of the users have a visit duration around *2-3 seconds*.

A *good rule of thumb* is not to use averaged metrics for global content. It is recommended to get metrics for specific parts of the website, and not to compare everything with aggregated values like the ones for the index page.

6.1.3 Improve Pattern Extraction and Visitor Segmentation

In Chapter 5 we introduced how to use *Pattern Extraction* algorithms to support the analysis of visitor segments. Although the pattern extraction is done automatically by the miner, the preparation of the dataset needs the human intervention. The selection of the segments by itself requires a selection of certain attributes or values to re-label the dataset. The selection could be guided either by the marketing expert (as we did in this research), but it would be useful to do it automatically.

The brute-force method to do it would represent a combinatorial problem, where pattern extraction can be done automatically for all the possible tuples of attributes. However, for attributes with more than two categorical values, several extra tuples need to be generated in a strategy of one-versus-all (because the current miner only works with two classes). Nevertheless, a different miner could be used to solve the latter issue.

Another approach could be to reduce the set of possible attributes to choose. We could use algorithms that compute feature importance and select only the ones that surpass a certain threshold.

We see that the main problem is to select which attributes to contrast. A further exploration using Machine Learning algorithms like Bayesian Networks or Association Rules could also provide hints on what attributes to choose. However, in the end, the pattern extraction is a process that follows after a decision by the marketing expert; it is done once a certain segment of interest is selected.

6.1.4 Visualization Tool Improvements

The current state of the visualization tool can be improved in several ways. Initially, it was not developed with a lot of user-tuning parameters, thus, the inclusion of a more sophisticated settings screen could be useful for the end-user. For example, in the case of the visits view, a slider can be used to change the width of the edges that connect pages and visits. The value would change with respect to the selected metric (e.g. visit duration) and therefore patterns like the ones we observed for bots (thicker edges) could arise.

Another area for improvement could be in the Query Console provided by the visualization tool. Since we are using Cytoscape as the main library to query the graph information, we are bounded to the API it provides, among them the syntax of the queries. Such syntax is not a standard representation, for example, it is not using the Conjunctive/Disjunctive Normal Form (CNF/DNF) that is used by common Pattern Extraction algorithms. Therefore, supporting a DNF syntax in the Query Console could provide a much better user experience for user. Moreover, as the user types, an automatically suggestion of attributes could be implemented.

6.1.5 Iterative OCC Improvement

As we did in Section 4.5, where we were able to detect hidden bots inside human traffic, we can repeat that experiment in a periodic way and therefore continuously improving the One-Class models. Such training, however, should be performed carefully because if we have again hidden bot traffic and we train the model with it, in the long term the OCC may take it as normal behavior and therefore always missing it.

Bibliography

[1] Matomo - Open Analytics Platform.

[2] URL Blacklist.

[3] ACARALI, D., RAJARAJAN, M., KOMNINOS, N., AND HERWONO, I. Survey of approaches and features for the identification of HTTP-based botnet traffic. *Journal of Network and Computer Applications 76*, September (2016), 1–15.

[4] AGGARWAL, C., AND WANG, H. *Managing and mining graph data*, vol. 40. 2010.

[5] AKAMAI. Real-time internet monitor akamai.

[6] ALEXA. Alexa Top 500 Global Sites, 2015.

[7] ANA, AND WHITE OPS, I. Bot baseline. fraud in digital advertising. Tech. Rep. January, WhiteOps/ANA, 2016.

[8] ATIENZA, D., HERRERO, Á., AND CORCHADO, E. Neural analysis of http traffic for web attack detection. In *Adv. Intell. Syst. Comput.* (2015), vol. 369, pp. 201–212.

[9] BEN-ARI, M. *Mathematical Logic for Computer Science*. Springer, 2012.

[10] BERRAR, D. Random forests for the detection of click fraud in online mobile advertising. ... *Workshop on Fraud Detection in Mobile Advertising (... 1* (2012), 1–10.

[11] BLUE, R., DUNNE, C., FUCHS, A., KING, K., AND SCHULMAN, A. Visualizing real-time network resource usage bt - visualization for computer .... *Vis. Comput. ... 5210*, Chapter 12 (2008), 119–135.

[12] BREUNIG, M. M., KRIEGEL, H.-P., NG, R. T., AND SANDER, J. LOF. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data - SIGMOD '00* (New York, New York, USA, 2000), vol. 29, ACM Press, pp. 93–104.

[13] BRIAN PUGH. Battling bots: comscore's ongoing efforts to detect and remove non-human traffic, Nov. 2012.

[14] CAMIÑA, J. B., MEDINA-PÉREZ, M. A., MONROY, R., LOYOLA-GONZÁLEZ, O., VILLANUEVA, L. A. P., AND GURROLA, L. C. G. Bagging-RandomMiner: a one-class classifier for file access-based masquerade detection. *Machine Vision and Applications* (jul 2018), 1–16.

[15] CASSETTO, O. Banishing bad bots with incapsula, Dec. 2014.

[16] CHAPELLE, O., AND CRITEO. Kaggle Display Advertising Challenge Dataset, 2014.

[17] CHI, E. H. Improving web usability through visualization. *IEEE Internet Comput. 6*, 2 (2002), 64–71.

[18] CIESLAK, D. A., HOENS, T. R., CHAWLA, N. V., AND KEGELMEYER, W. P. Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery 24*, 1 (2012), 136–158.

[19] COMSCORE. comScore: Measure what matters to make cross-platform audiences and advertising more valuable.

[20] COMSCORE. Invalid traffic, 2016.

[21] DFRWS. Dfrws 2009 forensics challenge challenge data and submission details.

[22] DISTIL NETWORKS. Digital ad fraud, 2017.

[23] DONG, G., AND BAILEY, J. *Contrast Data Mining: Concepts, Algorithms, and Applications.* 2012.

[24] FRANK, E., HALL, M., HOLMES, G., KIRKBY, R., PFAHRINGER, B., WITTEN, I. H., AND TRIGG, L. Weka-A Machine Learning Workbench for Data Mining. In *Data Mining and Knowledge Discovery Handbook.* Springer US, Boston, MA, 2009, pp. 1269–1277.

[25] FRANZ, M., LOPES, C. T., HUCK, G., DONG, Y., SUMER, O., AND BADER, G. D. Cytoscape.js: A graph theory library for visualisation and analysis. *Bioinformatics 32*, 2 (sep 2015), 309–311.

[26] GARCÍA-BORROTO, M., MARTÍNEZ-TRINIDAD, J. F., CARRASCO-OCHOA, J. A., MEDINA-PÉREZ, M. A., AND RUIZ-SHULCLOPER, J. Lcmine: An efficient algorithm for mining discriminative regularities and its application in supervised classification. *Pattern Recognit. 43*, 9 (Sept. 2010), 3025–3034.

[27] GOOGLE. Google AdWords: publicidad online de PPC de Google, 2017.

[28] GRANITTO, P. M., FURLANELLO, C., BIASIOLI, F., AND GASPERI, F. Recursive feature elimination with random forest for PTR-MS analysis of agroindustrial products. *Chemometrics and Intelligent Laboratory Systems 83*, 2 (2006), 83–90.

[29] GUGELMANN, D., GASSER, F., AGER, B., AND LENDERS, V. Hviz: Http(s) traffic aggregation and visualization for network forensics. *Digit. Investig. 12*, S1 (2015), S1–S11.

[30] GUTIERREZ-RODRÍGUEZ, A. E., MARTÍNEZ-TRINIDAD, J. F., GARCÍA-BORROTO, M., AND CARRASCO-OCHOA, J. A. Mining patterns for clustering using unsupervised decision trees. *Intell. Data Anal. 19*, 6 (2015), 1297–1310.

[31] HASTINGS, W. K. Monte Carlo sampling methods using Markov chains and their applications. 97–109.

[32] HAZEWINKEL, M. *Encyclopaedia of Mathematics*. Springer-Verlag, 1995.

[33] HEMPSTALK, K., FRANK, E., AND WITTEN, I. H. One-class classification by combining density and class probability estimation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2008), vol. 5211 LNAI, Springer Berlin Heidelberg, pp. 505–519.

[34] INFOSEC. Botnets unearthed - the zeus bot - infosec institute, 2013.

[35] IRVINE, M. Google AdWords Benchmarks for YOUR Industry [Updated!], 2018.

[36] JIARUI, X., AND CHEN, L. Detecting crowdsourcing click fraud in search advertising based on clustering analysis. *Proceedings - 2015 IEEE 12th International Conference on Ubiquitous Intelligence and Computing, 2015 IEEE 12th International Conference on Advanced and Trusted Computing, 2015 IEEE 15th International Conference on Scalable Computing and Communications, 20* (2016), 894–900.

[37] JUAN, Y., ZHUANG, Y., CHIN, W.-S., AND LIN, C.-J. Field-aware Factorization Machines for CTR Prediction. *Proceedings of the 10th ACM Conference on Recommender Systems - RecSys '16* (2016), 43–50.

[38] KALMEGH, S. Analysis of WEKA Data Mining Algorithm REPTree , Simple Cart and RandomTree for Classification of Indian News. *International Journal of Innovative Science, Engineering & Technology 2*, 2 (2015), 438–446.

[39] KASPERSKY. Kaspersky cyberthreat real-time map.

[40] KHEIR, N. Behavioral classification and detection of malware through http user agent anomalies. *J. Inf. Secur. Appl. 18*, 1 (2013), 2–13.

[41] KISSMETRICS. Kiss Metrics Platform, 2017.

[42] KUMARI, S., YUAN, X., PATTERSON, J., AND YU, H. Demystifying Ad Fraud.

[43] LIU, F. T., TING, K. M., AND ZHOU, Z.-H. Isolation-Based Anomaly Detection. *ACM Transactions on Knowledge Discovery from Data 6*, 1 (mar 2012), 1–39.

[44] LOGSTALGIA. Logstalgia - a website access log visualization tool.

[45] LOPEZ-CUEVAS, A., MEDINA-PEREZ, M. A., MONROY, R., RAMIREZ MARQUEZ, J., AND TREJO, L. A. FiToViz: A Visualisation Approach for Real-time Risk Situation Awareness, jul 2017.

[46] LOYOLA-GONZÁLEZ, O., MEDINA-PÉREZ, M. A., MARTÍNEZ-TRINIDAD, J. F., CARRASCO-OCHOA, J. A., MONROY, R., AND GARCÍA-BORROTO, M. PBC4cip: A new contrast pattern-based classifier for class imbalance problems. *Knowledge-Based Systems 115* (jan 2017), 100–109.

[47] MAHMOUD, A. Detecting complex fraud in real time with graph databases - the developerworks blog, 2017.

[48] MARCHANT, I. 4 retos de la industria publicitaria en méxico, Aug. 2016.

[49] MARTÍNEZ, J. R. A google, los marketers le rodearon la manzana, Apr. 2017.

[50] MEDINA-PÉREZ, M. A., MONROY, R., CAMIÑA, J. B., AND GARCÍA-BORROTO, M. Bagging-TPMiner: a classifier ensemble for masquerader detection based on typical objects. *Soft Computing 21*, 3 (feb 2017), 557–569.

[51] MITCHELL, M. N. *A Visual Guide to Stata Graphics*. 2012.

[52] MRC. Invalid traffic detection and filtration guidelines addendu. Public Comment Version, Oct. 2015.

[53] MÜLLER, A. C., AND GUIDO, S. *Introduction to Machine Learning with Python*. 2016.

[54] NEASBITT, C., PERDISCI, R., LI, K., AND NELMS, T. Clickminer: Towards forensic reconstruction of user-browser interactions from network traces. *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. - CCS* (2014), 1244–1255.

[55] NEO4J. White paper: Fraud detection discovering connections - neo4j graph database, 2015.

[56] O'BRIEN, D., GUPTA, M., AND GRAY, R. Cost-sensitive multi-class classification from probability estimates. *... of the 25Th International Conference on ...* (2008), 712–719.

[57] OENTARYO, R., LIM, E.-P., FINEGOLD, M., LO, D., ZHU, F., PHUA, C., CHEU, E.-Y., YAP, G.-E., SIM, K., NGUYEN, M. N., PERERA, K., NEUPANE, B., FAISAL, M., AUNG, Z., WOON, W. L., CHEN, W., PATEL, D., AND BERRAR, D. Detecting Click Fraud in Online Advertising : A Data Mining Approach. *Journal of Machine Learning Research 15* (2014), 99–140.

[58] PAPER, C., FRE, M., AND KO, D. Analysis of the open advertising data set.

[59] PERERA, K. S., NEUPANE, B., FAISAL, M. A., AUNG, Z., AND WOON, W. L. A novel ensemble learning-based approach for click fraud detection in mobile advertising. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8284 LNAI* (2013), 370–382.

[60] PETER ADAMS. Open Web Analytics Repository, 2018.

[61] RUADHAN O'DONOGHUE. *AMP: Building Accelerated Mobile Pages*. Packt Publishing, 2017.

[62] RUNKLER, T. A. *Data analytics: Models and algorithms for intelligent data analysis*, 2nd editio ed. Springer Vieweg, 2016.

[63] SAITO, T., AND REHMSMEIER, M. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE 10*, 3 (2015), e0118432.

[64] SAKIB, M. N., AND HUANG, C. T. Using anomaly detection based techniques to detect HTTP-based botnet C&C traffic. *2016 IEEE International Conference on Communications, ICC 2016* (2016).

[65] SILVA, S. S., SILVA, R. M., PINTO, R. C., AND SALLES, R. M. Botnets: A survey. *Comput. Networks 57*, 2 (2013), 378–403.

[66] SIROKER, D., KOOMEN, P., AND HARSHMAN, C. *A/B-Testing: The Most Powerful Way to Turn Clicks into Customers*. John Wiley & Sons, Incorporated, 2013.

[67] SPIROU, J. ClientJS.

[68] TALKING DATA. TalkingData AdTracking Fraud Detection Challenge — Kaggle, 2018.

[69] TANEJA, M., GARG, K., PURWAR, A., AND SHARMA, S. Prediction of click frauds in mobile advertising. In *2015 Eighth International Conference on Contemporary Computing (IC3)* (aug 2015), IEEE, pp. 162–166.

[70] THAI-NGHE, N., GANTNER, Z., AND SCHMIDT-THIEME, L. Cost-sensitive learning methods for imbalanced data. In *Proceedings of the International Joint Conference on Neural Networks* (2010).

[71] TORRANO, C., PÉREZ, A., AND ÁLVAREZ, G. HTTP DATASET CSIC 2010, 2010.

[72] UDAYA SAMPATH K. PERERA MIRIYA THANTHRIGE, JAGATH SAMARABANDU, X. W. Machine Learning Techniques for Intrusion Detection. *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)* (2016), 1–10.

[73] UNIVERSITY COLLEGE LONDON. Google Code Archive - Open Advertising Dataset, 2012.

[74] VIZARD, S. Google under fire as brands pull advertising and ad industry demands action, 2017.

[75] WANG, C.-J., AND CHEN, H.-H. Learning User Behaviors for Advertisements Click Prediction. *SIGIR, Internet Advertising Workshop*, 1 (2011).

[76] WANG, K., XU, G., WANG, C., AND HE, X. A Hybrid Abnormal Advertising Traffic Detection Method. *2017 IEEE International Conference on Big Knowledge (ICBK)* (2017), 236–241.

[77] XIE, G., ILIOFOTOU, M., KARAGIANNIS, T., FALOUTSOS, M., AND JIN, Y. Resurf: Reconstructing web-surfing activity from network traffic.

[78] ZEIFMAN, I. Bot traffic report 2016. Tech. rep., Incapsula, Jan. 2017.

[79] ZHANG, X., LASHKARI, A., AND GHORBANI, A. A lightweight online advertising classification system using lexical-based features. *ICETE 2017 - Proceedings of the 14th International Joint Conference on e-Business and Telecommunications 4*, January (2017).