Differentiated consumption of services through OTT applications in the framework of the data plan

Trabajo de Grado

Lina Isabel Aristizábal Casanova

Director: MSc. Juan Sebastián Rojas Meléndez
Co-Director: PhD. Juan Carlos Corrales Muñoz

Departamento de Telemática
Facultad de Ingeniería Electrónica y Telecomunicaciones
Universidad del Cauca
Popayán, Cauca, 2019

Differentiated consumption of services through OTT applications in the framework of the data plan

Lina Isabel Aristizábal Casanova

Trabajo de grado presentado a la Facultad de Ingeniería Electrónica y tele comunicaciones de la Universidad del Cauca para obtener el título de: Ingeniero en Electrónica y Telecomunicaciones

Director: MSc. Juan Sebastián Rojas Melendez
Co- Director: PhD. Juan Carlos Corrales Muñoz

Departamento de Telemática
Facultad de Ingeniería Electrónica y Telecomunicaciones
Universidad del Cauca
Popayán, Cauca, 2019

ACKNOWLEDGEMENTS

Thanks to Universidad del Cauca, PhD. Juan Carlos Corrales Muñoz and MSc. Juan Sebastián Rojas Meléndez for their guidance and advice throughout the development of this undergraduate thesis.

STRUCTURED ABSTRACT

The expansion of mobile communications and OTT applications have caused mobile operators to have a high level of uncertainty regarding the quality indicators of the services of the application layer that they offer, this is caused because today the encrypted information that the applications handle makes this task more difficult. As network operators do not have enough information on the customer side, and having in mind that the little information that these operators collect from users, is used mostly for marketing purposes, it causes that they cannot offer personalized plans, better quality of service and a better user experience.

With this in mind, the objective of this project is to monitor and classify the consumption of a user's OTT services within the framework of an LTE network data plan. However, as there is no access to a real LTE network, in this case a simulated LTE network will be taken.

To supply the need above, it is proposed to take a simulated LTE network that is installed within the network of the Universidad del Cauca, in which it is possible to simulate different Internet users and servers and through which information is exchanged. However, the main objective of this undergraduate thesis is to exchange information of specific OTT applications, something that this simulator does not have, since its traffic generator only creates generic Internet traffic. For this reason, it was decided to investigate different traffic generators in the hope of finding one that would adapt to the needs of this research project. Despite this exhaustive research, it was concluded that no current traffic generator is capable of creating traffic from specific OTT applications, and for this reason it was decided to create a synthetic OTT application generator.

This synthetic generator creates flow of applications such as WhatsApp, YouTube, Skype, Google and Spotify, resulting, after a long process, different datasets of the applications mentioned above. With these datasets, cleaning processes were followed with the CRISP-DM methodology, later they were grouped to form a single dataset and finally this dataset was validated with different machine learning algorithms such as J48, Bagging, IBK, NaiveBayes, among others.

As future works, it is proposed to do the modeling and the synthetic generation of other OTT applications, to create an IDE that everybody can handle and be able to use the generator in a more intuitive way, to validate the datasets using other machine learning algorithms, among others.

Keywords: OTT applications, LTE network, machine learning, data set, traffic classification, traffic generators.

RESUMEN ESTRUCTURADO

La expansión de las comunicaciones móviles y las aplicaciones OTT han provocado que los operadores móviles tengan un alto nivel de incertidumbre frente a los indicadores de calidad de los servicios de la capa de aplicación que ellos mismos ofrecen, siendo esto, gracias a que hoy en día la información encriptada que las aplicaciones manejan hace mas difícil esta tarea. Como los operadores de red no tienen la suficiente información del lado del cliente, y teniendo en cuenta que la poca información que estos operadores recolectan de los usuarios, es mas que todo usada para fines de marketing, provoca que no se puedan ofrecer planes personalizados ni que puedan ofrecer una mejor calidad de servicio y una mejor experiencia de usuario.

Teniendo en cuenta lo anterior, el objetivo de este proyecto es monitorizar y clasificar el consumo de los servicios OTT de un usuario en el marco del plan de datos de una red LTE. Sin embargo, como no se tiene acceso a una red LTE verdadera, en este caso se va tomar una red LTE simulada.

Para cumplir el objetivo anterior, se plantea tomar una red LTE simulada que esta instalada en la red de la Universidad del Cauca, en la cual es posible simular diferentes usuarios y servidores de Internet y a través de la cual se hace intercambio de información. Sin embargo, el objetivo principal de esta tesis de pregrado es poder intercambiar información de aplicaciones OTT específicas, algo que este simulador no posee, ya que su generador de tráfico solo crea tráfico genérico de Internet. Por esto, se tomo la decisión de investigar diferentes generadores de tráfico con la esperanza de poder encontrar uno que se adaptara a las necesidades de este proyecto de investigación. A pesar de esta exhaustiva investigación, se llego a la conclusión de que ningún generador de tráfico actual es capaz de crear tráfico de aplicaciones OTT especificas, y por este motivo se opto por crear un generador sintético de aplicaciones OTT.

Este generador sintético, crea flujo de aplicaciones como WhatsApp, YouTube, Skype, Google y Spotify, dando como resultado, después de un largo proceso, diferentes datasets de las aplicaciones mencionadas anteriormente. Con estos datasets se siguieron procesos de limpieza con la metodología CRISP-DM, posteriormente se agruparon para formar un solo dataset y finalmente este dataset se valido con diferentes algoritmos de Machine learning como lo son, J48, Bagging, IBK, NaiveBayes, entre otros.

Como trabajos futuros, se propone hacer el modelamiento y la generación sintética de flujos de otras aplicaciones OTT, crear un IDE para que todo el mundo pueda manejar el generador de manera mas intuitiva, validar el conjunto de datos usando otros algoritmos de machine learning, entre otros.

Palabras clave: Aplicaciones OTT, red LTE, aprendizaje de maquina, conjunto de datos, clasificación de tráfico, generadores de tráfico.

CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

CHAPTER 1

INTRODUCTION

1.1 Context

Currently, new applications and services are in constant developing and evolution in an exponential way, supported by the expansion and deployment of LTE[1] networks around the world, this being one of the largest solutions of next-generation mobile systems [1]. LTE has allowed a large increase in Internet traffic [2] as it is a data-oriented mobile network. Its modern architecture offers higher speeds in data link, low latency, requires a simple and economical maintenance [2], improves QoS[2] [3] and also allows the offer of various services to users, which allows them to enjoy better experiences in the consumption of applications.

Considering the previous paragraph, it is clear that the Internet is undergoing major changes in recent times, presenting a constant transformation and introducing different applications and services known as OTT[3]. These kind of applications provide a service over a data channel, either through a wired network, Wi-Fi or through mobile networks, taking advantage of the large providers' infrastructures, providing similar services, obtaining a higher profit margin as services on demand while avoiding infrastructure costs. All of this has caused that the business models of the traditional operators are affected in a negative way since the users prefer the OTT applications and services instead of the conventional services offered by traditional operators, generating economic losses for them; Examples of OTT applications are VoIP[4], VoD[5], video streaming, P2P[6], Web, among others.

1.2 Motivation

Nowadays, the expansion of encrypted communications has caused network operators to have a high level of uncertainty when it comes to quality indicators at the application layer,

[1] *Long Term Evolution*
[2] *Quality of Service*
[3] *Over the Top*
[4] *Voice over Internet Protocol*
[5] *Video on Demand*
[6] *Peer to Peer*

such as video or audio quality, blocking duration, among others [4]. Furthermore, there is a lack of extensive knowledge related to the behavior of users within the network, which implies that QoS management represents a significant challenge for operators as they cannot offer personalized plans or do not know what kind of services or mobile devices fit their clients' preferences. Currently, the information that the network operators have, is collected from Datasets, such as the CDR[7]s, and they are mainly used for marketing purposes, only generating the data plans offered to the users, whether prepaid or postpaid. Hence, the information gathered by traditional operators does not take into consideration the QoE[8] of users when they are using the applications on their smartphones, which sometimes leads to a stalemate in the improvement of services [5].

Therefore, mechanisms such as network monitoring (control and surveillance of the network) and traffic classification (categorization of the network information) play an important role since they permit to capture different Internet flows, gathering data that allows to obtain an adequate perspective of the information exchanged inside the network [6], enabling network administrators to detect threats, maintain the quality of the service, prevent the collapse of networks, among other functionalities [7].

1.3 Problem Definition

Given that applications and the world of smartphones and devices with Internet connection capabilities are increasing and each time are more sophisticated, network operators must be able to provide users with a better quality of service and user experience. With this in mind, traditional operators have to find the most suitable way to not be left behind in front of the exponential gains that the OTT applications providers are obtaining. Therefore, traditional operators need to leverage all the data that is exchanged inside their networks in order to obtain a complete knowledge of their customers' behavior and needs in terms of services and applications.

In view of the above, there is a need to design and implement an environment or mechanism that supports the monitoring of an LTE mobile network, allowing the construction of datasets, in order to gather information that holds the Internet consumption behavior of users within a network in an implicit way (without bothering the user), in such a way that the operators can take advantage of the benefit that this type of information can offer. Based on what has been previously described, this undergraduate thesis presents the following research question:

[7] *Call detail record*
[8] *Quality of experience*

How to implicitly monitor an LTE network to classify consumption trends of OTT services from mobile users?

1.4 Objectives

Considering the previous motivation and the research question, this project aims to monitor and classify the consumption of a user's OTT services in the framework of the data plan of a simulated LTE network.

In order to accomplish the objective an identification of a tool that allows the simulation of a LTE network infrastructure must be done, to be able to capture the OTT service consumption traffic such as video, messaging, browsing and audio generated on the simulated network users. A dataset will be collected, containing the network attributes related to the consumption of OTT services by user. An implicit classification of the consumption of OTT services will be perform through machine learning algorithms, validating the dataset obtained.

1.5 Research Contributions

The contributions that are expected with this undergraduate thesis are:

- The construction of an environment in which it is possible to monitor and collect traffic from OTT applications generated by users connected to an LTE mobile network.
- A list of attributes that must be monitored in an LTE network in order to facilitate the construction of datasets related to the consumption of OTT applications done by users.
- A dataset with the attributes taken from the LTE network that enable a posterior classification of OTT services.

1.6 Document Structure

This document has been divided into 7 chapters described below.

- Chapter 1 presents the Context, Motivation, Problem statement, Objectives, Research Contributions, and the structure of this document .
- Chapter 2 presents the background about the relevant topics concerning this research. This topics include mobile networks, traffic capturers, traffic generators, dataset, LTE simulators, OTT applications, traffic classification and machine learning. It also contains the related works that describes this proposal.
- Chapter 3 presents the research that was done regarding the traffic generators and the integration with the simulated environment.

- Chapter 4 presents the process that was performed in order to obtain a synthetic generator of OTT applications and the datasets obtained alongside with the cleaning process.
- Chapter 5 presents the tests implementing the classification through machine learning algorithms and its results.
- Chapter 6 presents conclusions and future work.

CHAPTER 2

STATE OF THE ART

2.1 Background

In this chapter, the concepts that were necessary for the development of this project are presented. The initial description is about mobile networks, traffic capturers, traffic generators, traffic classification, dataset and OTT applications. Subsequently the concept of machine learning is introduced. Finally, a brief survey of an LTE network simulator and its components is presented. Additionally, a set of related works about LTE simulators, traffic monitoring, datasets, traffic classification and OTT applications is presented.

2.1.2 Mobile Networks:

Mobile networks are those that allow the transmission of both, voice and multimedia data, through a mobile or cellular device, without having to be connected to any physical link, sending information through microwaves. It could be described as the capacity of using mobile technology while on the move [8].
These networks have evolved over the years and the use of smart phones and tablets has increased in the growth of mobile data traffic and signaling traffic [9]. The standards for these networks are changing and the evolution can be seen in standards such as 2G[9], 3G[10], 4G[11], WiMax, EDGE[12], GPRS[13], among others, being LTE one of the most used networks in the world [8]. This technologies are introduced by the 3GPP[14], who is the one in charge to define specifications and produce reports of these standards. It covers cellular telecommunications, including radio access, the core transport network, and service capabilities - including work on codecs, security, quality of service - and thus provides complete system specifications [10].

[9] Second generation
[10] Third generation
[11] Fourth generation
[12] Enhanced Data rates for GSM Evolution
[13] General Packet Radio Service
[14] 3th Generation Partnership Project

2.1.3 Traffic Captors:

The tools that capture the traffic of the current networks, usually called *Sniffers*, are mechanisms that are responsible for the monitoring and analysis of the network traffic. Its main function is to examine packets, protocols and frames that are sent over the Internet, allowing the capture and visualization of the data. These can be based on software or hardware, collect the necessary data from a data network and then provide the possibility of using the information as needed. Normally these *Sniffers* are used only to capture the traffic of a local network, which means, it only captures the traffic from the network it belongs. If the problem is to capture traffic from other networks, other tools should be used or a change of the network's infrastructure must be done [11].

2.1.4 Traffic Generators:

The evolution of mobile networks is highly accelerated, which makes networks very diverse in terms of protocols, applications, technologies and devices [12]. This evolution created the need of increasing the Internet bandwidth having in mind that the traffic of end users must be well supported. For this reason, traffic generators are vital in the design, development and administration of networks. At the same time, network security has become imminent and tools are created to adequately reflect network conditions and topologies that allow testing the characteristics, problems and advantages of a network [13].

Traffic generators are a tool used to inject packets into a network in a controlled manner. These tools inject network packets of synthetic traffic and are capable of adapting to different network conditions [14].

2.1.5 Traffic Classification:

Traffic classification is an automatic way to categorize the traffic that comes from the network in order to regulate it as data flows [15]. This characterization depends on various parameters such as protocols or techniques such as the port-based technique, the technique based on payload, statistical classification and DPI[15] [16]. With this classification, network traffic can be handled in different classes or services, or users can be differentiated within a network.

[15] *Deep packet inspection*

2.1.6 Dataset:

Datasets are sets of data that collect, capture and retain a large amount of information organized in byte streams and stored in logical registers [17]. In the present, datasets can contain any type of information, e.g. medical or insurance records, and these records can be stored in a program that will eventually make use of them. These datasets can be cataloged as appropriate, they can be viewed and they can be organized in different ways, according to how the information is planned to be accessed [18]. The data that is collected in the dataset is usually collected because it needs to be analyzed, an example of a tool for the analysis of datasets can be Microsoft Excel, since it allows to store different types of data and later analyze or manipulate them according to what is needed.

2.1.7 Over-The-Top Applications

Over The Top applications are Internet applications that provide different services or functionalities to end users by leveraging the ISPs[16] infrastructure. Some of the most common functionalities are video calls, voice calls, video streaming, video conferencing, IM[17], among others. Besides these applications are often the ones competing with the applications provided by the operators of telephony and internet [19].

2.1.8 Machine Learning

Machine learning is a branch of AI[18] that has been investigated since the 90's [20]. It belongs to a computational branch of algorithms that has evolved, and are designed to imitate human intelligence, learning from different environments. The algorithms that compose machine learning are computational algorithms that are not programmed to produce a specific output, on the contrary, they are programmed in such a way that these algorithms change their infrastructure as they have more experience (training) to be better each time [21].
In view of the above, machine learning is being used in different branches of research recently being used for the evaluation and classification of network traffic, such as HTTP[19], FTP[20], SMTP[21], etc., helping the quality of service, monitoring and analysis of networks [22].

---

[16] *Internet Service Providers*
[17] *Instant messaging*
[18] *Artificial intelligence*
[19] *Hypertext Transfer Protocol*
[20] *File Transfer Protocol*
[21] *Simple Mail Transfer Protocol*

Most of all, it is a tool used by the ISPs, to know what kind of applications goes through their network, and along with the traffic classification service providers can have a map containing the performance of the network [23].

Machine learning is divided into different techniques, and the most used are: supervised learning and unsupervised learning, a brief description of them is given below.

2.1.8.1 Supervised learning

Each input data training element is matched with a known classification label allowing the algorithm to see similarities and differences between the objects it is going to classify. It learns to identify the most important qualities of the datasets. This is done so the algorithm can be able to recognize instances that have not been seen by it before and classify them properly [21].

2.1.8.2 Unsupervised learning

Unlike supervised learning, these algorithms do not have a set of output tags that allow the algorithm to make comparison between one element or another [21], that is, they do not have a previous guide treating the instances as random variables. These algorithms take the datasets and learn on the move, being able to adjust what was learned earlier and changing it as the learning is being done. At the output of this learning a data classification model is formed [24].

2.1.9 LTE Simulator

LTE represents an emerging technology that provides mobile users the opportunity to access broadband internet through their smartphones. It is clear that the aspects of LTE networks is a topic that is worth investigating for both industrial and academic communities [25]. For this reason the results obtained in the master's thesis titled "Virtualized Evolved Packet Core for LTE Networks" developed in Universidad Del Cauca is taken into account within this project, where an LTE simulator based on the EPC[22] architecture is presented.

This simulator contains the modules (MME[23], HSS[24], SGW[25], PGW[26]) along with a RAN[27] simulator and a SINK node [9]. These modules act as a server or client in each of their ports.

[22] *Evolved Packet Core*
[23] *Mobility Management Entity*
[24] *Home Subscriber Server*
[25] *Serving Gateway*
[26] *Packet Data Network Gateway*
[27] *Radio Access Network*

The control and the communication of data pass through the objects sending requests and answers between several entities of the system. To use this simulator it is necessary to have access to virtual machines based on Linux OS[28] (Ubuntu 14.04). Six different virtual machines are needed, where each one of the machines represents a different network module.

Figure 1 illustrates the architecture is shown below and a brief description of the modules that integrate this LTE EPC network is given:

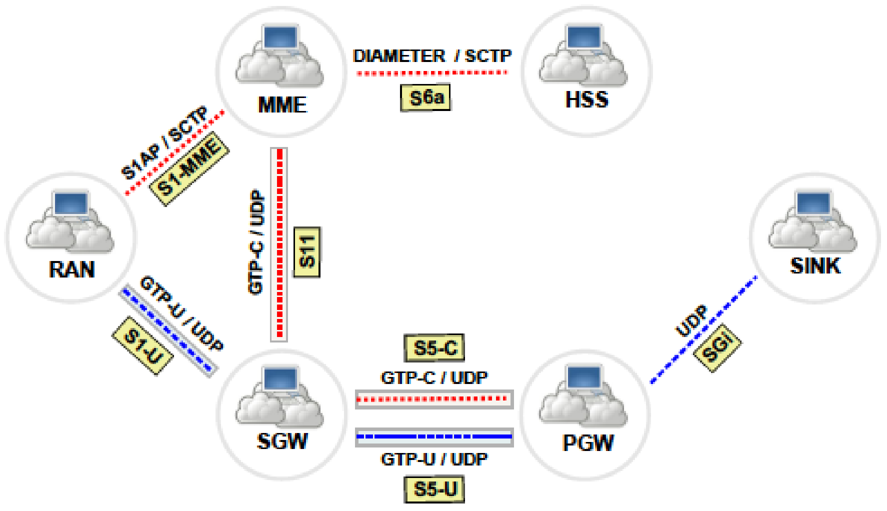

Figure 1. NFV[29]-Based LTE EPC System [9]

- **RAN:** Combines the UE[30] and eNodeB[31] modules, where UE represents the device used by the end users and the eNodeB represents the base station for the coverage of mobile users. It generates control and data traffic in the EPC network. This RAN simulator creates threads for each client object that is created, and what it does is to handle the communication of the control plane and the data plane with MME and SGW respectively [26].

- **MME:** This module communicates with the HSS to find the client's login information, looking for information related to it and updating the location. It also communicates with the SGW module for procedures related to the session and the carrier. It maintains a global map where all the communications are located and stores the information related to the connection and the status of the UEs. This hash map is accessed or updated on requests and responses from UE, HSS and SGW. [26]

[28] *Operative System*
[29] *Network Functions Virtualization*
[30] *User Equipment*
[31] *Evolved NodeB*

- **HSS:** Does operations related to databases and behaves as a client for the communication with the MME module. It is used to store information related to the UE such as authentication, subscription profile, and location tracking. It is in charge of answering the questions of the MME module with the information he seeks and processes from the database [26].

- **SGW:** This module manages the communication of the control plane with MME and the information of the data plane with PGW. It uses an upload server object and a download client object. Also similar to the MME, it maintains a hash map to store all the context information of the UE [26].

- **PGW:** Manages data plane communication with SGW and the SINK module. The information related to the UE is accessed or updated in a separate hash map. It has an IP[32] table map that is used to assign static IP addresses for each UE that connects to the network [26].

- **SINK:** This module is used to represent a PDN[33] server. Its purpose is to receive the generated traffic load and return the knowledge as a traffic download. The module uses a client/server object for the data transfer of uploading and downloading with PGW [26].

2.2 Related works

In this section a brief description of the most relevant related works is presented, taking into account the research question previously presented. It should be noted that at the moment there are no similar works or contributions made for the specific area of study. For this, it was implemented a systematic mapping of academic documents, based on the methodology proposed in [27], that allows to find different papers related to this project. The sources used for the collection of information were: IEEExplore, Springerlink, ResearchGate and Scopus, using Springerlink as the main source of information.

The topics of interest for this project are: traffic monitoring, which was divided into two topics: detection of intruders and traffic captors; LTE simulators, datasets, aimed at detecting intruders and mobile networks; the classification of traffic, centered on classification techniques and classification in mobile networks; and finally the OTT applications. With the topics of interest already identified, a review of the articles that were found is made, discarding those that do not contribute to the research project, distributing the remaining

[32] *Internet protocol*
[33] *Packet Data Network*

articles or works in the following way: 3 Documents for traffic monitoring, 2 Documents for LTE simulators , 4 Documents for datasets, 3 Documents for traffic classification and finally 2 Documents for OTT applications.

2.2.1 Traffic Monitoring - related works

As mentioned before, for this thematic core different approaches defined as traffic captors and intrusion detection were considered.

Starting with the traffic captors, in article [11] the importance of the captors is explained, emphasizing that these tools are good for data analysis, helping to examine packages, protocols and frames sent over the networks, whether they are wired or not, showing the content of the packet, which helps to understand the behavior of the network, make good use of bandwidth resources and detect anomalies in the network security. These tools are usually called *sniffers* and can be based on hardware or software, but they always collect traffic information locally. In this work they also make a brief characterization of different *sniffers* such as Tcpdump, which is a tool that runs in console mode and is available for almost all operating systems and is a free software; Wireshark, is one of the most used *sniffer,* because it allows to observe the fields content with the highest level of detail possible; CommView, is a commercial tool designed for Windows and requires an Ethernet network of 10/100/1000 Mbps; Kismet, is a program that works in various operating systems, operates through the console. But before starting using this tool, the wireless cards driver must be understood and administrator permissions must be obtain to change different configurations settings; NetworkMiner is a tool that works mainly for Windows systems, this tool does not aim to collect information about network traffic, but to collect information about the host; and finally there is OmniPeek, which is a commercial network analyzer that has a very easy-to-use GUI[34], designed for both wireless and Ethernet networks. The above tools are the most used, however, there are other tools that are not like traditional *sniffer*, and have many more features. These tools are: InSSIDer and NetStumbler.

In [28], they speak of the indispensable thing that is the monitoring of the network to be able to increase the safety of the same and to be able to understand the performance of modern networks. However, there is a very large gap between monitoring applications and traffic capture tools, since today's applications work in high-level layers, such as the transportation layer, and these tools work in the network layer. This sometimes causes irrelevant or duplicate packets of traffic to be captured. That is why they present a tool called Scap, an API[35] that works in the transport layer and allows to capture traffic with very few packet losses, besides it helps different tools to improve their performance in a significant way. It is

[34] *Graphical user interface*
[35] *Application programming interface*

based on aggressive optimizations at the kernel level and the NIC[36]s. Therefore, they conclude that the approach they gave to this tool is the most adequate to close the gap in the capture and monitoring of traffic, as networks become faster and the applications more sophisticated.

In the second defined approach (Intrusion Detection) there is [29], which talks about how the continuous analysis and monitoring of the network is a very heavy task. The capture and processing of the packets in a continuous way is demanding computationally speaking, and it is even more when in the day there are peaks of traffic that can over dimension the network. For an intrusion detection environment the information must be delivered as quickly as possible and thus be able to make an early detection of insecurities. However, the oversized networks at peak times cause this information to be delayed and therefore the network data starts to pile up. Because of it, they present a simulated environment tool with variable network loads and large datasets based on Hadoop, which uses cloud bursting to break some traffic and improve the load network performance up to a 50%.

2.2.2 LTE simulators - related works

For LTE simulators, only the papers after 2014 were taken into account. This decision was made since some of the simulators were developed with a prior infrastructure, only providing the physical layer of the network. Also, some of the simulators did not focus on the data exchange within the network, they were not scalable and were used for an only purpose.

In the first article, [30], they explain how networks simulation is an optimal way for research on LTE networks. However, there are no simulators that allow monitoring specific applications, since within the networks generic data is sent. They created the LTE Open-Sim open simulator, which allows the transport and application layers to be implemented and do not have the limitation of the other simulators. It simulates a virtual LTE network, with the ability to connect real hosts through a real-time wired link. In this case, this simulator is made and optimized to track video streaming applications and focus on QoE research in the transmission of video through an LTE network. They present a simple simulator that consists of 3 modules in its main architecture, the Virtual LTE network module, Transit gateway and Routing table. The first is used to provide the simulated LTE network with IP connectivity. The second is used so that data packets can be transferred and received within the network, and the last one is used to give the routing rules when the packets are sent on the network.

In the article [31], a SimuLTE simulator based on OMNeT ++ is presented. It has a modular architecture, which facilitates the expansion and integration of the modules. It also allows to the LTE network not being the only available network, and allows it to be a part of a much

[36] *Network Interface Cards*

wider network. SimuLTE simulates the data plane of the RAN. The modules communicate through messages that are sent by gateways that act as interfaces. It presents a complete protocol stack, a realistic physical layer and programming capabilities. It is a simulator that has the ability to evolve according to the evolution that real networks are presenting and the authors of the article are still working on its improvement and include new features to the simulator.

2.2.3 Datasets - related works

The papers found in relation to the thematic core dataset core were divided into two different subtopics. The dataset oriented to intrusion detection and the dataset oriented to mobile networks.

Starting with the datasets oriented to IDS[37], in the paper [32] they talk about how IDS' are very important to detect malware in networks and how the non-optimization of these tools makes this task a very complicated one, since computational time is very high. They take different datasets that contain network traffic, taking into account only the attributes that can contribute to the identification of Malware, since many of these attributes may not be relevant, may be redundant or may not contain information on the intrusion detection; and then test these new datasets with classifiers like NaiveBayes, J48 and PART. They conclude that the technique used reduces by 82.93% the attributes that are computed and has an acceptable accuracy.

In paper [33] they talk about the importance of the defense against the attack mechanisms in the networks, and therefore they create a toolkit called Intrusion Detection dataset. This toolkit facilitates the creation of datasets that are ruled by four requirements: *named attacks*, *high data quality*, both attacks and normal traffic, *publicly available* to be reproducible and *flexibility* to test them in different scenarios. This tool lets inject traffic from named attacks in an environment that is totally controlled and allows to create the pcap files that contain the attacks and then be able to do the analysis.

The second subtopic for the Dataset are those oriented to mobile networks. In this part it was found the paper [34], where the authors based their research on the CDRs that are collected in the mobile networks. These CDRs contain information about users, how, when and with whom they are communicating. They may also contain location data and in some cases the age and gender of the person. These datasets allows to observe the users' behavior from a very high point of view, finding that the behavior of the people's mobility is very predictable, being able to monitor daily life and the response to catastrophes in a populations. These

[37] *Intrusion Detection System*

dataset are not for public knowledge and companies only give a certain amount of information to a few research groups to be managed.

The article [35] takes a dataset of cellular networks obtained by Orange, where they applied identification strategies for social influence. They emphasize that MNO[38]s take into account this data just for operational use or for the billing details. But these datasets have a lot of information about the initiator and receiver of the calls, the date and even the geolocation of the individuals, and sometimes it can be inferred the work place and home of the subscribers, and also give socio-economic information of the environment. The study shows that social influence has a great impact on customers and that MNOs can benefit from these analysis in order to improve the user's QoE.

2.2.4 Traffic Classification - related works

This thematic core is also divided by two subtopics: traffic classification techniques and traffic classification on mobile networks.

In the articles related to traffic classification techniques, it has been found [36], where they talk about how difficult it has become to classify traffic in encrypted networks. For this reason, they propose a classification of traffic based on neuronal networks. This method integrates the extraction of features, the selection of features and a classifier in the same frame. It is a method based on deep learning and is used 1D-CNN[39] as a learning algorithm using an end-to-end strategy. This technique does not divide the problem into parts, instead, directly learns the non-linear relationship between the traffic input and the output label. They conclude that CNN-based algorithms have a good potential for classifying traffic since there were significant improvements respect to the previously used algorithms, as is the case of the traditional method of encrypted traffic classification using the strategy of dividing the problem in several parts.

In work [37], they talk about the different approaches that have been used over the years for the traffic classification, and emphasizes that these techniques cannot explain the relationship between the characteristics that describe the traffic flow and classes of the corresponding traffic. For this reason, they use the MOEFC fuzzy classifier that has good precision and interpretability.
They used two different dataset extracted from a real internet traffic network. Their results showed that this technique reaches a precision of 93% and the models generated are characterized by high levels of interpretability. This was done by taking a set of data to train

[38] *Mobile network operators*
[39] *Convolutional neural networks*

the classifier and then using the other for testing reasons, finding that the classification rate, the rate of true positives and the rate of false positives have an acceptable level of precision.

In the traffic classification oriented to mobile networks, paper [38] talks about how in telecommunications networks it is needed a lot of detection and classification of anomalous behaviors for a quick response to any occurring failure. They show that if a project is working with a dataset with the majority of the traffic being anomalous, the learning systems will take this behavior as something normal and those failures will not be able to be classified properly. Therefore, this paper proposes a method for detecting anomalies given in two stages: datasets filtering and anomalies classification by an automated process. They made use of the mobile network, Nokia Serve atOnce Traffica, which allowed them to monitor the QoS in real time, the service use and the traffic passing through the entire mobile network. They found that the proposed FSM[40] method detected a greater number of anomalies by obtaining the data in a live LTE network. This method does not seek to replace the methods that are normally used for the anomalies detection, but complements the existing methods.

2.2.5 OTT Applications - related works

For this subsection, the papers related to OTT applications are oriented to mobile networks. Taking into account the gaps that mobile services providers have when it comes to these technologies.

In [39] the authors introduce the experience quality function for services based on SDN[41]. They talk about how the MNOs are falling behind with the earnings, while the OTT applications are gaining momentum and are keeping the earnings. This approach allows the MNOs to offer a better service for OTT applications, ensuring good performance of these through traffic management mechanisms. Thus, QoE-Serv allows operators to be in the middle of OTT providers and customers, offering free services or more advanced qualities for premium users. So, this approach will leave a gap open to introduce new business models for both OTT providers and MNOs.

In [5], the authors talk about how LTE network operators do not take into account the application or the content that is being transmitted in the network. Therefore the client’s experience in terms of the use of OTT applications is not very good, since each session has a demand for individual bandwidth and is very dynamic. What network operators do nowadays is to maximize spectral efficiency, but this is not linked to the user experience. For that matter, it is proposed an experience quality manager that formulates dynamically the QoE objectives for the applications, adapts the bandwidth for the specific QoE of the application, ensuring that each session has the correct amount of resources needed for a good

[40] *failure significance metric*
[41] *Software defined networks*

QoE, instead of providing circumstantial improvements. To achieve this, they are based on the type, content characteristics and traffic patterns of the OTT applications, and thus make the user experience much better within mobile networks.

The table below presents a summary of the proposed scheme for the classification of articles, along with a brief description of each defined thematic group.

Thematic Groups	**Categories**	**Related works**	**Description**
TRAFFIC MONITORING	Intrusion Detection	[29]	In this paper they talk about the traffic detection in networks, but more specifically about the intrusion detection. Taking into account the different tools that allow capturing traffic, they talk about how sometimes the capture and analysis of traffic becomes a heavy task and present a solution to this problem, allowing a reduction in the traffic processing time.
	Traffic captor	[11], [28]	These papers talk about the different traffic captors used for different purposes in the networks. They show the most used *Sniffers* and how these tools are the ones used for the monitoring and analysis of traffic on the networks.
LTE SIMULATORS	LTE network Simulator	[30], [31]	These works focus on different LTE network simulators. They expose the tools they have used and which of these tools are used in the simulators proposed. What stands out is that all the simulators presented in this part are proprietary.
DATASETS	Intrusion Detection	[32], [33]	These research projects expose the great need to cover the data with a good security system. They mainly speak about the characteristics that the different network information has,

			emphasizing on the attributes that must be taken into account when processing the dataset for the intrusion detection and malware.
	Mobile Networks	[34], [35]	These papers are focused on the traffic generated in the mobile network through different applications existing today, through smartphones. They talk about how the mobile network provides a large amount of information which can be used to create market strategies.
TRAFFIC CLASSIFICATION	Traffic classification techniques	[36], [37]	These articles talk about the different traffic classification techniques that exist at the moment. They are also focused on the existing problem with the network encrypted data and how they can have an optimization of the working time for computers when the information classification is being done.
	Traffic classification on mobile networks	[38]	In this paper they talk about how network administrators need to adequate tools to make a more efficient management of LTE networks. The network can behave abnormally and introduce errors without network administrators realizing it. In order to fulfill this purpose, they created the FSM method to be able to provide greater accuracy when detecting anomalies in the networks.
OTT APPLICATIONS	Mobile networks	[39], [5]	These works are focused on OTT applications in current mobile networks. They talk about the quality of service in LTE networks, and how these applications have affected mobile service providers. They explain how MNOs can take

			advantage of the information that OTT applications deliver regarding customers.

Table 1. Related Works classification.

2.3 Gaps

Previously it was presented the different approaches of the works related to the proposed thematic groups. It can be seen that in the topics of traffic monitoring and dataset, intrusion detection approaches were found, this being one of the most treated topics, since the security in data networks is an issue that concerns everybody. Very few works were found concerning mobile networks, and more specifically to LTE networks. In these papers they generalize the capture, analysis and monitoring of network traffic for generic networks. On the other hand, LTE simulators have a good approach in terms of network management, taking into account the real networks and scalability, so various purposes can be fulfilled with them, such as having a simulator capable of working in the transport and application layer of a network or have a realistic physical layer with the complete protocol stack. In the thematic core of OTT applications, they want to demonstrate that with network monitoring operators, mobile service providers can benefit by taking advantage of the QoE. In the traffic classification techniques, they talk about different approaches that can be used to have the most optimal classification for the different types of traffic.

The following table will show the identified gaps of the works that were most relevant according to the thematic nuclei:

Some of these works are related to the construction or use of datasets. However, not many of them take into account the harvest of OTT application traffic generated by users within a mobile network.

In none of the works mentioned above an evidence of a data classification in mobile networks was found, from the OTT applications that users use, to obtain a better quality of service.

None of the works considered an integration of the LTE network simulator and the data capture, as this project aims, since they all talk about the tools needed to perform this but separately, that is, they do not talk about the interaction between the simulator with the data capture tools, nor the creation of a dataset from the integration of these tools.

Traffic monitoring related works		
Related work	**Contributions**	**Gaps**
[11]	It talks about the most used tools for capturing traffic, describes each one in a brief but accurate way.	It has no relationship with any specific type of traffic or with LTE networks.
[28]	With its Scap tool, it allows to capture traffic from the transport layer and not from the network layer.	Its orientation is general and does not cover the capture of specific applications.
[29]	They present a simulated tool, based on cloud bursting helping to minimize the weight of a network load is so the traffic processing is done in a much faster way.	Does not talk about specific traffic and does not include LTE networks.
LTE Simulators related works		
Related work	**Contributions**	**Gaps**
[30]	They present a simulator that works in the transport and application layer, with the ability to connect real hosts to the network.	They focus on QoE for video transmission and do not cover other applications such as VoIP or IM.
[31]	They make a simulator based on OMNeT ++, with great scalability of the network so it can evolve as the LTE networks progress.	It is a much more focused simulator in the physical layer and the RAN data plane. It does not focus on OTT applications or the different ways of capturing the traffic that goes through the network.
Dataset related works		
Related work	**Contributions**	**Gaps**
[32]	They let know that the cleaning of the dataset reduces the computational time and in addition it can increase the accuracy with which the classifiers work.	They only take into account the IDS and do not talk about mobile network traffic or OTT applications.

[33]	They create a tool capable of injecting attacks traffic to the networks, later being able to extract the dataset for its later analysis.	The generated traffic is only malicious traffic, which contains attacks on the networks.
[34]	They talk about the CDRs that companies collect, and that from these data behaviors can be inferred of certain groups of users, such as knowing how they can respond to catastrophes.	They do not focus on specific applications, they only monitor the mobility of users and how predictable their behavior can be.
[35]	They talk about how MNOs can benefit from the information they collect from CDRs. Inferring the work or house place of a person.	Despite being based on a dataset provided by a cellular company, it does not take into account much information that CDRs may contain regarding traffic.
Traffic classification related works		
Related work	**Contributions**	**Gaps**
[36]	It proposes a traffic classification based on neural networks.	It talks about the classification of encrypted traffic, but does not take into account the traffic classification of specific applications.
[37]	Sometimes the classifiers cannot give an exact relationship between the characteristics that describe the traffic flow and the classes corresponding to the traffic. That is why they proposed the MOEFC classifier which gives a good precision and has good interpretability.	The traffic they take for this project is generic and does not come from LTE networks.
[38]	For contemporary LTE networks, a quick response system is needed in real	Despite working on a simulated LTE network, the traffic they collect is for the

	time. Therefore they propose a method that allows to detect anomalies in the network in an effective way.	evaluation of anomalies and not for the traffic of OTT applications.
OTT Applications related works		
Related work	**Contributions**	**Gaps**
[39]	They show how MNOs can take advantage of OTT applications to introduce new business models.	They do not talk about a system that allows the capture of traffic implicitly for its analysis.
[5]	They talk about how the network operators are failing respect to the users' QoE and present an alternative based on the type of traffic, the content characteristics and the traffic patterns of the OTT applications so the QoE improves exponentially.	Although they take into account the traffic characteristics of OTT applications, they do not focus on specific applications.

Table 2. Identified gaps.

Summary

In this chapter there is a brief explanation of the necessary concepts for this undergraduate thesis such as mobile networks, traffic capturers, traffic generators, traffic classification, dataset and OTT applications, also giving a brief description of the LTE simulator that was chosen for this project. Following are the works related to traffic monitoring, LTE simulators, OTT applications, traffic classification and datasets.

Consequently, the gaps that the investigated works have with this undergraduate thesis are exposed, showing the contributions and taking into account the objectives proposed in the first chapter of this document.

CHAPTER 3

TRAFFIC GENERATORS COMPARISON AND INTEGRATION TESTS

In this chapter, the different traffic generators that were found, the characteristics that each of them has and why the integration could not be performed, since the tools that were found did not fulfill the project's objectives, are presenter. To give a brief context of what was looked for in the generators some of the main characteristics that were needed are mentioned as follows. The traffic generator should: have a client-server architecture, generate traffic of specific OTT applications, to be able to manipulate the protocols, be an open source software, among others. Further on, it will be explained in a specific way why the integration between the generator and the LTE simulator could not be done, and a small architecture and description of what was wanted with these tools will be given.

3.1 Traffic generators comparison

As it is known, traffic generators are very useful tools nowadays, because they allow to design, develop and test networks. They are usually implemented by network administrators in order to generate synthetic or real traffic in a controlled environment to check the status of networks, not only on their topologies but also on the elements that integrate them such as routers, firewall, IDS systems, etc.

These traffic generators have different categories according to their characteristics, some of them are shown below:

3.1.1 Reproduction engines

They take as a reference the previously captured traffic to send the packets through the network [13].

- **Tcpreplay:** It was created by Matt Undy in 1999, since then this tool has had a lot of collaborators over the years and it is a suite of GPLv3[42]. This generator edits and reproduces network traffic that was previously captured by tools such as Wireshark or tcdump. It allows to classify traffic as a client or server, rewrite packets from layer 2, 3 and 4 [13], and reproduce traffic back on the network through devices such as switches, routers, firewalls, etc. It is an open source software for UNIX OS and it can also work on Windows under Cygwin[43]. The traffic sent through Tcpreplay is a layer 2 traffic and this tool does not have a client – server architecture [40].
- **TCPivo:** It is an open source Linux based traffic replayer. It is a high-performance packet replay tool that employs mechanisms for managing trace files and accurate low-overhead timers to achieve high throughput and accuracy. It replays packets at a very high rate on an x86-based server and using low-latency kernel patches [41]. Its previous name was NetVCR and it is supported by the National Science Foundation under Grant EIA-0130344 and the generous donations of Intel Corporation [42].

3.1.2 High performance generators

These performance generators are generally used to test the network from end to end, working normally at high speed rates [13].

- **iPerf:** It was created by ESnet and Lawrence Berkeley National Laboratory under the BSD License. It is an open source program and it can run on various platforms like Windows, Linux, Android, MacOS X, among others [43]. It is mostly used for bandwidth tests, jitter delay and loss rate [13]. It allows to measure and adjust the performance of the network. It has the client-server functionality and can create data flows to measure the performance between the two ends in one or both directions and it can create different simultaneous connections between client and server [43].
- **BRUTE:** Its name means Brawny and RobUst Traffic Engine. BRUTE has been developed as a part of a project founded by the Italian Ministry of education, University and Research [44]. It is a packet traffic generator working at the Linux kernel level that guarantees controllable behavior [13]. It has been designed to produce high load of customizable network traffic. It can accurately generate traffic flows up to very high bit rates, achieving high precision and performance in the traffic generation [45]. To achieve this, it uses traffic patterns like constant bit rate, Poisson, Poisson Arrival of Burst, constant inter-departure time, etc [46]. It has a modular architecture and sends traffic for IPv4 and IPv6 networks.
- **Bruno:** It is a traffic generator based on BRUTE, which combines the flexibility of the software with the performance the hardware provides [46]. It gives higher

[42] *General Public License version 3.*
[43] *Is a large collection of GNU and Open Source tools which provide functionality similar to a Linux distribution on Windows* [103].

precision values, in both, performance and ipt[44] level [14]. It is also able to obtain a great number of simultaneous flows, unlike BRUTE, and it was used with an ENP2611 Radisys pci-board equipped with the Intel IXP2400NP [46].

- **KUTE:** Its acronym means Kernel-based Traffic Engine. It is a packet generator at the Linux kernel level. It can be configured for any type of package [14]. It has been developed as a high performance traffic generator and receiver, to be used mainly with Gigabit Ethernet [47]. It can be used to test the performance of hardware, routers, switches, etc. It can accurately measure high packet speeds.
- **Ostinato:** It was created by P. Srivats back in 2010. It was an open-source generator licensed under GNU GPLv3, but since 2016 its creator is charging for binary downloads. It generates and analyzes traffic network and it is mainly used to test the network upstream and downstream links. It is used through a graphical user interface or a Python API. The interface level receives and transmits statistics and monitoring rates in the real-time network [48]. Users can define various traffic flows through the interface and easily transmit it to the network interface [13]. Supports protocols like: ARP[45], IPv4[46], IPv6[47], UDP[48], ICMP[49], IGMP[50], MLD[51], HTTP, SIP[52], RTSP[53], NNTP[54], VLAN[55], Ethernet [48].

3.1.3 Model based generators

These traffic generators use different stochastic models for various statistics such as the distribution of the package size and the correlation. Its main approach is to know if the generated traffic follows the statistics that are being established by the model [13].

- **Mgen:** Its name means Multi-generator and is open source software developed by the Naval Research Laboratory (NRL) PROTocol Engineering Advanced Networking (PROTEAN) Research Group. It has the capacity to perform tests and performance measurements of IP networks using TCP[56] and UDP traffic, using a client-server architecture, supporting storage of the transport buffer, message counting and improvement of the payload. This tool generates traffic patterns in real time so that

[44] *It is a user space utility program that allows a system administrator to make configurations of the tables provided by the linux kernel firewall and the chains and rules it stores (iptable)*
[45] *Address Resolution Protocol*
[46] *Internet protocol version 4*
[4747] *Internet protocol version 6*
[48] *User Datagram Protocol*
[49] *Internet Control Message Protocol*
[50] *Internet Group Management Protocol*
[51] *Multicast Listener Discovery*
[52] *Session Initiation Protocol*
[53] *Real Time Streaming Protocol*
[54] *Network News Transport Protocol*
[55] *virtual LAN*
[56] *Transmission Control Protocol*

the network can be loaded in several ways. The generated traffic is received and saved and then it can be analyzed [49].

3.1.4 High-level and self-configurable generators

They are based on the highest level of the network traffic model and are able to automatically configure their parameters based on live measurements, therefore the traffic that is seen at the exit is almost identical to the traffic presented in the network [13].

- **Harpoon:** It was created by Joel E. Sommers in 2005. It is a flow level traffic generator. Harpoon uses a set of distribution parameters to generate artificial traffic flows with characteristics that exhibit the same qualities as the actual traffic measures, including spatial and temporal characteristics. Harpoon can be used to generate traffic for testing applications or protocols, or to test network switching hardware. It is a free software with a client-server architecture [50]. It does not generate traffic from specific applications, just TCP and UDP generic traffic.
- **Swing:** It was created by Kashi Venkatesh Vishwanath and Amin Vahdat back in 2005. It is a free software and it has only been design for Linux OS. Swing is a high-level generator that models the precise behavior of the user, network and application, based on the traffic observed in a single point, obtaining the distributions and the behavior of the network. It is a responsive traffic generator that accurately captures the packet interactions of a range of applications using a simple structural model. It is very configurable and allows the user to change the conditions of the network, the combination of applications and the characteristics of the application in order to generate new traffic (this generator replicates the protocol from different applications but it does not label as YouTube, WhatsApp, among others.). It has a client-server architecture and it is mostly used to emulate a trace between client and server within a link. Its authors implemented custom generators and listeners based on the application characteristics they extracted, they generate traffic for HTTP, UDP, FTP, SMTP, among others [51].
- **TMIX:** It is a generator capable of producing realistic synthetic traffic based on vectors, taking a packet header previously captured from a network link as input. It works for environments like ns[57]-2 or ns-3, GTNets software simulators and Linux and BSD[58]-based testbeds [52]. Its authors used a trace of TCP/IP header previously taken from different network links like campus networks, wide-area backbone networks, corporate intranets, wireless networks, etc., and constructed a model for all the TCP connection found in the network. TMIX takes a set of connection vectors and emulates the behavior at the socket level of the source application, like ftp or web servers, that originally created the connection in the network, this set of connection

[57] *The Network Simulator*
[58] *Berkeley Software Distribution*

vectors is what drives the traffic generation [53]. The output of TMIX is validated by the performance, RTT[59] and the size of the flows of the distributions [13].

- **LiTGen:** Its acronym means Light Traffic Generator. This tool plays application traffic like web, mail and P2P, based on real parameters extracted from sessions and objects characteristics [14]. It is an easy-to-use open loop traffic generator that statistically models wireless traffic per user and per application. This traffic generator is based on a user-oriented approach and a hierarchical model. This model is made up of several semantically significant levels each characterized by a specific traffic entity. For each traffic entity, a set of random variables are defined, whether related to time or size. LiTGEN generates traffic from higher-level entities (sessions) to lower-level entities (packages). It is used to generate traffic corresponding to different user applications. Traffic is generated for each user independently. The final synthetic trace is obtained by superimposing the synthetic traffic of all users and all applications. It is validated by two types of metrics: wavelet based analysis for scaling behavior of the packet arrival process and queuing model fitting for performance characteristics. In an operational network, these statistics can be derived from the knowledge of the operator of the customer's subscription services [54]. The download source of this generator wasn’t found.
- **D-ITG:** Its name means Distributed Internet Traffic Generator. It is a platform capable to produce Ipv4 and Ipv6 traffic by accurately replicating the workload of current internet applications such as VoIP, without labeling them. D-ITG is available on Linux, Windows OSX and FreeBSD and it is a completely free open source program. It was created by COMICS[60] group from the University of Napoli Federico II [55]. It is a network measurement tool capable of measuring the most common performance metrics (packet loss, delay) at the packet level. It can generate traffic by following stochastic models for the packet size and the time between outputs to mimic the protocol behavior at the application level. It can replicate traffic’s statistical properties of different known applications (telnet, VoIP, voice activity detection, RTP[61], DNS[62] and network games) and allows TCP, UDP, SCTP1[63], ACCP1[64] and ICMP protocols. The passive FTP-type protocol is also allowed to perform experiments in the presence of NAT[65]. D-ITG is capable of generating multiple unidirectional flows from many senders to many receivers [56].

[59] *Round trip time*
[60] *COMputer for Interaction and CommunicationS*
[61] *Real Time Transport Protocol*
[62] *Domain Name System*
[63] *Stream Control Transmission Protocol*
[64] *Adaptive congestion control protocol*
[65] *Network Address Translation*

3.1.5 Other generators

- **Tomahawk:** It was created by Brian Smith and TippingPiont Inc. under the Reciprocal Public License and it is an open source tool [57]. This generator is designed specifically to measure the insecurity of NIPS[66] systems and replay network traffic previously saved with Tcpdump. It sends packets containing some of the most known attacks and waits for the NIPS system to fail [13]. A single Tomahawk server can generate 200-450 Mbps of traffic and multiple servers can generate up to 1 Gbps, playing one or more network captures in pcap / tcpcump format. It can also perform assessments of NIPS behavior under extreme attack loads. A single PC can generate 25-50 thousand connections/second of network traffic and it is only available for Linux based OS [57].
- **Bit-Twist:** It was developed by Addy Yeow Chin Heng and it was released in 2006. Bit-Twist is a simple but powerful generator of Ethernet packages based on libpcap. This tool can regenerate traffic captured on a live network and is useful for simulating network traffic, testing firewall, IDS, IPS and solving various problems in the network sending multiple tracking files at once. It has a complete editor of tracking files having control over most of the fields in Ethernet, ARP, IP, ICMP, TCP and UDP headers with correction of automatic composition addition of the header. Adds the user's payload to existing packages and saves them into another trace file. Highly programmable, with proper handling it can become a very flexible tool to generate packages. Runs in different OS like MacOS X, Linux and Windows, and it is a free software [58].
- **PackETH:** Miha Jemec developed this generator under the GPL license in 2003 and it is an open source tool. It is a GUI and CLI[67] tool that generates Ethernet packets. Allows to send or create a possible packet or sequence of packets in the Ethernet link. Supports Ethernet II, ARP, IPv4, IPv6 protocols, user-defined network layer payload, UDP, TCP, ICMP, IGMP, user-defined transport layer payload, RTP, jumbo frames. It gives the time between packages, number of packages to send, sending with maximum speed, change parameters during the shipment, save the configuration in a file and upload from there, supports pcap format [59].
- **Rude / Crude:** It was developed in 2000 by Juha Laine, Sampo Saaristo and Rui Prior, and it is distributed under the GPLv2 license. It name stands by Real time UDP data emmiter / collector for rude. It is a small and flexible program that generates network traffic and can receive and register on another side of the network with crude. Currently these programs can only generate and measure UDP traffic [60].

[66] *Network-based intrusion prevention systems*
[67] *Command-line interface*

- **OTG (Openairinterface traffic generator):** It is a realistic tool for traffic generation at the packet level for scenarios of emerging applications. It is developed in C, which allows traffic to be generated with a restriction in real time and duration in real time. If OTG is directly connected to the user plane protocols, it is able to reproduce packet headers as in a real network protocol stack according to the user defined configuration. Both transmitter and receiver traffic statistics are generated and analyzed to derive the various measurements in the application-specific key performance indicators (performance, loss rate, latency, jitter). Its authors concentrate their efforts on applications like online gaming and M2M[68]. It was created in 2012 [61].
- **Pktgen:** Its name stands by Packet generator. It is a traffic generator based on software driven by the DPDK[69] fast packet processing framework powered by Intel's DPDK at 10Gbit wire rate traffic with 64 byte frames. It can act as a transmitter or as a receiver. It has a runtime environment to configure and start and stop traffic flows. It can show metrics in real time for several ports. It can generate packets in sequence when it iterates MAC[70], IP addresses or source ports or destinations. It can handle UDP, TCP, ARP, ICMP, MPLS[71], GRE[72] packets. It can be controlled remotely through a TCP connection. It can run command scripts to configure repeatable test cases [62].
- **Trafgen:** It was created by Daniel Borkmann for the Linux Netsniff-ng toolkit project, and it is maintained by Tobias Klauser. It is a multi-threaded network traffic generator, based on nmap mechanisms. It uses the socket interface of the Linux package that postpones full control over the data packets and the headers of the packages in the user space. It has a powerful packet configuration language, which is quite low and is not limited to particular protocols. This generator is very useful for many types of load tests in order to analyze and subsequently improve the behavior of systems in attack situations. It has the potential to perform fuzz tests, which means that it can generate a configuration of packets with random numbers in all or certain packet offsets that are generated recently when a packet is sent. The low level nature of trafgen makes it independent of protocols and that is why it is very useful in scenarios where stress tests are needed. The pcap traces can also be converted into a trafgen packet configuration [63].
- **Mausezahn:** As the generator above, is was created for Netsniff-ng and its author is Herbert Haas [64]. It is a high-level packet generator that can be run on a hardware-software device and comes with a CLI. Any diversity of packages can be elaborated. For example, it can be used to test the behavior of the network in extreme

[68] *Machine-to-Machine*
[69] *Data Plane Development Kit*
[70] *Medium access control*
[71] *Multiprotocol Label Switching*
[72] *Generic Routing Encapsulation*

circumstances (stress tests, malformed packages) or test the hardware-software devices for different attacks [13]. It is a free traffic generator written in C, which allows a user to send all possible packets. Its speed is close to the Ethernet limit and it depends on the hardware platform. The types of packages currently supported are ARP, BPDU[73], IP, TCP, RTP, and DNS with limited support for ICMP. It has been designed as a fast traffic generator, so it can quickly overwhelm a LAN[74] segment with undefined packets. It also supports security audits, it is very likely to create malicious packets, SVN[75] floods, specify port ranges and addresses, DNS and ARP poisoning [64].

- **Packet generator tool (NetScan Tools):** The purpose of this tool is to create TCP, UDP, ICMP, ARP, CDP[76] or RAW format packet or set of packets to send to target, then observe the target's response with a packet capturing tool like Wireshark. The tool can also playback previously captured packet files. It allows to have full control over the headers: ethernet source and destination MAC addresses, IP, TCP, UDP or ICMP header fields. It can send different types of packets in succession using scripting. It can also play back to previously saved packet capture file. This tool is not used to saturate an interface, and it is not a high speed traffic generator. The traffic created by this generator is unidirectional and uses WinPcap to send the packets [65].
- **Multistream packet generator and analyzer:** Multi Stream UDP / TCP Traffic Generator and Analyzer is a hardware-based Ethernet capable of generating multi-stream Ethernet traffic of varying packet length and also analyze the loopback traffic. It has the capability to Generate and Analyze up to 16 UDP streams of traffic of various packet lengths. This tool finds itself especially useful for end-to-end testing of 1 Gbps and 10 Gbps WAN[77] links [66].
- **TCPcopy:** It was created by NetEase. It is a replay tool that copies the live behavior of a TCP flow from an online server to a target server. It allows to modify TCP/IP headers and send this modify packets to the target server, so the target server would receive them as an online request form end-users. This is useful because networks administrator can do a live testing and reduce errors before the system is deployed. It is also able to overwhelm a server with traffic copying real-world data. This generator can be used for application based on HTTP, Memcached, POP3[78], SMTP, MySQL, etc. Its architecture has two objects, an online server, and a test server [67].
- **Caliper:** It is a tool that creates precise and responsive traffic. It was made for Gigabit Ethernet networks. Caliper takes the packet previously capture and send them into the network with a precise inter-transmission times, injects dynamically created

[73] *Bridge Protocol Data Units*
[74] *Local Area Network*
[75] *Supervised Visitation Network*
[76] *Cisco Discovery Protocol*
[77] *Wide Area Network*
[78] *Post Office Protocol*

packets of TCP and other protocols. Caliper is not a software itself and has been built on NetFPGA board to achieve high accuracy for the inter-arrival times of packages. It integrates with existing software tools like NetThreads that is a tool which allows to run threaded software on the NetFPGA. Both, Caliper and NetThreads are available as a free software [68].

3.1.6 Evaluation criteria and analysis

This part will evaluate the criteria chosen to select the generator that best suits the purposes that this project has. First, a description of the parameters that were taken into account will be given, justifying why they are important for this part. Then, a comparative table will be shown among all the generators that were investigated showing which ones meet the chosen criteria and which ones do not.

Initially, the first criteria chosen was that the generator should be open source, meaning that anyone could download it without having to pay or donate to the developers. This part is important because for this undergraduate work was not taken into account the purchase of any software, adding that when the programs are free, it is easier to manipulate and make changes if necessary, something that cannot be done when the software is proprietary. The second criteria was that the generator had a client/server architecture. This was taken into account since the LTE simulator, presented in chapter 2, consists of 6 different virtual machines in which the different modules of a physical LTE network are simulated. This leads to the existence of a client part, a server part, and a module responsible for carrying out the communication of the respective modules (the simulator will be explained in detail in the next section). Without a traffic generator that cannot be added to this kind of architecture, the data passing through the network would remain on a single virtual machine and there would not be a correct simulation of the communication performed by the clients and the servers within an LTE network, therefore the simulator would not be used correctly.

The next criteria is that the generator should be capable of generating multiple flows within the network. As mentioned before, a client/server architecture is needed, hence the fact that the generator must be able to create several clients and several servers exchanging data simultaneously is needed. This is important because it is the way an LTE network behaves in real life. Not only one user is connected and not only a single server fulfills all the user’s requests, on the contrary, several thousand users are connected to the network and constantly exchange data packets and this is the objective considered in this undergraduate thesis.

The next criteria, and perhaps the most important criteria of all, is that the generator should be capable of creating traffic from specific OTT applications. As mentioned earlier, now the Internet is changing very quickly and that involves the creation and development of different and new applications that come to the market. A part of this are the OTT applications. For this reason, a generator that is able to create the flow of a specific OTT application, ensuring that such flow belongs to that exact application, is needed. This means, creating a flow for Spotify or Skype while being able to identify which packets belong to which application without an invasive process like DPI.

The last criteria taken into account has to do with the documentation that could be found for each of the generators. This part is quite important because it lets know how the tool works and how it is installed, letting see the objectives for which the developers created the different generators. Without this documentation or installation guide the generators cannot be properly manipulated and there would be no clear idea of what each of them can do.
Taking these criteria into account, table 3 presents a comparison with all the generators that were researched in the previous section are shown below.

Generator	**Open source software**	**Client/Server architecture**	**Multiple flows**	**Specific application traffic**	**Documentation**
TCPreplay	✓	✗	✗	✗	✓
TCPivo	✓	✗	✗	✗	✓
Iperf	✓	✓	✓	✗	✓
Brute	✓	✗	✓	✗	✓
Bruno	✓	✗	✗	✗	✗
Kute	✓	✗	✗	✗	✓
Ostinato	✗	✗	✓	✗	✓
Mgen	✓	✓	✗	✗	✓
Harpoon	✓	✓	✗	✗	✓
Swing	✓	✓	✗	✗	✓
TMIX	✓	✗	✗	✗	✗
LiTGen	✓	✓	✗	✗	✗
D-ITG	✓	✓	✗	✗	✓
Tomahawk	✓	✗	✗	✗	✓

Bit-Twist	✓	✗	✓	✗	✓
PackETH	✓	✗	✗	✗	✓
Rude/Crude	✓	✗	✗	✗	✗
OTG	✓	✗	✗	✗	✗
Pktgen	✓	✗	✗	✗	✓
Trafgen	✓	✗	✗	✗	✓
Mausezahn	✓	✗	✗	✗	✓
Packet generator Tool	✓	✗	✗	✗	✓
Multistream packet generator and analyzer	✗	✗	✗	✗	✗
TCPcopy	✓	✗	✗	✗	✗
Caliper	✗	✗	✗	✗	✗

Table 3. Qualification of the generators characteristics.

As it can be seen in table 3, almost all generators are open source, something that is quite suitable for what is needed in the project. On the contrary, It can be seen that three of them do not meet this criteria; Ostinato is a software that was initially free, but for a couple of years its developer has been asking for donations for every download that is made of the last version that is released, Caliper is a generator that has two parts, a part in software and a part in hardware. According to its developers, the software is completely free, but the fact of being based on a hardware element makes it generate an additional cost; Finally the Multistream packet generator and analyzer is a generator based entirely on hardware, which must be purchased for an additional cost.

The next evaluation criteria that can be seen in the table is the client/server architecture that the generators have. Here it can be seen that the list of generators is reduced to six, concluding that most generators are only used to inject traffic into networks for the sole purpose of testing them, either to repair damage or failure or to develop new networks.

For generators that are capable of creating multiple data streams the list is reduced to four. Almost all generators are able to create a single thread of information exchange within the network, being unable to send different data streams at the same time.

For the most important criterion, as mentioned above, it can be noted that there is no generator that supplies the desired feature, which is that the tool should be capable of generating traffic from specific OTT application. This leads to the conclusion that there is no traffic generator that can meet the objective of the project.

Finally, the documentation found for the generators is very diverse. Many of them are generators that are on the market, they have their own web pages, help forums and manuals. However many others despite having articles that explain and appreciate the vision of their developers, do not have pages on the Internet where it can be reviewed or found the download links. For some, there needs to be a direct communication with the developers, in order to be able to have the software. Others are obsolete or have not been worked on in recent years. From others a reliable documentation couldn’t be found and another part of the generators were not developed by the same people, meaning that the community in general can contribute to the code creating new features, but without having the proper support for failures and functioning of the tool.

3.2 Integration and Tests

This part talks about the integration and the tests that were made to be able to fulfill the objectives of the project. First a brief introduction is given to the LTE simulator that was chosen; this simulator was taken into account because it is the simulator that has been working previously in the network of the University of Cauca. Next the revision of the generators is presented and how they were discarded one by one to be able to find the ideal generator that supplied the objective of the thesis.

As it was said before, for this project the LTE simulator mentioned in chapter 2, section 2.1.9 was taken into account. As it could be seen there, this simulator consists of 6 modules, RAN, MME, HSS, SGW, PGW and Sink. These 6 modules were installed in 6 different virtual machines on the Linux OS Ubuntu 14.04. This simulator has a traffic generator called Iperf3 installed in the RAN and Sink module, and it has the Wireshark sniffer installed in the PGW machine.

Before starting, to use the modules on each machine, it must be ensured that there is a connection between them by sending the PING[79] command with the corresponding IP

[79] *Command that allows the verification of the state of a connection between two hosts*

addresses. Then, inside of each virtual machine, there is a folder with the name corresponding to the module of the machine. In that folder there is a .cpp file (for example mme.cpp), that file must be edited to add the IP addresses corresponding to the links a module has with the rest of the others (see figure 2), and thus guarantee the correct data flow between them.

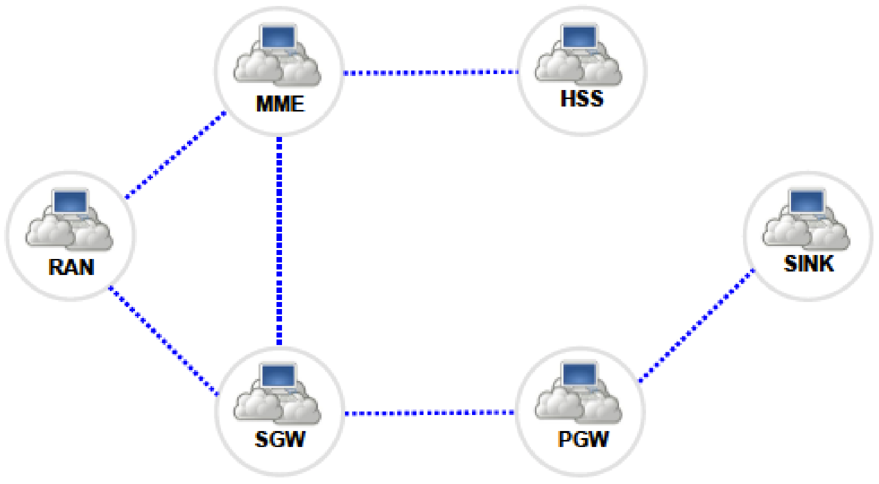

Figure 2. Connection between virtual machines.
[69]

This simulator generates two types of traffic, control traffic, and data traffic.

The control traffic occurs when the UE's in the RAN module continuously make the process of connection and disconnection to the network to create the control traffic. This traffic is the one that goes through the RAN, MME and HSS modules, remember that MME and HSS are the modules that store user information, such as location, connection and user authentication. Despite this being an important part, control traffic is not of great importance for this project. Data traffic, on the other hand, is where this work is centered.

For this part, the LTE simulator has a traffic generator called Iperf3, with which TCP data can be sent with certain bandwidth and a certain time duration. To begin with the exchange of traffic, it must be launch the iperf3 server in the Sink module and the client in the RAN module, along with the commands to initialize the other modules in the simulator. In order to capture this traffic, Wireshark was used, which is installed in the PGW module. It is important to mention that the PGW module is an intermediary between the RAN and Sink, and through it the information that is uploaded and downloaded in the network will go through it.

In spite of all the above, in order to fulfill the first objective of this project, an open source generator is needed, it has to have a client/server architecture, that is capable of generating multiple data flows in the network and more important, that it is capable of generating traffic of specific OTT applications.

In the previous section, it can be seen that there was a review of 25 generators, all with different characteristics. The first generators that were taken into account were those that were open source, and with a client/server architecture, these were: Iperf, Mgen, Harpoon, Swing, and D-ITG. The other generators were not taken into account since they do not have a client/server architecture, besides two of them (Multistream packet generator and analyzer and Caliper) are based on hardware elements, they were dismissed because these hardware elements have to be bought and as mentioned before, one of the criteria is that they do not have an additional cost, furthermore they work only injecting traffic into the network. For some of them, the download source was not found and there was not much information about them. Others are based on information previously collected with traffic capture tools such as Tcpdump, to replay among the network, some generators were focused on the precision with which the packets are sent, such as arrival times and packet loss. The others were more focused on testing networks for its design, development, and deployment, also counting those used for Intrusion Detection Systems and Ostinato presents its own graphic user interface and it is a non-free generator.

Continuing with the generators that were chosen, Mgen and Harpoon were the first to be dismissed from this part, because they only are able to generate TCP and UDP traffic. D-ITG and Swing seemed the most suitable for this work since these generators can send data traffic such as HTTP, SMTP or RTP, which are the protocols used for browsing or VoIP. However, this was not enough, because although they allow to choose the application protocol that is being sent, there is no way to label the traffic as generated by an application, that is, it cannot be ensure that a flow created with these generators is from an OTT application like WhatsApp or YouTube.

Therefore in order to find a traffic generator that fulfilled the project's needs, the following alternative was found. The Wireshark sniffer was used along with nDPI[80] library, a plugin that helps to label the captured packages by application, this plugin is part of the ntopng tool that will be introduced in the next chapter. nDPI helps to control and classify traffic accurately because it analyzes both, the content and the header of the packets, looking for any signature to specify which application data goes through the network (DPI using Quotient Filter). With the help of this tool, applications like WhatsApp, YouTube, Spotify, Skype and Google could be identified and used for the purpose of the project. In figure 3 it can be seen an architecture of how this tools integrate with each other and what is the output of the system.

[80] *Ntopng Deep Packet Inspection*

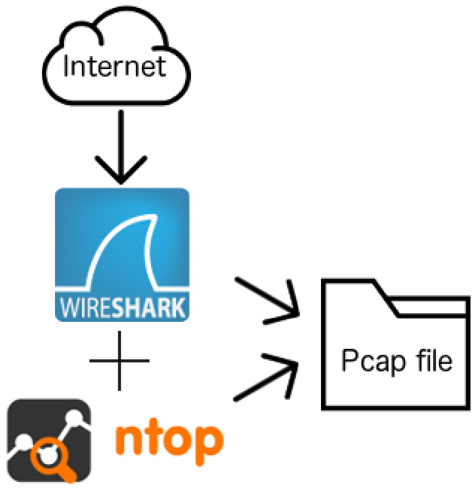

Figure 3. How Wireshark and nDPI work.

After this, a program made with Python programming language named pcap_to_ditg was found. This program allows to pass flows from pcap files to D-ITG. However, this process could not be done since the tool did not work as easily as it is presented in its documentation (). In order to know what was causing the failure, the core script from pcap_to_ditg should be understood and modified and it was necessary to understand how to integrate a new plugin to D-ITG, which was not a trivial task.

Furthermore, since the traffic generator implemented by pcap_to_ditg was D-ITG, it was necessary to perform a modification within the core of the LTE simulator which had iPerf3 as its only traffic generator. To achieve such change, all the scripts of each of the six machines had to be modified, knowing exactly where to perform the traffic generator replacement. The previously described tasks would require an important amount of time and effort delaying the course of the investigation without guaranteeing that what was needed within the project would be fulfilled. Therefore another alternative was needed.

With this in mind, considering that iPerf3 was the traffic generator within the LTE simulator, another alternative was taken into account. With this generator the collected information, between the communication established by clients and servers, could be ordered by server, indicating that the traffic going to a specific server belongs to a specific application. Such possibility would allow to differentiate the traffic generated by each OTT application. However, it is an inadequate alternative since this tool is only capable of generating generic traffic without the possibility of protocol manipulation nor the data exchanged in the traffic that is being sent. Therefore, it is not possible to ensure that the generated traffic closely simulates a communication between a client and an OTT application.

As a final alternative, using a replay generator such as TCPreplay was considered since this tool allows to replay the traffic from specific OTT applications previously captured and identified with Wireshark and nDPI. This tool reproduced the pcap files without any difficulty. However, with this approach the tool does not present a client/server architecture, therefore it does not fulfill the established evaluation criteria.

After all the previous integration tests, it was concluded that there is not a generator that meets the needs of this project and therefore in order to fulfill the first specific objective another solution had to be carried out.

In the next chapter, an alternative capable of fulfilling the proposed objectives and proceed with the research work will be given.

Summary

This chapter presented the research done for the different existing traffic generators. It was found that these generators are divided into different categories such as reproduction engines, high performance generators, generators based on models, and high level and self-configurable generators, also finding others that were not within these categories.

These generators serve different purposes and were created to meet goals needed by their authors. Many of them were created for network testing, their design, configuration and deployment. Others were created to test IDS’s, which send attacks through networks and enables the network administrators to have the ability to see the faults and errors found, to fix them. Other generators based on hardware systems were also found, which are able to send packets with a much higher speed than software-based generators can send. Some others were focused on the precision of the packages, so the research for these generators is based on the jitter, the inter-arrival times of the packets, the packet loss rate, among others.

What was found was that all the generators send generic traffic through the networks and none, until now, is specialized in sending traffic of specific OTT applications such as WhatsApp, YouTube, Netflix, Skype, etc.

Finally, a description of the simulator with the traffic generator integration is made and the justification of why the project could not fulfill its first specific objective is given.

CHAPTER 4

SYNTHETIC GENERATOR OF TRAFFIC FROM OTT APPLICATIONS AND TESTS

This chapter presents in detail the synthetic generator of traffic from OTT applications traffic (SYNGEN) that had to be developed, taking into account that in the previous chapter it was concluded that none of the investigated traffic generation tools fulfilled the expected requirements to carry out this undergraduate thesis.

First, there is a brief description of the dataset that was taken as a model for the creation of SYNGEN and why this dataset is useful for the purposes of this undergraduate thesis. Subsequently, the different environments implemented to obtain this dataset are presented. Consequently, a description about the process of generating traffic of OTT applications in a synthetic way is given, and finally the different generated datasets and the cleaning process applied to them are explained.

4.1 Dataset description

In order to be able to generate synthetic traffic from OTT applications, first it is mandatory to have a dataset that serves as a model from which the information can be extracted. This dataset must have labeled flows of different OTT applications, i.e., for each flow that is stored, it necessary to be certain that this flow belongs to a given application. The dataset that was chosen, was taken from the paper [70]. In the paper the authors explain that the dataset was captured in one of the routers of the Universidad del Cauca during 6 days in 2017, collecting all the information that came from laptops, tablets and smartphones connected to the network.

In the paper they expose the process of data collection as illustrated in the following figure, showing the architecture of the system to obtain the desired dataset:

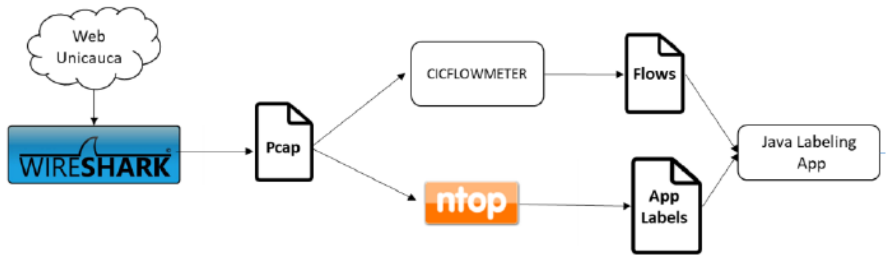

Figure 3. Process to obtain the model dataset.
[71]

The architecture shown in the previous figure shows clearly the programs and files that were used to fulfill the purpose of creating a dataset in the proper way. To begin with, a computer, where Wireshark was installed, was taken into a network core as a packet capturer. All data packets captured from IP traffic were stored and saved in a set of files with the extension .pcap. Subsequently, and having in mind that the packets captured by the Wireshark tool do not hold the information of the application that is being consumed on the flow, this files were taken as input to the CICFlowMeter and ntopng programs separately. CICFlowMeter is a tool that generates bi-directional IP flows from the provided pcap file, generating a total of 85 attributes, where the 6 most important attributes are: FlowID, SourceIP, SourcePort, DestinationIP, DestinationPort and Protocol.

Later, and considering that the key purpose of the authors was to be able to identify the OTT application being consumed on each IP flow (e.g. YouTube, WhatsApp, Skype, among others). With the help of ntopng, a tool capable of taking the pcap files as input, and then obtain the layer 7 protocols using an embedded library called nDPI, which performs a deep inspection on the packages, such purpose was achieved. Consequently, those two files that are obtained separately by each software application, are passed through the next software, the Java Labeling App which labels each IP flow. This application takes the two files and compares them through 4 attributes, source and destination IP addresses, and source and destination ports. Taking into account that these 4 attributes are the same in both files, this software adds two new attributes to the CICFlowMeter file: an application code, which is an integer from 0 to 226 and the name of the application that is being consumed on that IP flow.

The most important thing to note is that the file obtained, after the process done by the three software applications (CICFlowMeter, ntopng and Java Labeling App), is a dataset suitable for the objectives in this research project, since it contains IP flows labeled with their respective OTT applications which can serve as a model for the construction of the synthetic traffic generator.

4.1.1 Implemented software applications

This subsection gives a brief technical description of the software tools shown in the figure 3.

Wireshark: It is a free software package analyzer, under the GNU GPLv2 license. It captures the traffic that runs over the network from a host and is normally used for network troubleshooting, analysis, software development and communications protocol and also for academic purposes. It runs on almost all OS like Windows, Linux and MacOS [72].

CICFlowMeter: It is a network traffic flow generator and analyzer created by the Canadian Institute for Cybersecurity at the University of New Brunswick. It is a tool capable of creating bi-directional flow, that is, in forward direction (source to destination) and in backward direction (destination to source), providing more than 80 statistical attributes such as duration, number of packets, number of bytes, among others, doing this in both directions. To the exit it delivers a file with CSV format, where there are 6 columns labeled for each flow, flowID, SourceIP, DestinationIP, SourcePort, DestinationPort and protocol [73].

Ntopng: Is a more advanced version of ntop. It works on Linux, MacOS and Windows platforms. Among its features are: display network traffic in real time, monitor and report performance in real time (network latencies and applications, Time to and back (RTT), TCP statistics, such as retransmissions, off-order packets, lost packets, and transmitted bytes and packets). With the help of ntopng it was possible to obtain the layer 7 protocol (e.g., YouTube, Facebook, WhatsApp, etc.), since with CICFlowMeter this feature is not available. For this, DPI is used, through the nDPI tool embedded in ntopng [74].

Java Labeling Application: This application was developed in the article (Reference). What it does is a comparison of the files obtained from CICFlowMeter and ntopng, this comparison is done with four different attributes (source and destination IP addresses and source and destination ports). If these conditions in the two files are equal, then two more fields are added to the CICFlowMeter file, the application code which is an integer between 0 and 226, and the name of the application to which that stream belongs be it YouTube, Google, among others [71].

4.1.2 Dataset attributes description

For this section, CRISP-DM[81] methodology [75] was used, which is a methodology described as a model of hierarchical processes that has four groups of tasks: phase, generic task, task

[81] *Cross Industry Standard Process for Data Mining*

specialized, and process instance; and which is divided into some phases of the reference model, they are: Business understanding, Data understanding, Data preparation, Modeling, Evaluation and Deployment [76].

As mentioned before, the dataset obtained from Wireshark was passed through CICFlowMeter and ntopng separately. Then these two files were compared with Java Labeling Application and after this step it was obtained a single file where were the attributes analyzed with CICFlowMeter, adding the code and name of the application. This is part of business understanding, since it is known what is needed to achieve the objective. Next, the data understanding phase of the CRISP-DM methodology takes place and a brief description of the attributes that were obtained from the captured data stream is provided [70].

- **Flow.ID:** a flow identifier following the next format: SourceIP-DestinationIP-SourcePort-DestinationPort-TransportProtocol
- **Source.IP:** The source IP address of the flow.
- **Source.Port:** The source port number
- **Destination.IP:** The destination IP address.
- **Destination.Port:** The destination port number.
- **Protocol:** The transport layer protocol number identification (i.e.,TCP = 6, UDP = 17).
- **Timestamp:** The instant the packet was captured stored in the next date format: Dd/mm/yyyy HH:MM:SS
- **Flow.Duration:** The total duration of the flow
- **Total.Fwd.Packets:** The total number of packets in the forward direction.
- **Total.Backward.Packets:** The total number of packets in the backward direction.
- **Total.Length.of.Fwd.Packets:** The total quantity of bytes in the forward direction obtained from all the flow (all the packets transmitted). This is obtained from the Total Length field stored on the packets header.
- **Total.Length.of.Bwd.Packets:** The total quantity of bytes in the backward direction obtained from all the flow (all the packets transmitted). This is obtained from the Total Length field stored on the packets header.
- **Fwd.Packet.Length.Max:** The maximum value in bytes of the packets length in the forward direction.
- **Fwd.Packet.Length.Min:** The minimum value in bytes of the packets length in the forward direction.
- **Fwd.Packet.Length.Mean:** The mean value in bytes of the packets length in the forward direction.
- **Fwd.Packet.Length.Std:** The standard deviation in bytes of the packets length in the forward direction.

- **Bwd.Packet.Length.Max:** The maximum value in bytes of the packets length in the backward direction.
- **Bwd.Packet.Length.Min:** The minimum value in bytes of the packets length in the backward direction.
- **Bwd.Packet.Length.Mean:** The mean value in bytes of the packets length in the backward direction.
- **Bwd.Packet.Length.Std:** The standard deviation in bytes of the packets length in the backward direction.
- **Flow.Bytes.s:** The number of bytes per second in the flow.
- **Flow.Packets.s:** The number of packets per second in the flow.
- **Flow.IAT.Mean:** The mean value of the inter-arrival time of the flow (in both directions).
- **Flow.IAT.Std:** The standard deviation of the inter-arrival time of the flow (in both directions).
- **Flow.IAT.Max:** The maximum value of the inter-arrival time of the flow (in both directions).
- **Flow.IAT.Min:** The minimum value of the inter-arrival time of the flow (in both directions).
- **Fwd.IAT.Total:** The total Inter-arrival time in the forward direction.
- **Fwd.IAT.Mean:** The mean inter-arrival time in the forward direction.
- **Fwd.IAT.Std:** The standard inter-arrival time in the forward direction
- **Fwd.IAT.Max:** The maximum value of the inter-arrival time in the forward direction
- **Fwd.IAT.Min:** The minimum value of the inter-arrival time in the forward direction
- **Bwd.IAT.Total:** The total Inter-arrival time in the backward direction.
- **Bwd.IAT.Mean:** The mean inter-arrival time in the backward direction.
- **Bwd.IAT.Std:** The standard inter-arrival time in the backward direction.
- **Bwd.IAT.Max:** The maximum value of the inter-arrival time in the backward direction.
- **Bwd.IAT.Min:** The minimum value of the inter-arrival time in the backward direction
- **Fwd.PSH.Flags:** The number of times the packets sent in the flow had the pushing flag bit set as 1 in the forward direction. The Pushing flag allows to send information immediately without filling all the buffer size from a packet, notifying the receptor to pass the packet to the application at once, it is very useful for real time applications.
- **Bwd.PSH.Flags:** The number of times the packets sent in the flow had the PSH (pushing) flag bit set as 1 in the backward direction.
- **Fwd.URG.Flags:** The number of times the packets sent in the flow had the URG (Urgent) flag bit set as 1 in the forward direction. The URG flag is used to inform a receiving station that certain data within a segment is urgent and should be prioritized.

If the URG flag is set, the receiving station evaluates the urgent pointer, a 16-bit field in the TCP header. This pointer indicates how much of the data in the segment, counting from the first byte, is urgent.

- **Bwd.URG.Flags:** The number of times the packets sent in the flow had the URG (Urgent) flag bit set as 1 in the backward direction.
- **Fwd.Header.Length:** The header length of the packets flow in the forward direction.
- **Bwd.Header.Length:** The header length of the packets flow in the backward direction.
- **Fwd.Packets.s:** The number of packets per second in the forward direction.
- **Bwd.Packets.s:** The number of packets per second in the backward direction.
- **Min.Packet.Length:** The minimum length of the packets registered in the flow (both forward and backward directions).
- **Max.Packet.Length:** The maximum length of the packets registered in the flow (both forward and backward directions).
- **Packet.Length.Mean:** The mean value of the length of the packets registered in the flow (both forward and backward directions).
- **Packet.Length.Std:** The standard deviation of the length of the packets registered in the flow (both forward and backward directions).
- **Packet.Length.Variance:** The variance of the length of the packets registered in the flow (both forward and backward directions).
- **FIN.Flag.Count:** The number of times the packets sent in the flow had the FIN flag bit set as 1. In the normal case, each side terminates its end of the connection by sending a special message with the FIN (finish) bit set. This message, sometimes called a FIN, serves as a connection termination request to the other device, while also possibly carrying data like a regular segment. The device receiving the FIN responds with an acknowledgment to the FIN to indicate that it was received. The connection as a whole is not considered terminated until both sides have finished the shutdown procedure by sending a FIN and receiving an ACK.
- **SYN.Flag.Count:** The number of times the packets sent in the flow (in both directions) had the SYN (Synchronize) flag bit set as 1. The SYN (Synchronize) flag is the TCP packet flag that is used to initiate a TCP connection. A packet containing solely a SYN flag is the first part of the "three-way handshake" of TCP connection initiation. It is responded to with a SYN-ACK packet. Packets setting the SYN flag can also be used to perform a SYN flood and a SYN scan.
- **RST.Flag.Count:** The number of times the packets sent in the flow (in both directions) had the RST (Reset) flag bit set as 1 - (An RST says reset the connection. It must be sent whenever a segment arrives which apparently is not intended for the current connection - FIN says, "I finished talking to you, but I'll still listen to everything you have to say until you're done" (Wait for an ACK) RST says, "There is no conversation. I am resetting the connection!").

- **PSH.Flag.Count:** The number of times the packets sent in the flow (in both directions) had the PSH (Pushing) flag bit set as 1.
- **ACK.Flag.Count:** The number of times the packets sent in the flow (in both directions) had the ACK (Acknowledged) flag bit set as 1. To establish a connection, TCP uses a three-way handshake. Before a client attempts to connect with a server, the server must first bind to and listen at a port to open it up for connections: this is called a passive open. Once the passive open is established, a client may initiate an active open.
- **URG.Flag.Count:** The number of times the packets sent in the flow (in both directions) had the URG (Urgent) flag bit set as 1.
- **CWE.Flag.Count:** The number of times the packets sent in the flow (in both directions) had the CWR (Congestion Window Reduced) TCP flag set as 1. During the synchronization phase of a connection between client and server, the TCP CWR and ECE (Explicit Congestion Notification - Echo) flags work in conjunction to establish whether the connection is capable of leveraging congestion notification. In order to work, both client and server need to support ECN (Explicit Congestion Notification). To accomplish this, the sender sends a SYN packet with the ECE and CWR flags set, and the receiver sends back the SYN-ACK with only the ECE flag set. Any other configuration indicates a non-ECN setup.
- **ECE.Flag.Count:** The number of times the packets sent in the flow (in both directions) had the ECE (Explicit Congestion Notification Echo) TCP flag set as 1.
- **Down.Up.Ratio:** Download and upload ratio.
- **Average.Packet.Size:** The average size of each packet. It is important to notice that Packet Length specify the size of the whole packet including the header, trailer and the data that send on that packet. But Packet Size specify only the size of the header on the packet.
- **Avg.Fwd.Segment.Size:** The average segment size observed in the forward direction. A TCP segment is the Protocol Data Unit (PDU) which consists of a TCP header and an application data piece which comes from the upper Application Layer. Transport layer data is generally named as segment and network layer data unit is named as datagram but when UDP is used as transport layer protocol the data unit is called UDP datagram since the UDP data unit is not segmented (segmentation is made in transport layer when TCP is used).
- **Avg.Bwd.Segment.Size:** Average Segment size observed in the backward direction.
- **Fwd.Header.Length.1:** The header length of the packets flow in the forward direction. This attribute has the exact same values than the attribute Fwd Header Length, hence it can be a bug on the CICFlowMeter software.
- **Fwd.Avg.Bytes.Bulk:** The average number of bytes bulk rate in the forward direction. Bulk data transfer is a software-based mechanism designed to move large

data file using compression, blocking and buffering methods to optimize transfer times.

- **Fwd.Avg.Packets.Bulk:** Average number of packets bulk rate in the forward direction.
- **Fwd.Avg.Bulk.Rate:** Average number of bulk rate in the forward direction.
- **Bwd.Avg.Bytes.Bulk:** Average number of bytes bulk rate in the backward direction.
- **Bwd.Avg.Packets.Bulk:** Average number of packets bulk rate in the backward direction.
- **Bwd.Avg.Bulk.Rate:** Average number of bulk rate in the backward direction.
- **Subflow.Fwd.Packets:** The average number of packets in a subflow in the forward direction. The core idea of multipath TCP is to define a way to build a connection between two hosts and not between two interfaces (as standard TCP does). In standard TCP, the connection should be established between two IP addresses. Each TCP connection is identified by a four-tuple (source and destination addresses and ports). Given this restriction, an application can only create one TCP connection through a single link. Multipath TCP allows the connection to use several paths simultaneously. For this, Multipath TCP creates one TCP connection, called subflow, over each path that needs to be used. The detailed protocol specification is provided in RFC 6824
- **Subflow.Fwd.Bytes:** The average number of bytes in a subflow in the forward direction.
- **Subflow.Bwd.Packets:** The average number of packets in a subflow in the backward direction.
- **Subflow.Bwd.Bytes:** The average number of bytes in a subflow in the backward direction.
- **Init_Win_bytes_forward:** The total number of bytes sent in the initial window in the forward direction. TCP uses a sliding window flow control protocol. In each TCP segment, the receiver specifies in the receive window field the amount of additionally received data (in bytes) that it is willing to buffer for the connection. The sending host can send only up to that amount of data, before it must wait for an acknowledgment and window update from the receiving host.
- **Init_Win_bytes_backward:** The total number of bytes sent in the initial window in the backward direction.
- **act_data_pkt_fwd:** Count of packets with at least one byte of TCP data payload in the forward direction.
- **min_seg_size_forward:** Minimum segment size observed in the forward direction.
- **Active.Mean:** The mean time a flow was active before becoming idle.
- **Active.Std:** Standard deviation time a flow was active before becoming idle.
- **Active.Max:** Maximum time a flow was active before becoming idle.
- **Active.Min:** Minimum time a flow was active before becoming idle.
- **Idle.Mean:** Mean time a flow was idle before idle before becoming active.

- **Idle.Std:** Standard deviation time a flow was idle before becoming active.
- **Idle.Max:** The maximum time a flow was idle before becoming active.
- **Idle.Min:** The minimum time a flow was idle before becoming active.
- **Label:** The state of the flow (benign or malign).
- **L7Protocol:** This attribute represents the code number of the layer 7 protocol as obtained from nDPI in Ntopng application. It is a number that varies from 0 to 226 (e.g., 0 is labeled as Unknown application).
- **ProtocolName:** This attribute is the objective class of the dataset. It holds the application name following the code number stored in the L7Protocol attribute (e.g., YouTube, Yahoo, Facebook, etc.).

4.2 Synthetic generators

In this section, it will be explained the processes that were made to achieve the development of the synthetic generators that allowed the creation of the dataset needed in this undergraduate thesis.

As explained in the previous section, a dataset was obtained that serves as a model from which the data will be extracted and will allow to start generating synthetic flows for each OTT application. However, in the dataset there are many internet applications and flows that do not necessarily belong to OTT applications, and there are also many flows of diverse OTT applications. But, as mentioned in one of the objectives of this undergraduate thesis, it is necessary to create traffic of specific applications such as: video, messaging, browsing, audio and calls. Bearing this in mind, the dataset was analyzed and a specific application was chosen for each type of application exposed in the objective, these applications were: YouTube, WhatsApp, Google, Spotify and Skype.

Having chosen the applications, the dataset that was used as a model is taken and passed to Rstudio which is an IDE[82] developed for R; R is a programming language that provides a variety of statistical and graphical techniques [77]; When loading it in Rstudio, a brief script is made and the different flows of the selected applications, mentioned in the previous paragraph, are separated into different datasets.

Next, each dataset was taken and the attributes they had were analyzed in detail, making the decision not to model some of the attributes. By doing this, another phase of the CRISP-DM methodology mentioned above takes place, this would be the phase of data preparation.

[82] *Integrated Development Environment*

This decision was made since many of the attributes did not provide valuable information some of them being zero at all times. However, this does not mean that these attributes are not important in the actual data flows. The attributes that were discarded were as follows: Fwd.PSH.Flags, Bwd.PSH.Flags, Fwd.URG.Flags, Bwd.URG.Flags, FIN.Flag.Count, SYN.Flag.Count, RST.Flag.Count, PSH.Flag.Count, ACK.Flag.Count, URG.Flag.Count, CWE.Flag.Count, ECE.Flag.Count, Fwd.Avg.Bytes.Bulk, Fwd.Avg.Packets.Bulk, Fwd.Avg.Bulk.Rate, Bwd.Avg.Bytes.Bulk, Bwd.Avg.Packets.Bulk, Bwd.Avg.Bulk.Rate, this decision will be explained in section 4.3.2 from this chapter; additionally the attribute Flow.ID is discarded, as this is an identifier set by the CICFlowMeter tool; Timestamp is also a deleted, because it refers to the date and time when the information was collected, i.e., the date and time of the days and months of 2017 when Wireshark collected the information, which is irrelevant for this thesis purposes; Protocol is an attribute that is not taken into account because all the flows of the different datasets are given with the TCP protocol; Label is an attribute that says if the flow that goes over the network is benign or malign, in this case, for the applications mentioned above are always catalogued as benign flows, so it is not an attribute that provides information for the characterization of traffic; Fwd.Header.Length.1 is removed, because this attribute has the same value as Fwd.Header.Length and may be a CICFlowMeter error. Finally the L7Protocol, as it was said in the attributes definition, gives an integer number between 0 and 226 that represents each identified application with a different number. This attribute is not taken into account since in the traffic classification that is done for the validation of the dataset, this attribute would provide the answer to the machine learning algorithm affecting the classification process.

Taking into account the attributes mentioned above and why they are not going to be modeled, a total of 63 attributes were chosen for statistical modeling which are presented as follows: Source.IP, Source.Port, Destination.IP, Destination.Port, Flow.Duration, Total.Fwd.Packets, Total.Backward.Packets, Total.Length.of.Fwd.Packets, Total.Length.of.Bwd.Packets, Fwd.Packet.Length.Max, Fwd.Packet.Length.Min, Fwd.Packet.Length.Mean, Fwd.Packet.Length.Std, Bwd.Packet.Length.Max, Bwd.Packet.Length.Min, Bwd.Packet.Length.Mean, Bwd.Packet.Length.Std, Flow.Bytes.s, Flow.Packets.s, Flow.IAT.Mean, Flow.IAT.Std, Flow.IAT.Max, Flow.IAT.Min, Fwd.IAT.Total, Fwd.IAT.Mean, Fwd.IAT.Std, Fwd.IAT.Max, Fwd.IAT.Min, Bwd.IAT.Total, Bwd.IAT.Mean, Bwd.IAT.Std, Bwd.IAT.Max, Bwd.IAT.Min, Fwd.Header.Length, Bwd.Header.Length, Fwd.Packets.s, Bwd.Packets.s, Min.Packet.Length, Max.Packet.Length, Packet.Length.Mean, Packet.Length.Std, Packet.Length.Variance, Down.Up.Ratio, Average.Packet.Size, Avg.Fwd.Segment.Size, Avg.Bwd.Segment.Size, Subflow.Fwd.Packets, Subflow.Fwd.Bytes, Subflow.Bwd.Packets, Subflow.Bwd.Bytes, Init_Win_bytes_forward, Init_Win_bytes_backward, act_data_pkt_fwd, min_seg_size_forward, Active.Mean, Active.Std, Active.Max, Active.Min, Idle.Mean, Idle.Std, Idle.Max, Idle.Min, ProtocolName.

Bearing in mind the attributes that were chosen to start doing the statistical modeling of each one of them, the files containing the datasets of each one of the applications are taken separately. Rstudio is used again where a dataset of any application is provided in order to do the statistical modeling. First of all, the tool must recognize the dataset so that later it can be stored and managed in a more comfortable way and start with the modeling.

As a first step to start the modeling, the Cullen and Frey graph is obtained, which shows the square of the skewness and the kurtosis of the data distribution, thus helping to calculate which theoretical statistical distribution the attribute is more similar to, and be able to do the modeling accurately. Skewness is usually described as a measure of a dataset's symmetry or lack of symmetry. The normal distribution has a skewness of 0 [78]. The kurtosis parameter is a measure of the combined weight of the tails relative to the rest of the distribution [78]. This graph is made for each of the attributes of all the datasets of the different applications, this is because despite being the same attributes, the applications are totally different and have different attribute values (packet flows, arrival times, packet sizes, among others).

It should also be noted that not all attributes have the same modeling, since there are two types of data in the dataset, nominal and numerical. The nominal attributes are Source.IP, Source.Port, Destination.IP, Destination.Port and ProtocolName. It is important to mention that ProtocolName is an attribute that does not need statistical modeling since it is the unique label with the name of the application (either WhatsApp or Skype, among others). On the other hand, Source.IP, Source.Port, Destination.IP and Destination.Port are attributes that have to be modeled with the help of the Cullen and Frey graph. However, since these attributes are nominal, the distributions against which they are compared are different. The rest of the attributes of the dataset are numeric, are modeled in the same way and will be explained later. The main difference between numerical and nominal attributes is that the former are not discrete, that is, their statistical distribution have an infinite number of random possibilities; the latter are discrete, that is, their statistical distribution has values that are limited by finite numbers.

Continuing with the statistical modeling for a numerical attribute, when the process of obtaining the Cullen and Frey graph is done, it results in a graph like the one shown in figure 4:

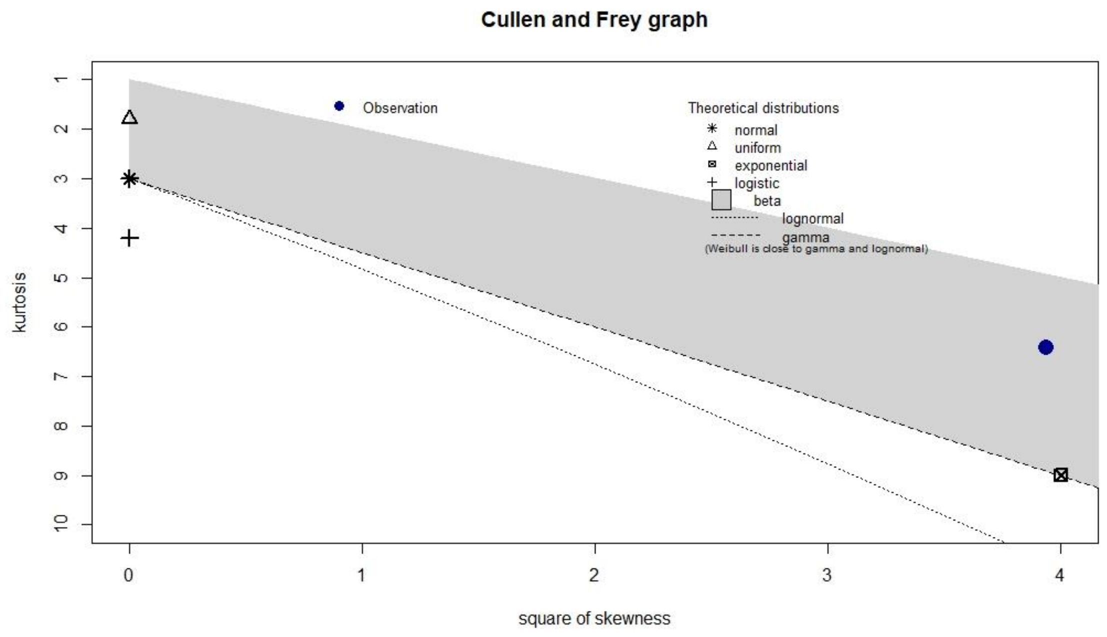

Figure 4. Cullen and Frey graph from Flow.Duration attribute of the YouTube Dataset.

Figure 4 shows the Cullen and Frey graph for the Flow.Duration attribute of the YouTube dataset. This attribute is compared against the normal, uniform, exponential, logistic, beta, lognormal, gamma and finally the Weibull distribution which is not plotted but is very similar to the lognormal and gamma distributions, using the square of skewness and the kurtosis values. As it can be noticed through the blue dot, this attribute has a behavior similar to a theoretical beta distribution. After the statistical distribution that fits the data is identified the parameters that describe the distribution of the attribute must be obtained.

First the normalization of its values must be done, since the beta distribution only has values in the interval [0,1]. After the normalization of the values of the attribute, the MLE[83] estimation is made, which is a statistical process to find the value of the parameters of the theoretical distribution that most closely resembles the distribution of the data being observed, so that the random values created are values close to those of the original attribute. It is important to mention that this estimation is made assuming statistical independence, that is, the values of one attribute do not depend on another. This statistical independence in real life is not true, since an attribute can depend on one or more attributes. It is taken this way, since the process to make the correlation of the values of the attributes can only be achieved in a mathematical way and until now there is no software tool that allows to manipulate the values of the attributes in a correlated way.

[83] *Maximum likelihood estimation*

To be able to observe the maximum likelihood estimation in the best way, this estimation is graphed as shown in figure 5.

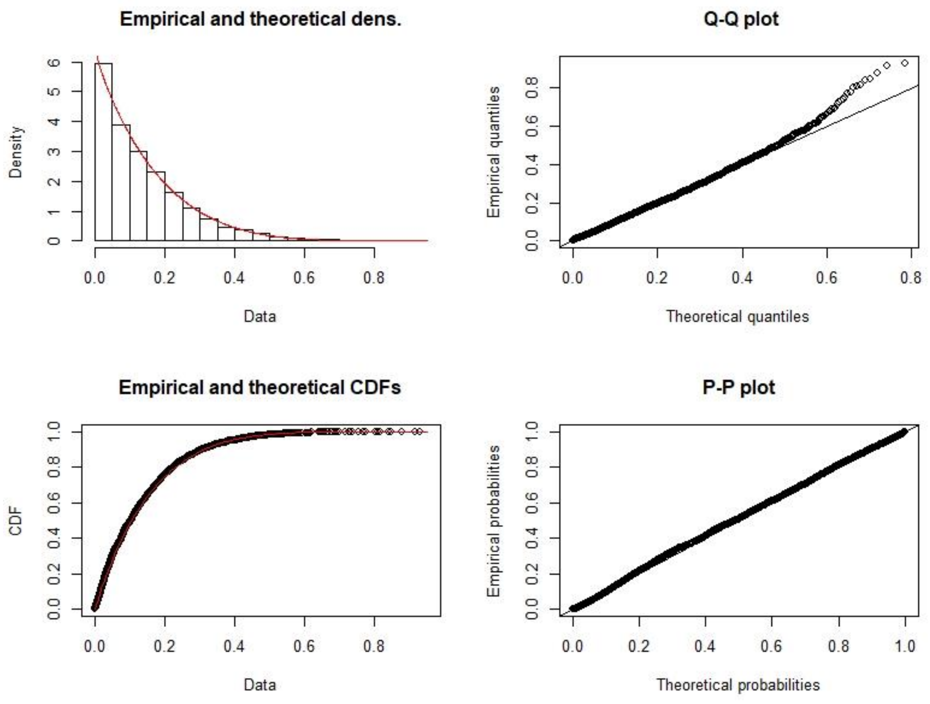

Figure 5. MLE plot from Flow.Duration attribute.

In the previous figure it can be seen that there are 4 different graphs. The first graph shows the empirical and theoretical density of the distribution in question, the red line being the theoretical and the histogram the empirical. Next to this graph is the Q-Q[84] plot that allows to observe how far or near are the values of empirical distribution compared with an ideal theoretical distribution [79]. Next, below the first graph is the CDF[85] graph of the empirical and theoretical distributions, which shows the probability that the values of the empirical distribution take the values of the theoretical distribution [80]. Finally there is the P-P[86] plot that allows to evaluate how similar the values between the empirical distribution and the theoretical distribution are [81].

Bearing this in mind, it can be said that the theoretical values and the values obtained are almost identical, which is why the modeling of the attribute is being done correctly.

Consequently, to create the random values of the attribute, the parameters of the maximum likelihood estimation are taken out, for this case, as the distribution of the attribute in question

[84] *quantile-quantile*
[85] *cumulative distribution function*
[86] *probability–probability*

is similar to a beta distribution the parameters that are needed to create the new values are alpha and beta, giving in this case in the following way:

$$\alpha: 0.2811929$$
$$\beta: 0.718752$$

Equation 1. Beta parameters for Flow.Duration Attribute.

After having obtained the parameters for the distribution, the new values for the attribute are randomly generated. Having these new values, the next step is to denormalize them. As mentioned before, the beta distribution only works with numbers in the interval of [0,1], for this reason the values had to be normalized. However, the real values of the attribute are not values that are in that interval, so in order to obtain the real values of the attribute the denormalization is done, thus obtaining the generation of synthetic values of the attribute. When this is concluded, the first synthetic generator for an attribute of a given application is made.
With this, it can be said that there is going to be as many generators as there are attributes in each dataset, excluding the attributes that are not going to be modeled.

It should be noted that not all of the attributes analyzed belonged to a single statistical distribution, many of them were in a kind of limbo in which it could be seen that the attributes could belong to two or three different theoretical distributions. For this case, the same process described above was done with each of the theoretical distributions to which the attribute resembles. Figure 6 shows Cullen and Frey's graph demonstrating this.

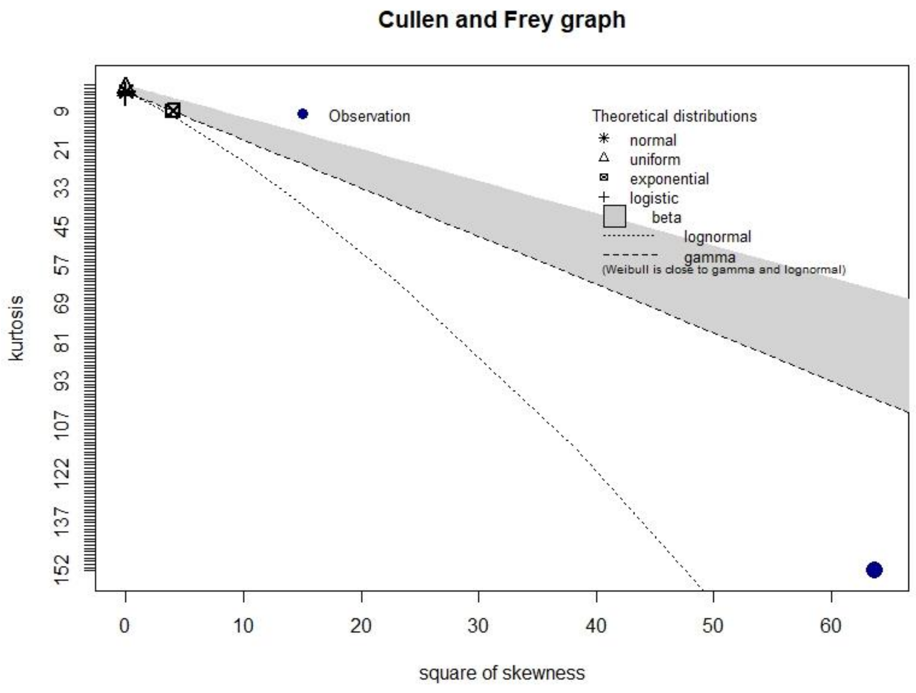

Figure 6. Cullen and Frey graph from Fwd.Packet.Length.Max attribute of the Google Dataset.

As it can be seen in the figure above, the Fwd.Packet.Length.Max attribute of the Google dataset is in the middle of the Gamma and lognormal distributions, and as said before, it should be also compared against the Weibull distribution because this one is similar to the lognormal and gamma distributions, so this attribute has to be compared with the three distributions mentioned.

To do this, the same steps of the process described above are followed, normalize the values of the attribute, and get the graphs for the distribution of the attribute compared with the graphs of the theoretical distribution using the method of maximum likelihood estimation. The result can be seen in figures 7, 8 and 9.

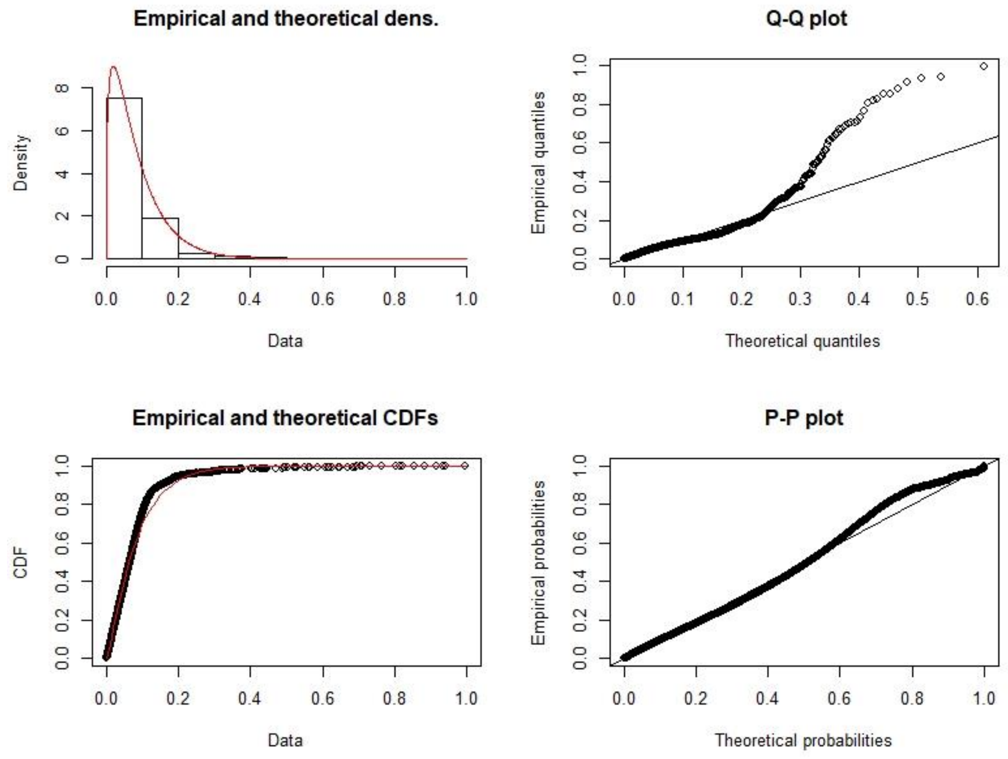

Figure 7. MLE plot from Fwd.Packet.Length.Max attribute for Gamma distribution.

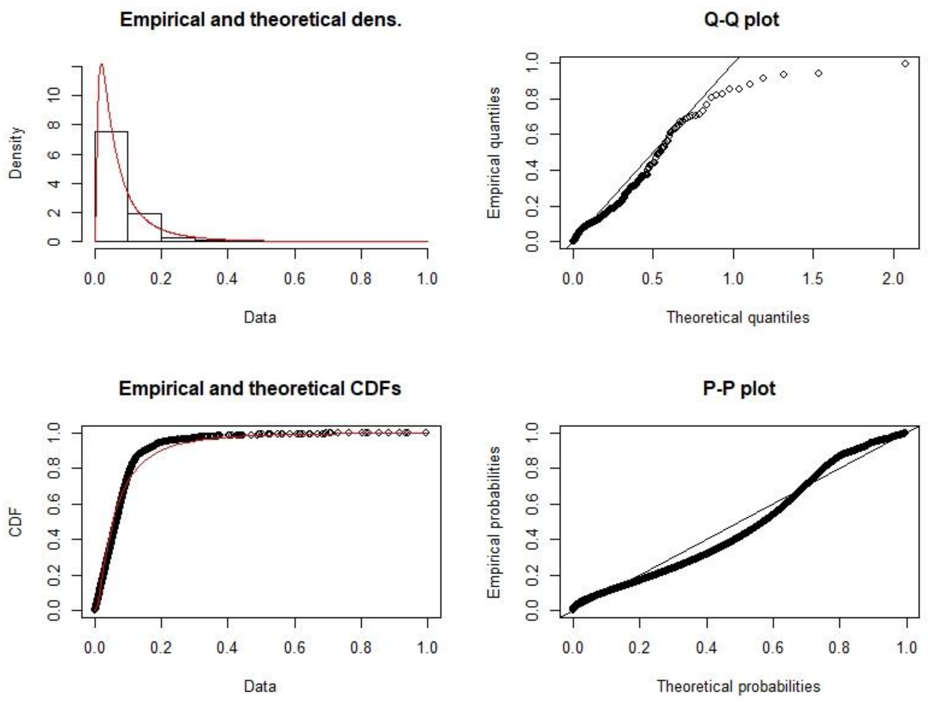

Figure 8. MLE plot from Fwd.Packet.Length.Max attribute for Lognormal distribution.

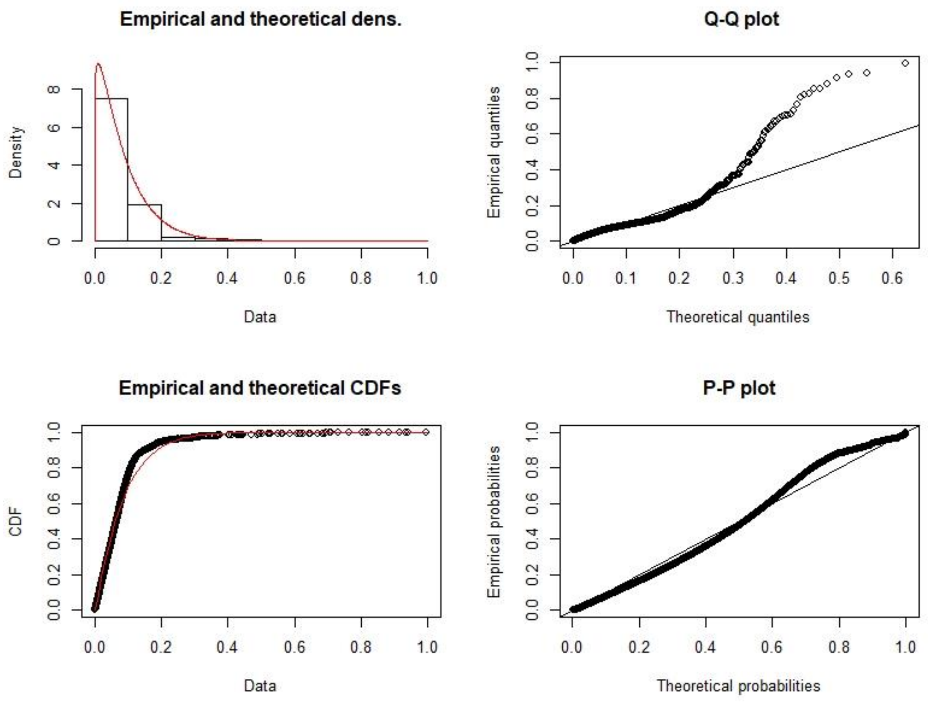

Figure 9. MLE plot from Fwd.Packet.Length.Max attribute for Weibull distribution.

As can be seen in the previous figures, at first glance it cannot be chosen the distribution to which the values of the attribute are most similar. Therefore, in order to get the proper distribution a Kolmogorov-Smirnov (K-S) test must be done. But before proceeding with this test, the parameters describing each of the distributions must be obtained.

For the gamma distribution alpha and beta parameters are needed, for the lognormal distribution Mi and sigma parameters, and finally for the Weibull distribution kappa and lambda parameters are needed. The results for this particular attribute can be seen below:

$$\alpha: 1.290911$$
$$\beta: 15.485522$$

Equation 2. Gamma parameters for Fwd.Packet.Length.Max Attribute.

$$\mu: -2.919426$$
$$\sigma: 1.017915$$

Equation 3. Lognormal parameters for Fwd.Packet.Length.Max Attribute.

$$\kappa: 1.09487015$$
$$\lambda: 0.08659681$$

Equation 4. Weibull parameters for Fwd.Packet.Length.Max Attribute.

After having made three different generators for this attribute, and obtaining the new synthetic values, the K-S test takes place. This test consists in comparing the values obtained with the values of the theoretical distribution, quantifying the distance between the values of the theoretical distribution to the values of the empirical distribution obtained from the data [80]. For this case, the result of the K-S test is as follows:

$$D = 0.0034971$$
$$\text{p-value} = 0.5737$$

Equation 5. Distance and p-value of the K-S test for Gamma Distribution, Fwd.Packet.Length.Max Attribute.

$$D = 0.0052334$$
$$\text{p-value} = 0.1293$$

Equation 6. Distance and p-value of the K-S test for Lognormal Distribution, Fwd.Packet.Length.Max Attribute.

$$D = 0.003731$$
$$\text{p-value} = 0.4895$$

Equation 7. Distance and p-value of the K-S test for Weibull Distribution, Fwd.Packet.Length.Max Attribute.

Where D is the quantified distance between the theoretical distribution and the empirical distribution and p-value is the value of the null hypothesis that the two distributions, empirical and theoretical, are equal, this null hypothesis is accepted as long as the value is greater than 0.05 [80]. In its order, the first distance corresponds to the gamma distribution, then the lognormal distribution and finally the Weibull distribution. The theoretical distribution that fits best the empirical data is the one with the quantified distance closer to zero. In this case, the closest distance to zero is the one of the gamma distribution, so the values that are going to be in the attribute generated will be those obtained by this distribution, also taking into account that the p-value has a value greater than 0.05, so the Gamma distribution fits perfectly for the generation. This part highlights the fact that this attribute is created with 3 different synthetic generators, but in the end the data that will be placed in the final dataset of the application, are the attributes generated with the distribution that the K-S test showed as the best fit to the data of the attribute.

Another example like the previous one can be seen in the Fwd.Packet.Length.Std attribute of the Spotify dataset. Figure 10 shows the Cullen and Frey graph, where it can be seen that this attribute only belongs to the lognormal and Weibull distributions. Figures 11 and 12 show the graph of the estimation of maximum likelihood, seeing that no easy recognition can be made to know which distribution best describes the attribute.

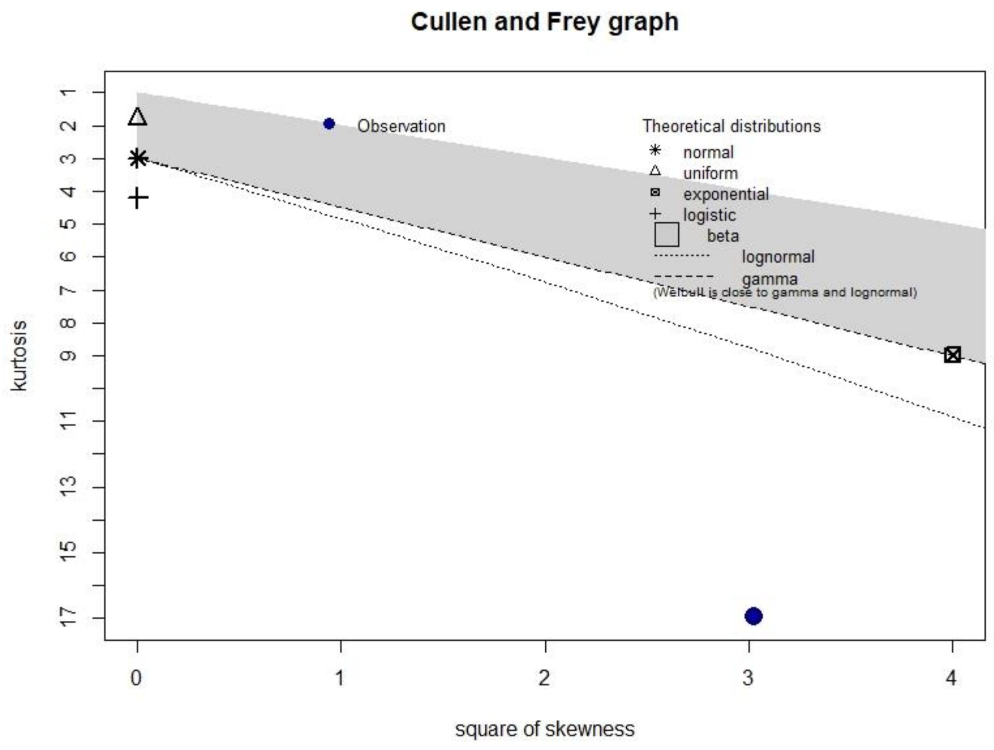

Figure 10. Cullen and Frey graph from Fwd.Packet.Length.Std attribute of the Spotify Dataset.

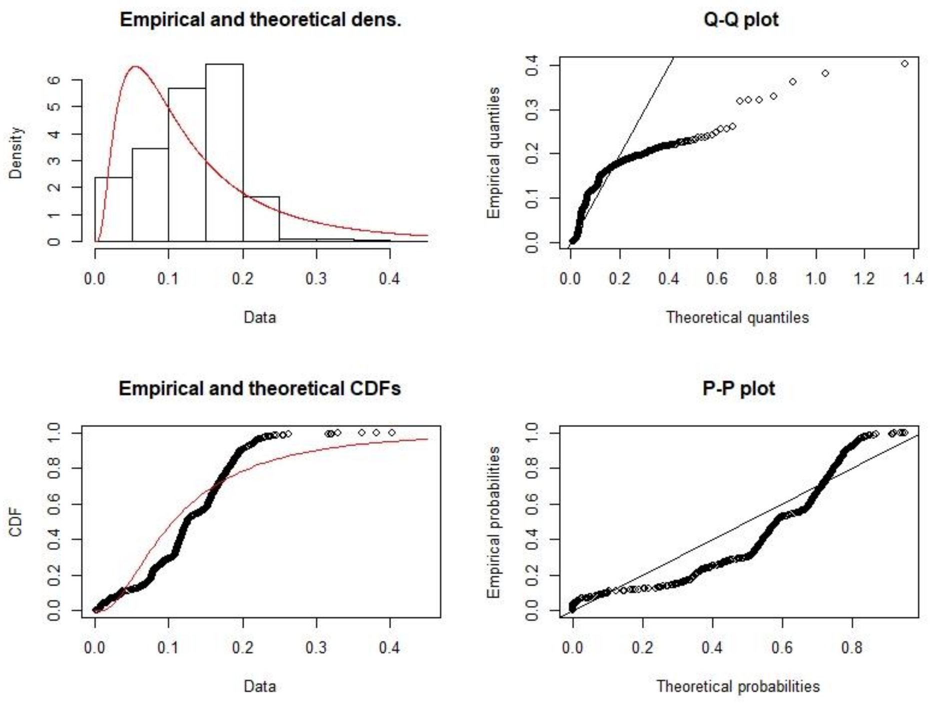

Figure 11. MLE plot from Fwd.Packet.Length.Std attribute for Lognormal distribution.

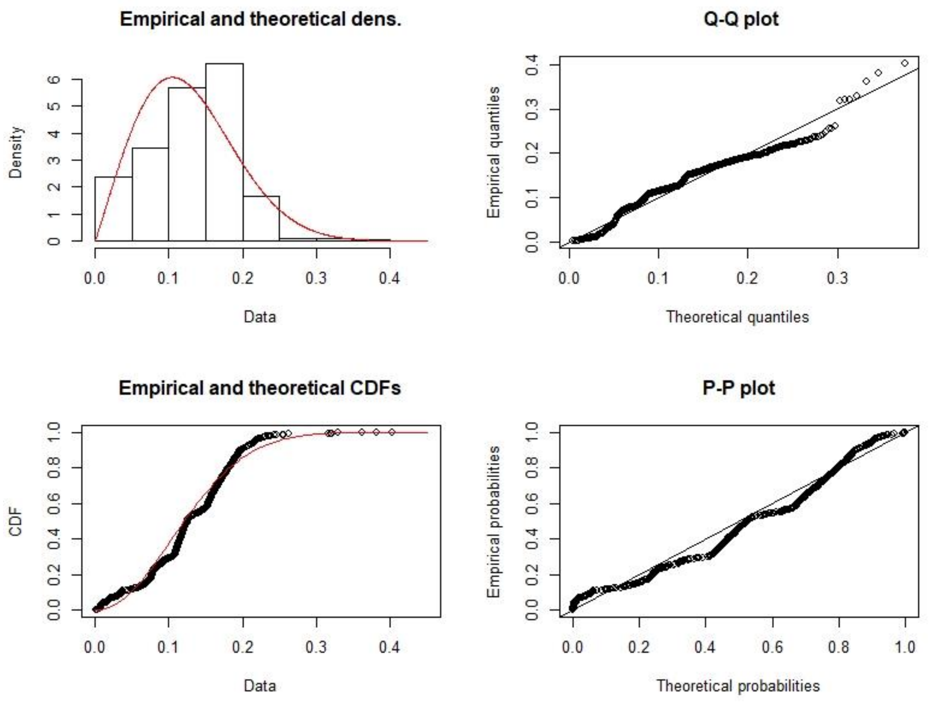

Figure 12. MLE plot from Fwd.Packet.Length.Std attribute for Weibull distribution.

For this attribute the same process is done, the values of the parameters are extracted for each distribution, lognormal and Weibull and then to be able to generate the synthetic data in a random way, then the K-S test is done and as a result the appropriate distribution is chosen for the values of the attribute. Below are the results of both, the parameters that describe the distributions and the K-S tests to see which distribution most closely resembles the attribute in question. The first two parameters correspond to the lognormal distribution, the following to the Weibull distribution and finally the distance of the lognormal distribution and the distance of the Weibull distribution.

$$\mu\text{: } -2.2476436$$
$$\sigma\text{: } 0.8088235$$

Equation 8. Lognormal parameters for Fwd.Packet.Length.Max Attribute.

$$\kappa\text{: } 2.0639209$$
$$\lambda\text{: } 0.1446731$$

Equation 9. Weibull parameters for Fwd.Packet.Length.Max Attribute.

$$D = 0.0024338$$
$$\text{p-value} = 0.9285$$

Equation 10. Distance and p-value of the K-S test for Lognormal Distribution for Fwd.Packet.Length.Max Attribute.

$$D = 0.0047187$$
$$\text{p-value} = 0.2155$$

Equation 11. Distance and p-value of the K-S test for Weibull Distribution for Fwd.Packet.Length.Max Attribute.

In the results given above, it is seen that the quantified distance from the distribution closest to zero is the one of the lognormal distribution, so that the values generated with that distribution will be the values set in the new attribute for the application dataset, also the p-value is greater than 0.05.

As mentioned above, not all values of all attributes were numerical, there are also attributes with nominal values, for this it is taken as an example the Source.IP attribute of the Skype application dataset. As this attribute is nominal, the distributions with which the comparison is made are different, in this case they are compared with the normal, negative binomial and Poisson distributions. With this attribute, a previous process had to be done in order to obtain the Cullen and Frey graph. As the values of this attribute have a different denotation, what should be done is to convert the IP addresses to decimal numbers and having these numbers the Cullen and Frey graph can be obtained as shown in Figure 13.

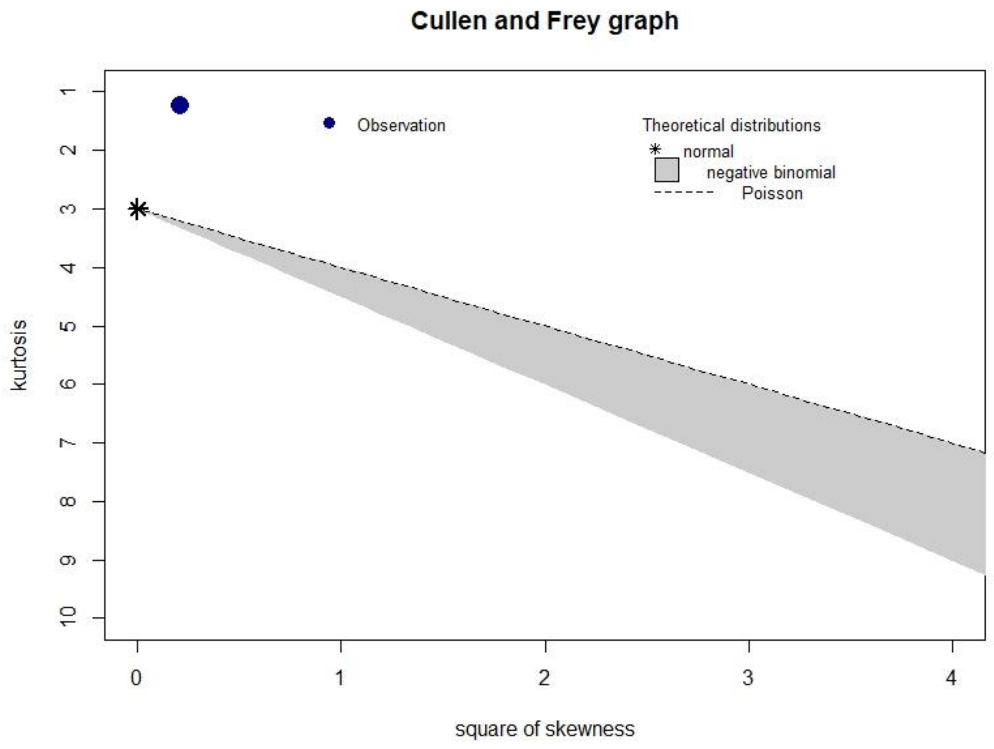

Figure 13. Cullen and Frey graph from Source.IP attribute of the Skype Dataset.

In this graph it can be see that the attribute can actually belong to any of the distributions, so the parameters for each distribution must be obtained. For this part it was not necessary to use the maximum likelihood estimation since, as mentioned before, the nominal values are over a finite range of possibilities. After having the parameters the same procedure is done again with the K-S test and in this case the distribution to which this attribute is more similar is the normal distribution. After knowing which is the adequate distribution for the new values of the attribute, in this case those values must be taken and the inverse process has to be done, that is, go from decimal numbers to IP addresses again, i.e., given the decimal number 2069496119, its equal as an IP address is 123.90.1.55. After this process, the new values are ready to be used in the new dataset.

The procedures described above were done for each of the attributes of each application, taking into account whether the attribute was numerical or nominal. As mentioned before, a generator was made for each one of the attributes of the different datasets of each one of the applications, leaving approximately 62 different generators for each application, not to mention the ones that had to be discarded after the K-S test was performed, leaving with a total of 464 different synthetic generators, 104 for YouTube, 94 for Google, 79 for Spotify, 103 for Skype and 88 for WhatsApp.

4.3 Creation and cleaning of datasets

In this section, it is explained the creation of the different datasets starting from the procedures described previously. It explains the creation process and how many datasets were created for this project. Additionally, the data cleaning that was done in the different datasets, the methodology that was used to do this cleaning and the result that is obtained after doing it is described.

To better understand the process, Figure 14 will show a small illustration of the steps that were followed from the model dataset until obtaining the clean datasets.

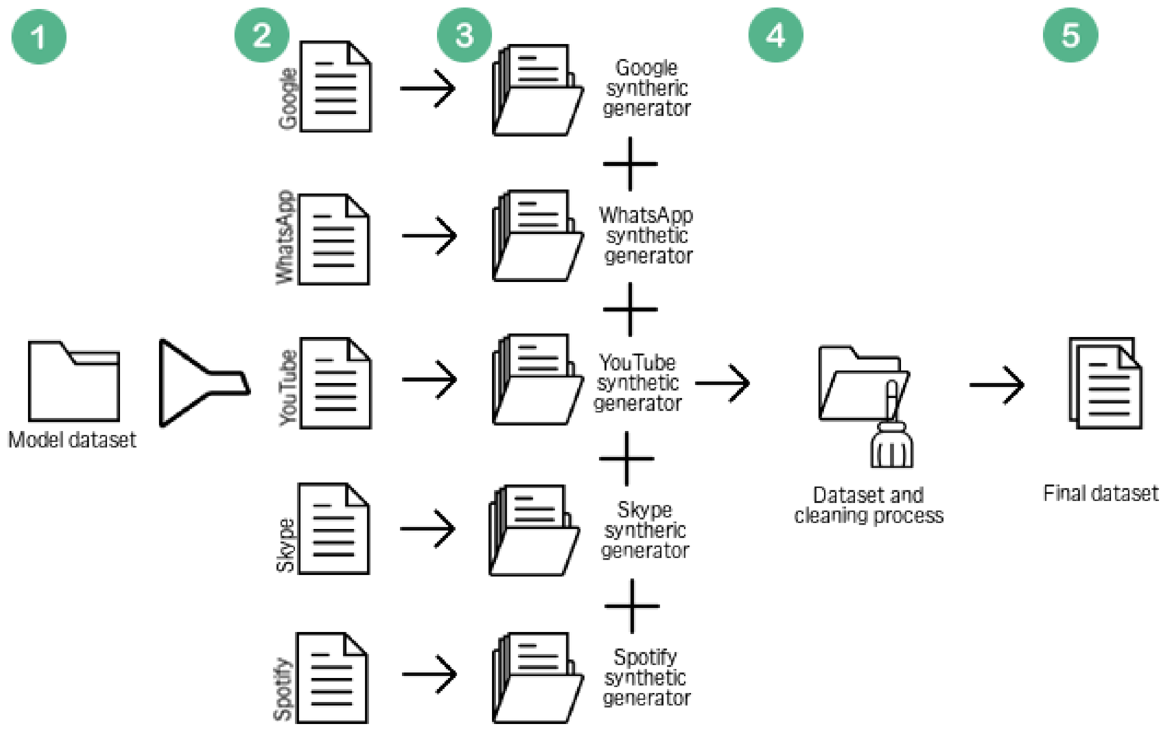
1
2
3
4
5
Google
Google synthetic generator
WhatsApp
WhatsApp synthetic generator
YouTube
YouTube synthetic generator
Skype
Skype synthetic generator
Spotify
Spotify synthetic generator
Model dataset
Dataset and cleaning process
Final dataset

Figure 14. Dataset processes.

As it can be appreciated in the previous figure, as first measure the model dataset is taken, and the filtering the OTT applications that were chosen to make the statistical modeling is done. Then each of the small datasets that belong to each of the applications is taken separately, and the statistical modeling of each of the attributes from each dataset is done. Then it can be seen the different synthetic generators that were created for each of the attributes of the datasets and how the union of these generators result in dataset with the new synthetic data by applications. Subsequently each of these datasets goes through a cleaning and grouping process to result in a dataset that contains the synthetic data based on the information of the OTT applications that were modeled at the beginning. As a final measure the dataset that is obtained is the one that will be validated with the different classification algorithms, however this part will be explained in the next chapter.

4.3.1 Creation of the datasets

As mentioned in section 4.2, using Rstudio the generators of the different attributes for each application are created. To understand this better, there are 5 separate datasets from each application: WhatsApp, Spotify, Skype, Google and YouTube, remembering that these datasets were filtered from the dataset that was taken as a model which is explained in section 4.1. Each dataset is loaded in a separate script where the statistical modeling of each of the attributes was started and then the creation of the new values was performed. After creating these generators for each attribute, with the help of Rstudio these new values from the attributes were grouped, each attribute being a different column. In this way, new datasets were created for each application. Having all the new datasets of the applications separately, in a new R script all the datasets were loaded and the process of grouping all of them into one big dataset was done, obtaining the one with which this project is going to work.

For this part, several datasets with different numbers of instances for each attribute of the applications were created. This number of instances were created when the attribute was created, e.g, when it was known to which distribution the attribute resembled more and had the parameters of each distribution, then the random values with the number of instances were created and the number of instances can be as much as it is needed.

First a dataset was created in which each application had 50,000 instances for each attribute, leaving a dataset of 250,000 instances in total. Then datasets of 30,000, 20,000, 10,000 and 5,000 instances were created, leaving datasets of 150,000, 100,000, 50,0000 and 25,000 instances respectively.

It is also worth mentioning that for each application two different datasets were created with the same numbers of instances, that is, two datasets of 50,000 instances be attributes were created, two datasets with 30,000 instances by attributes, and so on. It was done in this way because at the end there would be a training dataset and a test dataset to then proceed to make the validation of the datasets that will be explained in chapter 5.

4.3.2 Data cleaning process

In this section the cleaning process of the datasets is done according to [76]. In this paper the authors talk about making an effective cleaning without damaging the quality of the data of the dataset. To be able to do this, first a diagnosis of the quality of the collected data has to be made. This diagnosis verifies the quality of the data and is based on the CRISP-DM methodology, which was named in section 4.1 of this chapter. This data cleaning belongs to the data preparation phase of this methodology, which considers the following problems: outliers, atypical values that deviate from the normal values of the variable under observation; noise, defined as irrelevant data within a data set; inconsistency, contradictory instances, i.e., sometimes there are two instances that have the same name but differ in their values, or on the contrary, instances with different names whose values are the same; incompleteness, refers to lost data that may exist in datasets; finally timeliness, refers to the time in which the data were collected, e.g if the data that is collected really belongs to the timeline that is wanted, or if on the contrary this data is out of the required timeline.

As stated at the beginning of section 4.2, many of the attributes were discarded because all their instances were zero, these attributes in the data quality diagnosis fit into the noise category as they are considered irrelevant to the dataset. However, this does not mean that these communication attributes are irrelevant because they are not. Remembering, many of these attributes are: Fwd.PSH.Flags, Bwd.PSH.Flags, Fwd.URG.Flags, Bwd.URG.Flags; The flags PSH and URG are part of the TCP stack and serve to indicate that this is the last byte sent from the connection to go once to the application and to report urgent data that must be prioritized respectively. But as mentioned before, these flags are normally in zeros and it would not be necessary to do a statistical modeling since their approximation will always be a vector of zeros. Other attributes that make part of this are: FIN.Flag.Count, SYN.Flag.Count, RST.Flag.Count, PSH.Flag.Count, ACK.Flag.Count, URG.Flag.Count, CWE.Flag.Count, ECE.Flag.Count; this is the count of the flags that there are when sending a packet, they are also important in the real connection since they indicate how many times the flag appears, however and as it was said before, these attributes are also zeros and this being a creation of simulated data it is not necessary to take into account these attributes.

Fwd.Avg.Bytes.Bulk, Fwd.Avg.Packets.Bulk, Fwd.Avg.Bulk.Rate, Bwd.Avg.Bytes.Bulk, Bwd.Avg.Packets.Bulk and Bwd.Avg.Bulk.Rate are other of the attributes in which it was found that their majority are zeros, these attributes also contribute information in the real connection, however for this project they were not taken into account. Flow.ID is an identifier put by the CICFlowMeter application, which has nothing to do with the data flow. Protocol is an attribute that only says if the connection was made through TCP or UDP protocol, in this case all connections are made through TCP protocol. Label is an identifier that says if the IP flow communication is benign or malign, something that is important in the packets, but being these applications known, label will always be benign type.

The attribute Fwd.Header.Length.1 has the same values and instances as the attribute Fwd.Header.Length, so this falls into the category of inconsistency with the data and the attribute is discarded. The last attribute discarded before starting the generation of synthetic data was L7Protocol, this decision was made because depending on the application to which the flow belonged, this attribute indicated a unique identifier number for the application. This was going to generate problems when classifying the traffic, since the algorithms would not be tested properly, being this attribute the answer that the algorithm needs to make an excellent classification.

After having removed the attributes that could interfere with the quality of the dataset data and having the new dataset with the generated synthetic data, a procedure to evaluate the generated data and analyze if it had any problem is made. In this case, it was found that there were attributes that had lost instances in the new dataset. These instances fall into the category of incompleteness. All instances of the attributes were reviewed and rows that had incomplete instances were discarded. It should be noted that this did not occur in all datasets and not all attributes had problems with incomplete instances.

Next, it was continued with the revision of the values that had been generated previously and it was found that in some cases there were problems with the Source.Port and Destination.Port attributes of some of the datasets of the applications. These problems are catalogued as outliers since the values that were generated were atypical values with respect to those that were expected. To give a little context, the ports that exist on the Internet are finite ports that go from port 0 to port 65.535. In many cases when doing the statistical modeling along with the process explained in section 4.2, this attribute seemed to have a distribution similar to the normal distribution, so it was proceeded to create new instances with this distribution. The problem was that this distribution delivered numbers much larger than 65,535 in almost all its instances and for this reason more than half of the data had to be discarded. However, the binomial negative distribution delivered the values needed within the established parameters, so these two attributes had to be evaluated and changed by the distribution to which they belonged.

The problem mentioned above occurs because the tests depend on the data being randomly generated. That is, in section 4.2 it was explained that the generated data are taken to make a K-S test, which indicates that it takes the generated data and compares them with the theoretical distribution. However, the values of the quantified distance delivered by the test will vary depending on the random values generated, so in some cases the K-S test may present a smaller distance in a certain distribution, but it is another distribution that actually makes a better modeling of the data. Although the binomial negative distribution showed better results for these two attributes, there were some instances that were out of the established range, therefore the rows in which the Source.Port or Destination.Port had a number greater than 65,535 were discarded.

This concludes the cleaning of the dataset, it should be noted that this cleaning was done before forming the complete dataset with all applications, meaning, the cleaning was done by taking the datasets of the different applications separately.

Summary

This chapter gave a brief description of the dataset model that was used to make the statistical modeling of the different attributes that are present in a communication over IP. There was also a brief introduction of the tools needed to obtain an adequate format of the dataset taken as a model for its later manipulation.

Next, a detailed explanation of how the statistical modeling of each of the attributes was done is presented, explaining that before starting, the OTT applications of the original dataset were filtered in different datasets separated by application. Then the datasets are taken separately and the statistical modeling of each of the attributes is started, also separately. This statistical modeling results in a statistical distribution to which the values of the attribute instances are more similar to. Knowing which distribution each attribute belongs to, the synthetic data is generated randomly and once all the data has been modeled, all the data is grouped being each attribute and its instances a different column in each dataset. By doing this, it culminates in the creation of the different datasets for each of the OTT applications, YouTube, Spotify, Skype, Google and WhatsApp.

It should be noted that in the construction of the new dataset it could be created different instances for each of the attributes, taking into account that in the dataset there must be the same number of instances in all of the attributes. Datasets were created with 50.000, 30.000, 20.000, 10.000, and 5.000 instances per attribute, leaving datasets of 250.000, 150.000, 100.000, 50.000 and 25.000 instances when grouped all datasets of the different OTT applications. Two different datasets were created by applications and number of instances, a training dataset and a test dataset.

Finally, the data preparation phase of the CRISP-DM methodology was explained, where the quality of the data was analyzed, taking into account the categories that compose this phase, noise, incompleteness, outliers, inconsistency and timeliness. When attributes and values that fit in those categories were discarded, the datasets are now clean and ready for the next phase of the project.

CHAPTER 5

DATASET VALIDATION AND MACHINE LEARNING TESTS

This chapter talks about the validation of the different datasets that were obtained in the previous chapter. The validation of the datasets is done through machine learning classification algorithms, with the help of the Weka tool. This validation is made only for two of the datasets, these are the one with 50.000 and 25.000 instances, bearing in mind that for each number of instances there were two different datasets, one for training and one for tests.

In the previous chapter it was talked about how the different datasets were created for different OTT applications, taking into account the number of instances of each attribute. The decision was to create two different datasets with the same number of instances by attributes, so there were two datasets, one for training and one for testing. Machine learning validation was done taking these two datasets.

Finally, it was obtained the results of all of the algorithms that were used for the validation of the datasets, making also a T-test and arriving at the conclusion of which algorithm was the one that made the best classification.

The following subsection will explain the algorithms used and the classification metrics of these algorithms.

5.1 Metrics and algorithms

This section is going to talk about performance metrics and machine learning algorithms. These metrics are composed by: the F-Measure, precision, recall, confusion matrix and kappa statistic. Additionally, the machine learning algorithms used for the validation and testing of the datasets will be discussed, taking into account that these algorithms fall into the category of supervised algorithms. The chosen ones were: J48, RandomForest, AdaboostM1, Bagging, IBK and NaiveBayes, The decision to take these algorithms was based on the articles [82] and [83] which shows the performances of each of them.

5.1.1 Performance metrics

In this subsection, a brief description of the most important classification metrics in machine learning can be found.

- **Confusion Matrix:** When putting a machine learning algorithm to work it can be noticed that the result is a set of tests that is often shown as a two-dimensional matrix with a row and a column for each class that has the dataset that is being validated. Each element of the matrix shows the number of test examples for which the real class is the row and the expected class is the column, showing 4 different types of results, true positive, false positive, true negative and false negative. The good results of this matrix correspond to large numbers in their main diagonal and to small numbers, ideally zero, outside the diagonal, thus it can be noticed if the classification algorithm serves or not [84].

  The four types of results that can be found in the confusion matrix are shown below:

  True positive: Are those instances that the classifier assigns to a certain class and that really belong to that class.

  False positive: Are those instances that belong to a certain class but that the classifier assigns to another class.

  False negative: Are those instances that were assigned to a class but do not belong to that class.

  True negative: Are those instances that were not assigned to a class and did not really belong to that class.

- **Recall:** It is defined as the number of instances assigned to a class of the total number of instances that belong to that class. To understand this better, Recall is defined as [15]:

$$\boldsymbol{Recall = \frac{TP}{TP + FP}}$$

Equation 12. Recall definition.

  Where TP is true positive and FP is false positive.

- **Precision:** It is defined as the number of instances assigned to a certain class of the total number of instances that exist in all classes of the dataset. Precision is defined as [15]:

$$\boldsymbol{Precision = \frac{TP}{TP + FN}}$$

Equation 13. Precision definition.

Where TP is true positive and FN is false negative.

- **F-Measure:** It is the harmonic measure obtained from precision and recall. This helps to measure the reliability of the test performed with a variation of results between 0 and 1. The closer it is to 1 the more reliable the classifier is. F-Measure is defined as [84]:

$$\boldsymbol{F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall}}$$

Equation 14. F-Measure definition.

- **Kappa Statistics:** It is a measure that takes into account the randomness in the performance of a classification algorithm, so that it can be determined if this performance is due to chance or not. It is also defined in a range of 0 to 1, where 0 refers to a classifier totally affected by chance and 1 represents an ideal classifier [84].

5.1.2. Definition of Algorithms

- **J48:** Is the implementation of a decision tree classifier called C4.5, which is a classification algorithm based on a binary decision tree [24]. J48 has a tree structure with nodes representing characteristics that show possible values that connect these characteristics. It uses the divide and conquer approach to build a tree with division criteria of the gain ratio based on entropy, making use of the key concepts of information theory to know which attribute to select, allowing the user to easily understand the decision tree created [83].

- **Random Forest:** It is based on decision tree classification algorithms to generate a large number of decision trees in which each tree is built taking as main source different samples of the original data using a random selection of characteristics in the tree induction process, so that in the end this algorithm results in a classifier in the form of many individual decision trees [83].

- **AdabosstM1:** It is an important classifier based on sets. The idea of AdaBoost is to be able to combine it with other classifiers, such as decision trees, to improve its accuracy and performance. AdaBoost takes as its basis a classifier built from training data, assigns equal weights to all samples of training data, depending on the performance of the classifier, and then modifies the weight of each sample from the training data. Another classifier is then established to focus on examples of training data that were incorrectly obtained from the base classifier [83].

- **Bagging:** An assembly method that creates individuals for fit by training each classifier in a random redistribution of the training set. The training set for each classifier is generated by randomly extracting N examples, N being the size of the original training set. Many of these N original elements can be repeated in the resulting training set, while others may not appear in it. Each individual classifier in the set is generated with a different random sampling of the training set. This classifier is also combined with different classification algorithms [85].

- **IBK:** The IBK or K-Nearest Neighbor classification classifies instances according to their similarity being one of the most popular algorithms for pattern recognition and a lazy type of learning. Its operation is based on the fact that an object is classified by the majority of its neighbors, with K always being a positive integer. It will weight the contribution of each one of the neighbors close to the sample according to the distance, giving greater value to the neighbors closest to the sample. The closest neighbors of the attributes may be totally irrelevant which makes the algorithm not perform well under these conditions, so the less relevant attributes of the data sets must be removed [86].

- **NaiveBayes:** It is an algorithm based on the Bayesian theorem and an assumption that all attributes are equally important and independent of each other for a given class. It estimates the Gaussian distribution of attributes for each class based on a pre-labeled training set. It uses the previous probability or result to then determine the subsequent probability of a new instance, approximating each attribute by means of a Gaussian distribution [24].

5.2. Algorithm configuration and different tests

This section talks about the configuration that was taken into account for the classification algorithms that were used for this part of the undergraduate thesis and also gives a very brief introduction to the different tests that were performed with the different datasets and classification algorithms.

5.2.1 Algorithms configuration

As mentioned above, the tool used for the different machine learning algorithms was Weka. This tool is a suite containing a collection of automatic learning algorithms for data mining tasks developed by the University of Waikato, New Zealand. It contains tools for data preprocessing, classification, regression, grouping, feature selection, association rules and visualization. In addition, it also allows the development of new automatic learning schemes.

When the validation of the datasets with this tool was about to start, first, it was chosen one by one the algorithms that were going to be used. As a first algorithm, the decision trees were chosen, one of which was J48, leaving the configuration that gives Weka by default, followed by RandomForest in where was changed the number of iterations that the algorithms does, this was done because by doing so many iterations the consumption of hardware resources was really high and the classifier failed to finish the modeling, so the number of iterations done for the datasets of 50,000 instances were 5 and for the datasets of 25,000 instances were 10.

Then two algorithms that are ensemble methods were taken, which were AdaboostM1 and Bagging, both leaving them with the default values and using them with the decision tree J48. Then it was taken the KNN (K Nearest Neighbor) algorithm that is represented within Weka as IBK, giving as a nearest neighbor number 25 using a cross validation approach to determine such number and without changing any other default values in Weka. Finally it was taken the Naive Bayes algorithm, also leaving the values that Weka has by default.

5.2.2 Different tests performed

In this section it is presented the 3 different tests that were performed with the datasets of 50.000 and 25.000 instances.

- **CrossValidation:** It is a test in which a certain number of partitions or folds are chosen for the dataset, for example, the number 4 is chosen. This means that the dataset is split into approximately 4 equal parts, and what it does is take a quarter of the dataset to do the training and the remaining three quarters are taken to do the tests. This procedure is repeated four times, so each part of the dataset is used for training and tests. However, for this test a CrossValidation with 10 fold is used since numerous tests with different datasets and different classification algorithms have shown a better performance and there are some theoretical evidences that demonstrate this [84].

- **Percentage Split:** In this kind of test, a percentage of instances from the dataset that will be used for training the algorithm is chosen and the rest is used for testing. In this case the configuration that was taken was the one that comes by default in Weka with a percentage split of 66% [84].

- **T-test:** The T-test is used to make the comparison of each algorithm with respect to another. This means, an algorithm is taken for a certain dataset and with that same dataset another algorithm is tested. This leads to a comparison between the two algorithms, which in the T-test gives a result where the percentages of each of the tests performed with the different algorithms appear. These tests help to decide if one algorithm is much better than another, or if on the contrary these two algorithms are just as good to validate the dataset. In this case the T-test was done comparing the algorithms J48, RandomForest, AdaboostM1, Bagging, IBK and NaiveBayes, being done with cross validation of 10 fold [84].

5.3. Classification tests and results

Two of the datasets named in the previous section were taken into account for this part. As mentioned before, 10 different datasets were obtained, each of them containing different number of instances per attribute, 250,000, 150,000, 100,000, 50,000 and 25,000 instances, having from each number of instances two datasets, one for training and the other one for testing.

The first thing that was done was to take the training dataset of 250,000 instances and the first algorithm that was evaluated was the J48 using the first test with Cross validation of 10 fold. Then, with the same Cross validation configuration explained in section 5.2, the test dataset with the same instances is taken and the algorithm is tested. For this dataset the only algorithms from which results were obtained for cross validation were J48, NaiveBayes and the IBK algorithms, this last one taking more than 7 hours trying to make the classification. The other algorithms such as AdaboostM1, Bagging and RandomForest could not be tested because the machine on which the classification algorithms were running did not have enough hardware resources to test these algorithms as these hardware resources were insufficient to launch the classification algorithms.

Taking this into account, then it was decided to start creating datasets with fewer instances, it was tested with the datasets of 150,000 instances and 100,000 instances but the hardware resources were still insufficient to complete the training and testing of the 3 missing algorithms.

Finally the datasets of 50.000 and 25.000 instances were taken, obtaining from these the results for the different algorithms that were required to test, J48, RandomForest, AdaboosM1, Bagging, IBK and NaiveBayes. It should be noted that in all cases it was also wanted to test the algorithms LibSVM and SMO, without being able to get results with any of the different dataset due to the same problem of insufficient resources of the machine.

The results of the different tests are shown below, showing first the tests obtained from the training phases and then the results obtained from the test datasets. In first place, the CrossValidation tests are presented, then the Percentage Split tests and finally the T-test with cross validation to make a comparison of the implemented algorithms. For space reasons only the tests performed with the dataset that has 50.000 instances will be shown, however, in appendix 1 can be appreciated the tests that were done with the dataset of 25,000 instances. Also, it is important to mention that the tests were done with the datasets that went through the data cleaning process described in section 4.3.2 of the previous chapter, as well as with the datasets without doing any kind of cleaning.

The results will be shown as follows: first the confusion matrix of each of the algorithms will be shown, followed by the performance metrics discussed in subsection 5.1.1 which are precision, recall, F-Measure and kappa statistic; and finally there is a brief explanation and interpretation of the results obtained with the different classification algorithms and the different tests done.

Before continuing with the results, it should be clarified that when talking about a dataset of 50,000 instances that have gone through a cleaning process, this number is only an approximation of the actual instances that exist in the dataset. The actual number of instances for the training dataset is 49,791 and the actual number of instances per application is 9,966 for Google, 9,964 for Skype, 10,000 for Spotify, 9,897 for WhatsApp and 9,964 for YouTube. The actual number of instances for the test dataset is 49,804 and the actual number of instances per application is 9,978 for Google, 9,971 for Skype, 10,000 for Spotify, 9,892 for WhatsApp and 9,963 for YouTube.

5.3.1 J48 with CrossValidation

- **Training dataset**

A	B	C	D	E	CLASSIFIED AS
8060	839	0	83	984	**A = Google**
828	8818	2	13	303	**B = Skype**
0	1	9953	46	0	**C = Spotify**
61	8	72	9729	27	**D = WhatsApp**

971	304	1	43	8645	**E = YouTube**

Table 4. Confusion matrix of J48 algorithm with CrossValidation, training dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.813	0.809	0.811
Skype	0.884	0.885	0.885
Spotify	0.993	0.995	0.994
WhatsApp	0.981	0.983	0.982
YouTube	0.868	0.868	0.868
Weighted Avg	0.908	0.908	0.908

Table 5. Precision, Recall and F-Measures of J48 algorithm with CrossValidation, training dataset.

Kappa Statistic: 0.8849

- **Test dataset**

A	B	C	D	E	CLASSIFIED AS
8027	817	0	62	1072	**A = Google**
822	8829	2	11	307	**B = Skype**
0	0	9918	82	0	**C = Spotify**
57	15	80	9707	33	**D = WhatsApp**
988	300	2	46	8627	**E = YouTube**

Table 6.Confusion matrix of J48 algorithm with CrossValidation, test dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.811	0.804	0.808
Skype	0.886	0.885	0.886
Spotify	0.992	0.992	0.992
WhatsApp	0.980	0.981	0.981
YouTube	0.859	0.866	0.863
Weighted Avg	0.906	0.906	0.906

Table 7. Precision, Recall and F-Measures of J48 algorithm with CrossValidation, test dataset.

Kappa Statistic: 0.8821

In this test it can be noticed that the diagonal of the confusion matrix has the largest number of instances, and that its surroundings have smaller numbers to a large extent, this means that the classification algorithm is good enough for these datasets. It can also be noticed that the kappa statistic does not have a very big difference when comparing the performance it had with the training and test dataset showing that the learning done with the training dataset is good enough to do the validation of other similar dataset; Additionally that the result of the Kappa statistic is close to 1 which means that the classifier is not due to chance. The F-Measure also shows measurements very close to 1 so it can be said that the classifier is reliable.

5.3.2 RandomForest with CrossValidation

- **Training dataset**

A	B	C	D	E	CLASSIFIED AS
5951	1613	85	633	1684	**A = Google**
1252	7396	108	649	559	**B = Skype**
14	19	9824	139	4	**C = Spotify**
120	149	816	8716	96	**D = WhatsApp**
1269	498	39	252	7906	**E = YouTube**

Table 8. Confusion matrix of RandomForest algorithm with CrossValidation, training dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.691	0.597	0.641
Skype	0.764	0.742	0.753
Spotify	0.904	0.982	0.941
WhatsApp	0.839	0.881	0.859
YouTube	0.771	0.793	0.782
Weighted Avg	0.794	0.799	0.795

Table 9. Precision, Recall and F-Measures of RandomForest algorithm with CrossValidation, training dataset.

Kappa Statistic: 0.749

- **Test dataset**

A	B	C	D	E	CLASSIFIED AS
5569	2243	314	304	1548	**A = Google**
2503	5659	451	399	959	**B = Skype**
8	373	8901	702	16	**C = Spotify**
1916	950	2391	4506	129	**D = WhatsApp**
2858	990	67	108	5940	**E = YouTube**

Table 10. Confusion matrix of RandomForest algorithm with CrossValidation, test dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.433	0.558	0.488
Skype	0.554	0.568	0.561

Spotify	0.734	0.890	0.805
WhatsApp	0.749	0.456	0.566
YouTube	0.691	0.596	0.640
Weighted Avg	0.632	0.614	0.612

Table 11. Precision, Recall and F-Measures of RandomForest algorithm with CrossValidation, test dataset.

Kappa Statistic: 0.5173

In this test the results vary greatly. In the first confusion matrix it can be seen that although the biggest values are in the diagonal, the values that are in their surroundings are not much smaller than those of the diagonal, existing a reduction of the values of the diagonal in the confusion matrix of the test dataset. To this can be added the fact that the Kappa statistics obtained in the training dataset and in the test dataset has a very large difference, so it can be said that the algorithm did not make a good learning, showing that in both cases the results may be affected by chance. F-Measures also lower their values, thus demonstrating that the classifier is unreliable.

5.3.3 AdaboostM1 (J48) with CrossValidation

- **Training dataset**

A	B	C	D	E	CLASSIFIED AS
9108	562	0	57	239	**A = Google**
2590	7333	0	12	29	**B = Skype**
0	0	9995	5	0	**C = Spotify**
20	0	71	9804	2	**D = WhatsApp**
1504	274	1	17	8168	**E = YouTube**

Table 12. Confusion matrix of AdaboostM1 algorithm with CrossValidation, training dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.689	0.91	0.786
Skype	0.898	0.736	0.809
Spotify	0.993	1.000	0.996
WhatsApp	0.991	0.991	0.991
YouTube	0.968	0.820	0.888
Weighted Avg	0.908	0.892	0.894

Table 13. Precision, Recall and F-Measures of AdabosstM1 algorithm with CrossValidation, training dataset.

Kappa Statistic: 0.8649

- **Test dataset**

A	B	C	D	E	CLASSIFIED AS
5801	2672	0	60	1445	**A = Google**
74	9818	1	14	64	**B = Skype**
0	0	9994	6	0	**C = Spotify**
13	1	78	9788	12	**D = WhatsApp**
131	555	0	21	9256	**E = YouTube**

Table 14. Confusion matrix of AdaboostM1algorithm with CrossValidation, test dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.964	0.581	0.725
Skype	0.753	0.985	0.853
Spotify	0.992	0.999	0.996
WhatsApp	0.990	0.989	0.990
YouTube	0.859	0.929	0.893
Weighted Avg	0.911	0.897	0.891

Table 15. Precision, Recall and F-Measures of AdaboostM1 algorithm with CrossValidation, test dataset.

Kappa Statistic: 0.8708

In this test it can be seen that the kappa statistic is a little better in the test dataset than in the training dataset, however the difference is not exaggeratedly large. It can also be seen that the F-Measures are better in the test dataset. In the two confusion matrices, the largest numbers are in the middle of the matrix, which means that it is a good classifier.

5.3.4 Bagging (J48) with CrossValidation

- **Training Dataset**

A	B	C	D	E	CLASSIFIED AS
7759	1482	0	78	647	**A = Google**
237	9621	2	11	93	**B = Skype**
0	0	9988	12	0	**C = Spotify**
20	0	78	9787	12	**D = WhatsApp**
412	478	1	36	9037	**E = YouTube**

Table 16. Confusion matrix of Bagging algorithm with CrossValidation, training dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.921	0.779	0.844

Skype	0.831	0. 966	0.893
Spotify	0.992	0. 999	0.995
WhatsApp	0.986	0. 989	0.988
YouTube	0.923	0. 907	0.915
Weighted Avg	0.931	0.928	0.927

Table 17. Precision, Recall and F-Measures of Bagging algorithm with CrossValidation, training dataset.

Kappa Statistic: 0.9096

- **Test Dataset**

A	B	C	D	E	CLASSIFIED AS
7858	1374	0	67	679	**A = Google**
249	9596	0	15	111	**B = Skype**
0	0	9976	24	0	**C = Spotify**
29	2	84	9753	24	**D = WhatsApp**
425	484	1	43	9010	**E = YouTube**

Table 18. Confusion matrix of Bagging algorithm with CrossValidation, test dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.918	0.788	0.848
Skype	0.838	0.962	0.896
Spotify	0.992	0.998	0.995
WhatsApp	0.985	0.986	0.985
YouTube	0.917	0.904	0.911
Weighted Avg	0.930	0.927	0.927

Table 19. Precision, Recall and F-Measures of Bagging algorithm with CrossValidation, test dataset.

Kappa Statistic: 0.9094

In this test it can be seen that this is one of the best classifiers that were tested. Kappa statistics are considerably close to 1, just like F-Measures. However, this may not be appreciated very well in confusion matrices as there are quite a few instances outside the diagonal that are not very close to zero, although the diagonal of the matrices have the largest number of instances.

5.3.5 IBK (25 neighbors) with CrossValidation

- **Training Dataset**

A	B	C	D	E	CLASSIFIED AS
6183	542	1504	1687	50	**A = Google**
954	5864	1194	1917	35	**B = Skype**
8	0	9915	77	0	**C = Spotify**
138	158	4652	4949	0	**D = WhatsApp**
2303	237	1295	1666	4463	**E = YouTube**

Table 20. Confusion matrix of IBK algorithm with CrossValidation, training dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.645	0.620	0.632
Skype	0.862	0.589	0.700
Spotify	0.534	0.992	0.694
WhatsApp	0.481	0.500	0.490
YouTube	0.981	0.448	0.615
Weighted Avg	0.701	0.630	0.627

Table 21. Precision, Recall and F-Measures of IBK algorithm with CrossValidation, training dataset.

Kappa Statistic: 0.5376

- **Test Dataset**

A	B	C	D	E	CLASSIFIED AS
6144	578	1477	1729	50	**A = Google**
1001	5886	1124	1923	37	**B = Skype**
3	0	9911	86	0	**C = Spotify**
133	145	4522	5092	0	**D = WhatsApp**
2274	246	1346	1738	4359	**E = YouTube**

Table 22. Confusion matrix of IBK algorithm with CrossValidation, test dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.643	0.616	0.629
Skype	0.859	0.590	0.700
Spotify	0.539	0.991	0.698
WhatsApp	0.482	0.515	0.498
YouTube	0.980	0.438	0.605
Weighted Avg	0.701	0.630	0.626

Table 23. Precision, Recall and F-Measures of IBK algorithm with CrossValidation, test dataset.

Kappa Statistic: 0.5378

With the Kappa statistics of both the training and test datasets it can be noted that the value is 0.5, more or less in the middle of the range in which this statistic is defined. This means that it is a classifier that is quite affected by chance, showing low F-Measures that show that it is not a very reliable classifier. This can also be seen in the confusion matrices where their diagonals despite having larger numbers, their surroundings are full of numbers similar to these.

5.3.6 NaiveBayes with CrossValidation

- **Training Dataset**

A	B	C	D	E	CLASSIFIED AS
8275	1263	2	15	411	**A = Google**
316	9527	0	4	117	**B = Skype**
31	40	9834	88	7	**C = Spotify**
94	261	813	8671	58	**D = WhatsApp**
256	218	0	1	9489	**E = YouTube**

Table 24. Confusion matrix of NaiveBayes algorithm with CrossValidation, training dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.922	0.839	0.874
Skype	0.842	0.956	0.896
Spotify	0.923	0.983	0.952
WhatsApp	0.988	0.876	0.929
YouTube	0.941	0.952	0.947
Weighted Avg	0.923	0.920	0.919

Table 25. Precision, Recall and F-Measures of NaiveBayes algorithm with CrossValidation, training dataset.

Kappa Statistic: 0.8997

- **Test Dataset**

A	B	C	D	E	CLASSIFIED AS
8312	1244	0	22	400	**A = Google**
311	9513	0	6	141	**B = Skype**
32	49	9827	84	8	**C = Spotify**
107	249	839	8648	49	**D = WhatsApp**
223	252	0	2	9486	**E = YouTube**

Table 26. Confusion matrix of NaiveBayes algorithm with CrossValidation, test dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.925	0.833	0.877
Skype	0.841	0.954	0.894
Spotify	0.921	0.983	0.951
WhatsApp	0.987	0.874	0.927
YouTube	0.941	0.952	0.946
Weighted Avg	0.923	0.919	0.919

Table 27. Precision, Recall and F-Measures of NaiveBayes algorithm with CrossValidation, test dataset.

Kappa Statistic: 0.8991

This being the last algorithm analyzed, taking into account the Kappa statistics can be inferred that this is also one of the best algorithms that were tested. The kappa statistic is close to 0.900, which makes it considerably close to 1, making this a classifier that is not very affected by chance. The F-Measures are also close to 1 making this a reliable algorithm. Also on the diagonal of the confusion matrices it can be noticed that the larger numbers belong to it and that in its surroundings the numbers are considerably smaller.

5.3.7 T-test with CrossValidation

DATASET	J48	RANDOMFOREST	ADABOOSTM1	BAGGING	IBK	NAIVEBAYES
Training dataset	90.64	79.39*	86.26	92.85v	63.14*	92.00v
	(v/ /*)	(0/0/1)	(0/1/0)	(1/0/0)	(0/0/1)	(1/0/0)

Table 28. T-test comparing the algorithms.

In the T-test made with CrossValidation of 10 folds for all the algorithms that were evaluated, J48 was taken as the algorithm with which the others were going to be compared. The results obtained were that Bagging was the best, then NaiveBayes, followed by J48. However, the difference between the performances of these three algorithms is not very big so the v next to the percentages given in the table indicates that any of the three algorithms can be used to make a very good validation for the dataset. It can also be seen that the result of the Adaboost algorithm has the middle sign, which indicates that it is an algorithm that is neither good nor bad, on the contrary it can be used to make a good classification, however it is not the most recommendable to do it. Finally it can be noticed that there is an * next to the percentages of RandomForest and IBK, leaving in evidence that these two algorithms do not have a good performance and that to make the validation of this set of data are not appropriate because the results obtained are very poor.

5.3.8 J48 with percentage Split

- **Training Dataset**

A	B	C	D	E	CLASSIFIED AS
2598	314	0	27	392	**A = Google**
319	2961	0	2	116	**B = Skype**
0	0	3378	16	0	**C = Spotify**
28	1	36	3312	14	**D = WhatsApp**
370	95	0	15	2935	**E = YouTube**

Table 29. Confusion matrix of J48 algorithm with Percentage Split, training dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.784	0.780	0.782
Skype	0.878	0.871	0.875
Spotify	0.989	0.995	0.992
WhatsApp	0.982	0.977	0.979
YouTube	0.849	0.859	0.854
Weighted Avg	0.897	0.897	0.897

Table 30. Precision, Recall and F-Measures of J48 algorithm with Percentage Split, training dataset.

Kappa Statistic: 0.8711

- **Test Dataset**

A	B	C	D	E	CLASSIFIED AS
8027	817	0	62	1072	**A = Google**
822	8829	2	11	307	**B = Skype**
0	0	9918	82	0	**C = Spotify**
57	15	80	9707	33	**D = WhatsApp**
988	300	2	46	8627	**E = YouTube**

Table 31. Confusion matrix of J48 algorithm with Percentage Split, test dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.811	0.804	0.808
Skype	0.886	0.885	0.886
Spotify	0.992	0.992	0.992

WhatsApp	0.980	0.981	0.981
YouTube	0.859	0.866	0.863
Weighted Avg	0.906	0.906	0.906

Table 32. Precision, Recall and F-Measures of J48 algorithm with Percentage Split, test dataset.

Kappa Statistic: 0.8821

In this test it can be seen that the confusion matrix of the training dataset has far fewer instances than the confusion matrix of the test dataset. This is because with the percentage split test the learning is done with 66% of the attributes that exist in the dataset and the rest are used for testing. In the kappa statistics it can be appreciated that the algorithm has a good performance and is not affected by chance, having in the diagonal of its confusion matrices the biggest numbers and with quite good F-Measures for which the algorithm is reliable.

5.3.9 RandomForest with percentage Split

- **Training Dataset**

A	B	C	D	E	CLASSIFIED AS
1899	676	37	106	613	**A = Google**
557	2371	5	99	366	**B = Skype**
3	5	3274	76	36	**C = Spotify**
31	78	525	2563	194	**D = WhatsApp**
651	241	12	52	2459	**E = YouTube**

Table 33. Confusion matrix of RandomForest algorithm with Percentage Split, training dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.605	0.570	0.587
Skype	0.703	0.698	0.701
Spotify	0.850	0.965	0.904
WhatsApp	0.885	0.756	0.815
YouTube	0.670	0.720	0.694
Weighted Avg	0.743	0.742	0.741

Table 34. Precision, Recall and F-Measures of RandomForest algorithm with Percentage Split, training dataset.

Kappa Statistic: 0.6778

- **Test Dataset**

A	B	C	D	E	CLASSIFIED AS
2603	2109	341	283	4642	**A = Google**
459	5398	468	376	3270	**B = Skype**
8	371	8924	679	18	**C = Spotify**
47	984	2646	4065	2150	**D = WhatsApp**
551	1086	84	83	8159	**E = YouTube**

Table 35. Confusion matrix of RandomForest algorithm with Percentage Split, test dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.710	0.261	0.382
Skype	0.543	0.541	0.542
Spotify	0.716	0.892	0.795
WhatsApp	0.741	0.411	0.529
YouTube	0.447	0.819	0.579
Weighted Avg	0.631	0.585	0.565

Table 36. Precision, Recall and F-Measures of RandomForest algorithm with Percentage Split, test dataset.

Kappa Statistic: 0.4815

The performance of this test with the RandomForest algorithm is considerably lower, leaving evidence that it is a classifier very affected by chance, very unreliable and low performance because in the diagonals of the confusion matrices is not seen eloquently that their numbers are much larger than those around them.

5.3.10 AdaboostM1 (J48) with percentage Split

- **Training Dataset**

A	B	C	D	E	CLASSIFIED AS
3154	135	0	14	28	**A = Google**
416	2970	0	3	9	**B = Skype**
0	0	3392	2	0	**C = Spotify**
21	0	35	3334	1	**D = WhatsApp**
749	96	0	8	2562	**E = YouTube**

Table 37. Confusion matrix of AdabosstM1 algorithm with Percentage Split, training dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.727	0.947	0.822
Skype	0.928	0.874	0.900

Spotify	0.990	0.999	0.995
WhatsApp	0.992	0.983	0.988
YouTube	0.985	0.750	0.852
Weighted Avg	0.925	0.910	0.912

Table 38. Precision, Recall and F-Measures of AdaboostM1 algorithm with Percentage Split, training dataset.

Kappa Statistic: 0.888

- **Test Dataset**

A	B	C	D	E	CLASSIFIED AS
5801	2672	0	60	1445	**A = Google**
74	9818	1	14	64	**B = Skype**
0	0	9994	6	0	**C = Spotify**
13	1	78	9788	12	**D = WhatsApp**
131	555	0	21	9256	**E = YouTube**

Table 39. Confusion matrix of AdaboostM1 algorithm with Percentage Split, test dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.964	0.581	0.725
Skype	0.753	0.985	0.853
Spotify	0.992	0.999	0.996
WhatsApp	0.990	0.989	0.990
YouTube	0.859	0.929	0.893
Weighted Avg	0.911	0.897	0.891

Table 40. Precision, Recall and F-Measures of AdaboostM1 algorithm with Percentage Split, test dataset.

Kappa Statistic: 0.8708

This test is much better than the previous one, it can be seen that the kappa statistics are around the values 0.87 and 0.89, which allows us to conclude that it is not a classifier that is very affected by chance. The F-Measure are close to zero, and the confusion matrices in their diagonals have the largest numbers showing that it is a very good performance and quite reliable algorithm.

5.3.11 Bagging (J48) with percentage Split

- **Training Dataset**

A	B	C	D	E	CLASSIFIED AS
2614	451	0	22	244	**A = Google**
71	3278	1	1	47	**B = Skype**
0	0	3390	4	0	**C = Spotify**
13	0	39	3334	5	**D = WhatsApp**
173	151	0	10	3081	**E = YouTube**

Table 41. Confusion matrix of Bagging algorithm with Percentage Split, training dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.910	0.785	0.843
Skype	0.845	0.965	0.901
Spotify	0.988	0.999	0.994
WhatsApp	0.989	0.983	0.986
YouTube	0.912	0.902	0.907
Weighted Avg	0.929	0.927	0.926

Table 42. Precision, Recall and F-Measures of Bagging algorithm with Percentage Split, training dataset.

Kappa Statistic: 0.909

- **Test Dataset**

A	B	C	D	E	CLASSIFIED AS
7858	1374	0	67	679	**A = Google**
249	9596	0	15	111	**B = Skype**
0	0	9976	24	0	**C = Spotify**
29	2	84	9753	24	**D = WhatsApp**
425	484	1	43	9010	**E = YouTube**

Table 43. Confusion matrix of Bagging algorithm with Percentage Split, test dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.918	0.788	0.848
Skype	0.838	0.962	0.896
Spotify	0.992	0.998	0.995
WhatsApp	0.985	0.986	0.985
YouTube	0.917	0.904	0.911
Weighted Avg	0.930	0.927	0.927

Table 44. Precision, Recall and F-Measures of Bagging algorithm with Percentage Split, test dataset.

Kappa Statistic: 0.9094

In this test, as in the previous one, it can be seen that Bagging is the algorithm that gives the best results when classifying datasets. The kappa are very close to 1, without evidencing a very big difference between the training dataset and the test dataset. Similarly, the F-Measures show that it is a very reliable classifier and in the confusion matrices it can be seen that the classifier has a great performance.

5.3.12 IBK (25 neighbors) with percentage Split

- **Training Dataset**

A	B	C	D	E	CLASSIFIED AS
2046	182	543	549	11	**A = Google**
348	1959	404	677	10	**B = Skype**
4	0	3364	26	0	**C = Spotify**
51	58	1585	1696	1	**D = WhatsApp**
832	73	512	543	1455	**E = YouTube**

Table 45. Confusion matrix of IBK algorithm with Percentage Split, training dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.624	0.614	0.619
Skype	0.862	0.577	0.691
Spotify	0.525	0.991	0.686
WhatsApp	0.486	0.500	0.493
YouTube	0.985	0.426	0.595
Weighted Avg	0.697	0.621	0.617

Table 46. Precision, Recall and F-Measures of IBK algorithm with Percentage Split, training dataset.

Kappa Statistic: 0.5269

- **Test Dataset**

A	B	C	D	E	CLASSIFIED AS
6144	578	1477	1729	50	**A = Google**
1001	5886	1124	1923	37	**B = Skype**
3	0	9911	86	0	**C = Spotify**
133	145	4522	5092	0	**D = WhatsApp**
2274	246	1346	1738	4359	**E = YouTube**

Table 47. Confusion matrix of IBK algorithm with Percentage Split, test dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.643	0.616	0.629
Skype	0.859	0.590	0.700
Spotify	0.539	0.991	0.689
WhatsApp	0.482	0.515	0.498
YouTube	0.980	0.438	0.605
Weighted Avg	0.701	0.630	0.626

Table 48. Precision, Recall and F-Measures of IBK algorithm with Percentage Split, test dataset.

Kappa Statistic: 0.5378

This test shows that this algorithm is one of the worst ever tested, with a kappa statistic showing that it is a classifier highly affected by chance. The F- Measures also show how unreliable it is and the confusion matrices corroborate that the performance is very poor.

5.3.13 NaiveBayes with percentage Split

- **Training Dataset**

A	B	C	D	E	CLASSIFIED AS
2773	424	0	3	131	**A = Google**
102	3246	0	2	48	**B = Skype**
9	9	3347	28	1	**C = Spotify**
27	72	312	2960	20	**D = WhatsApp**
82	80	0	0	3253	**E = YouTube**

Table 49. Confusion matrix of NaiveBayes algorithm with Percentage Split, training dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.926	0.832	0.877
Skype	0.847	0.955	0.898
Spotify	0.915	0.986	0.949
WhatsApp	0.989	0.873	0.927
YouTube	0.942	0.953	0.947
Weighted Avg	0.924	0.920	0.920

Table 50. Precision, Recall and F-Measures of NaiveBayes algorithm with Percentage Split, training dataset.

Kappa Statistic: 0,9003

- **Test Dataset**

A	B	C	D	E	CLASSIFIED AS
8312	1244	0	22	400	**A = Google**
311	9513	0	6	141	**B = Skype**
32	49	9827	84	8	**C = Spotify**
107	249	839	8648	49	**D = WhatsApp**
223	252	0	2	9486	**E = YouTube**

Table 51. Confusion matrix of NaiveBayes algorithm with Percentage Split, test dataset.

CLASS	PRECISION	RECALL	F-MEASURE
Google	0.925	0.833	0.877
Skype	0.841	0.954	0.894
Spotify	0.921	0.983	0.951
WhatsApp	0.987	0.874	0.927
YouTube	0.941	0.952	0.946
Weighted Avg	0.923	0.919	0.919

Table 52. Precision, Recall and F-Measures of NaiveBayes algorithm with Percentage Split, test dataset.

Kappa Statistic: 0.8991

This classifier, together with Bagging and J48, are the best performers in both the CrossValidation and Split percentage tests. Kappa statistics are very close to 1 and F-Measures show great reliability. The confusion matrices also show a very good performance of the algorithm being one of the best to do the validation of the different datasets.

Summary

In this chapter a brief summary of the metrics used in the machine learning classification algorithms is made, followed by a brief description of the algorithms used to do the tests and finally a short description of the tests that were performed to do the validation of the datasets.

It also exposes some of the problems that were presented when doing this validation with the different datasets and the solution that was chosen to carry out the validation of the datasets created.

Finally, the results obtained with the classification algorithms are shown, taking into account that the results were shown in the following way: as a first measure, the results of the dataset that had 50,000 instances that passed through the cleaning process were shown, doing the validation with algorithms such as J48, RandomForest, AdaboostM1, Bagging, IBK and NaiveBayes, making use of cross validation with 10 folds, showing the tests done in the training dataset and in the test dataset each of them having 49.971 instances and 49.804 instances respectively. Then it is shown the T-test performed with cross validation of 10 fold in order to make a comparison between the algorithms used. Finally the results obtained with these same algorithms are shown in the test and training dataset, but this time with the percentage split test. At the end of each test shown is given a brief explanation and analysis of the results that were obtained.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This section presents the conclusions that were obtained from this undergraduate thesis and also presents some of the future work that can be done for this area of research.

6.1 Conclusions

In this section the conclusions obtained for this undergraduate thesis were:

- After an extensive research and revision of the works related to this undergraduate thesis, it was reached the conclusion that until now there are no researches or similar projects that resemble what it is wanted to achieve with this work, taking into account that until now nobody has taken the initiative to integrate LTE simulators, with traffic generators, traffic capturers and that as a final result a dataset is obtained from them with all the information exchanged in the network.

- After making a study of the traffic generators exposed in this undergraduate thesis, it can be said that there is no generator capable of meeting the needs of this research work, which were being able to generate flows of different OTT applications having the absolute certainty of knowing to which exact application they belong, i.e., knowing if the flow belongs to Skype, Google, Spotify, and so on.

- The generation of synthetic flow of OTT applications can be done with a previous statistical modeling of a model dataset that contains instances of real OTT applications.

- With the OTT applications synthetic flow generator created in this undergraduate thesis, datasets can be obtained from different applications such as WhatsApp, YouTube, Skype, Spotify and Google.

- The dataset that was created, taking into account the results of the machine learning tests, is a good dataset since the different algorithms used gave good results, with which it can be concluded that the synthetic flow generators created meet the needs of this undergraduate thesis.

6.2 Future work

Considering the research that was done in this undergraduate thesis, the future works are:

- Collect a much larger dataset that has a greater number of users and a greater number of OTT applications and make the statistical modeling of flows of many more OTT applications, such as Netflix, Deezer, Instagram, Twitter, etc.

- Synthesize the flow of OTT application attributes without taking into account statistical independence. That is, analyzing the correlation that may or may not exist between one attribute and another.

- Create an environment with a user interface so that the generators created in this thesis are easy to use for the general public, having only to launch the program and being able to obtain datasets with the exchange of information from specific OTT applications.

- Make an integration of the exposed LTE simulator, with the OTT application flow generator that was created in this undergraduate thesis.

- To create an integrated environment of an LTE network simulator, an OTT applications traffic generator and a sniffer, that allows to create different number of users and servers, capable of exchanging data between them, where the user will have as final product a dataset with all the information of the data implicitly exchanged.

BIBLIOGRAPHY

[1] G. Drapper-Gil, . A. H. Lashkari, M. S. Islam Mamun and A. A. Ghorbani, "Characterization of Encrypted and VPN Traffic Using Time-Related Features," *In Proceedings of the 2nd International Conference on Information Systems Security and Privacy,* vol. 2016, pp. 1-9, Feb 2016.

[2] N. R. Brnovic, "Trends in the Evolution of Voice Services: A Comprehensive Survey," *Springer Science+Business Media,* p. 8, 26 Julio 2017.

[3] Y. Zaki, L. Zhao, C. Goerg and A. Timm-Giel, "LTE mobile network virtualization," *Springer Science+Business Media,* p. 9, 2011 Junio 2011.

[4] I. Orsolic, D. Pevec, M. Suznjevic and L. Skorin-Kapov, "A machine learning approach to classifying YouTube QoE based on encrypted network traffic," *Springer Science+Business Media,* 17 Abril 2017.

[5] B. Héder, P. Szilágyi and C. Vulkán, "Dynamic and Adaptive QoE Management for OTT Application Sessions in LTE," in *27th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC): Mobile and Wireless Networks*, 2016.

[6] V. Carela-Español, "Network Traffic Classification: From Theory to Practice," Barcelona, 2014.

[7] J. S. Rojas, J. C. Corrales, C. Olarte Rico, J. E. Plazas, J. A. Caicedo, A. M. Vargas, A. Rendon, E. J. Giron and C. Heidelberg Valencia, "Network Traffic Classification for Resource Management Based on a Bagging Hierarchical Approach," pp. 1-20, 2016.

[8] "Mobile Communication Technologies – Mobile Communication," [Online]. Available: http://freewimaxinfo.com/mobile-communication-technologies.html. [Accessed March 2019].

[9] "Virtualized Evolved Packet Core for LTE Networks," Bombey, 2016.

[10] "3GPP, A global iniciative," [Online]. Available: http://www.3gpp.org/about-3gpp/about-3gpp. [Accessed March 2019].

[11] L. Coya Rey and T. O. Ledesma Quiñones, "Herramientas de monitorización y análisis del tráfico en redes de datos," *Revista Telem@tica,* vol. 11, no. 2, pp. 46-59, Aug 2012.

[12] S. Srivastava, S. Anmulwar, A. Sapkal, T. Batra, A. K. Gupta and V. Kumar, "Comparative study of various Traffic Generator Tools," in *Proceedings of 2014 RAECS UIET Panjab University Chandigarh*, Chandigarh, 2014.

[13] E. R. Mata, "Evaluación de herramientas para escenarios de generación de tráfico," Madrid, 2015.

[14] S. Molnar, P. Megyesi and G. Szabo, "How to Validate Traffic Generators?," in *IEEE International Conference on Communications 2013: IEEE ICC'13 - 1st IEEE Workshop on Traffic Identification and Classification for Advanced Network Services and Scenarios (TRICANS)*, 2013.

[15] R. Archanaa, V. Athulya and T. Rajasundari, "A comparative performance analysis on network traffic classification using supervised learning algorithms," in *4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, 2017.

[16] R. Antonello, S. Fernandes, D. Sadok and J. Kelner, "Characterizing Signature Sets for Testing DPI Systems," *2011 IEEE GLOBECOM Workshops (GC Wkshps)*, pp. 678-683, Dec 2011.

[17] A. C. Tsoi, J. McDonell, A. Treloar and I. Atkinson, "Dataset acquisition, accessibility, annotation, e-research technologies (DART) Project," *Springer-Verlag*, pp. 53-55, 12 May 2007.

[18] z. Concepts, "IBM," 2010. [Online]. Available: https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zconcepts/zconcepts_book.pdf. [Accessed Feb 2017].

[19] J. J. Ganuza and M. F. Viecens, "Over-the-top (OTT) applications, services and content: implications for broadband infrastructure," Santiago de Chile, 2013.

[20] S. Huang, K. Chen, C. Liu, A. Liang and H. Guan, "A Statistical-Feature-Based Approach to Internet Traffic Classification Using Machine Learning," pp. 1-6, 2009.

[21] I. El Naqa and M. J. Murphy, "What is Machine Learning?," in *Machine Learning in Radiation Oncology: Theory and Applications*, Springer International, 2015, p. 9.

[22] R. Archanaa, V. Athulya, T. Rajadundari and V. K. Kiran M, "A Comparative Performance Analysis on Network Traffic classification using Supervised learning algorithms," in *2017 International Conference on Advanced Computing and Communication Systems*, Coimbatore, 2017.

[23] M. Shafiq, X. Yu, A. A. Laghari, L. Yao, N. K. Karn and F. Abdessamia, "Network Traffic Classification Techniques and Comparative Analysis Using Machine Learning Algorithms," in *2016 2nd IEEE International Conference on Computer and Communications*, 2016.

[24] Y. Wang and S.-Z. Yu, "Machine Learned Real-time Traffic Classifiers," in *Second International Symposium on Intelligent Information Technology Application*, 2008.

[25] G. Piro, L. A. Grieco, G. Boggia, F. Capozzi and P. Camarda, "Simulating LTE Cellular Systems: An Open-Source Framework," vol. 60, no. 2, p. 16, 2011.

[26] *vEPC 1.0 Developer Manual*, 2016.

[27] K. Petersen, R. Feldt, S. Mujtaba and M. Mattsson, "Systematic Mapping Studies in Software Engineering," in *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, Swinton, 2008.

[28] A. Papadogiannakis, M. P. Michalis Polychronakis and E. P. Markatos, "Stream-Oriented Network Traffic Capture and Analysis for High-Speed Networks," *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, vol. 32, no. 10, pp. 1-15, Octubre 2014.

[29] R. Morla, P. Gonçalves and J. G. Barbosa, "High-performance network traffic analysis for continuous batch intrusion detection," 14 Mayo 2016.

[30] Q. Zheng, H. Du, J. Li, W. Zhang and Q. Li, *Open-LTE: An Open LTE Simulator For Mobile Video Streaming,* 2014.

[31] A. Virdis, G. Stea and G. nardini, *Simulating LTE/LTE-Advanced Networks with SimuLTE,* Pisa, 2015.

[32] R. Singh, H. Kumar and R. K. Singla, "Analysis of Feature Selection Techniques for Network Traffic Dataset," *International Conference on Machine Intelligence Research and Advancement,* p. 5, 2013.

[33] C. Garcia Cordero, E. Vasilomanolakis, N. Milanov, C. Koch, D. Hausheer and M. Mühlhäuser, "ID2T: a DIY Dataset Creation Toolkit for Intrusion Detection Systems," *IEEE CNS 2015 Poster Session,* p. 2, 2015.

[34] V. D. Blondel, A. Decuyper and G. Krings, "A survey of results on mobile phone datasets analysis," *EPJ Data Science,* p. 55, 2015.

[35] Q. Han, "Social Influence Analysis using Mobile Phone Dataset," *17th IEEE International Conference on Mobile Data Management,* vol. 17, p. 6, 2016.

[36] W. Wang, M. Zhu, J. Wang, X. Zeng and Z. Yang, *End-to-end Encrypted Traffic Classification with One-dimensional Convolution Neural Networks,* 2017, p. 6.

[37] P. Ducange, G. Mannara, F. Marcelloni, R. Pecori and M. Vecchio, *A Novel Approach for Internet Traffic Classification based on Multi-Objective Evolutionary Fuzzy Classifiers,* Pisa, 2017, p. 6.

[38] R. B. Jerome and K. Hatonen, "Anomaly detection and classification using a metric for determining the significance of failures. Case study: mobile network management data from LTE network," *The Natural Computing Applications Forum ,* vol. 2016, p. 11, 17 Agosto 2016.

[39] E. Liotou, G. Tseliou, K. Samdanis, D. Tsolkas, F. Adelantado and C. Verikoukis, "An SDN QoE-Service for Dynamically Enhancing the Performance of OTT Applications," p. 2, 2015.

[40] "Tcpreplay," [Online]. Available: https://tcpreplay.appneta.com/wiki/history.html. [Accessed 2019].

[41] W.-c. Feng, A. Goel, A. Bezzaz, W.-c. Feng and J. Walpole, *TCPivo: A High-Performance Packet Replay Engine,* 2003.

[42] "TCPivo: A High Performance Packet Replay Engine," [Online]. Available: https://www.thefengs.com/wuchang/work/tcpivo/. [Accessed 2019].

[43] "iPerf - The ultimate speed test tool for TCP, UDP and SCTP," [Online]. Available: https://iperf.fr/. [Accessed 2019].

[44] "GitHub," [Online]. Available: https://github.com/awgn/brute. [Accessed 2019].

[45] N. Bonelli, S. Giordano, G. Procissi and R. Secchi, *BRUTE: A High Performance and Extensible Traffic Generator,* Pisa, 2005.

[46] G. Antichi, A. Di Pietro, D. Ficara, S. Giordano, G. Procissi and F. Vitucci, "BRUNO: A high performance traffic generator for network processor," in *Performance Evaluation of Computer and Telecommunication Systems, 2008. SPECTS 2008.,* 2008.

[47] 6 March 2007. [Online]. Available: http://caia.swin.edu.au/genius/tools/kute/. [Accessed 2019].

[48] P. Srivats, "Ostinato," [Online]. Available: https://ostinato.org/. [Accessed 2019].

[49] "Networks and Communication Systems Branch," [Online]. Available: https://www.nrl.navy.mil/itd/ncs/products/mgen. [Accessed 2019].

[50] J. E. Sommers, *Harpoon,* 2005.

[51] K. V. Vishwanath and A. Vahdat, *Swing: Realistic and Responsive Network Traffic Generation.*

[52] "Old Dominion University," 2006. [Online]. Available: https://www.cs.odu.edu/~inets/Public/Tmix. [Accessed 2019].

[53] M. C. Weigle, P. Adurthi, F. Hernandez-Campos, K. Jeffay and F. D. Smith, *Tmix: A Tool for Generating Realistic TCP Application Workloads in ns-2,* 2006.

[54] C. Rolland, J. Ridoux and B. Baynat, *LiTGen, a Lightweight Traffic Generator: Application to P2P and Mail Wireless Traffic,* 2007.

[55] A. Pescape, A. Botta, W. de Donato and G. Aceto, "D-ITG, Distributed Internet Traffic Generator," [Online]. Available: http://www.grid.unina.it/software/ITG/authors.php. [Accessed 2019].

[56] A. Botta, W. de Donato, A. Dainotti, S. Avallone and A. Pescape, *D-ITG 2.8.1 Manual,* Napoli, 2013.

[57] "Tomahawk," [Online]. Available: http://tomahawk.sourceforge.net/. [Accessed 2019].

[58] A. Y. C. Heng, "Bit-Twist," 2012. [Online]. Available: http://bittwist.sourceforge.net/index.html. [Accessed 2019].

[59] M. Jemec, "PackETH," [Online]. Available: http://packeth.sourceforge.net/packeth/Home.html. [Accessed 2019].

[60] J. Laine, S. Saaristo and R. Prior, "RUDE & CRUDE," 2002. [Online]. Available: http://rude.sourceforge.net/#install. [Accessed 2019].

[61] A. Hafsaoui, N. Nikaein and L. Wang, "OpenAirInterface Traffic Generator (OTG): A Realistic Traffic Generation Tool for Emerging Application Scenarios," in *IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems,* Arlington, 2012.

[62] K. Wiles, "The Pktgen Application," 2010. [Online]. Available: https://pktgen-dpdk.readthedocs.io/en/latest/. [Accessed 2019].

[63] D. Borkmann, "Trafgen - Linux Man Pages," [Online]. Available: https://www.systutorials.com/docs/linux/man/8-trafgen/. [Accessed 2019].

[64] V. Shah, "Mausezahn - Linux Man Page," [Online]. Available: https://www.systutorials.com/docs/linux/man/1-mausezahn/. [Accessed 2019].

[65] I. Northwest Performance Software, "NetScan Tools," [Online]. Available: https://www.netscantools.com/nstpro_packet_generator.html#notes. [Accessed 2019].

[66] "GL Communications Inc.," [Online]. Available: https://www.gl.com/gigabit-network-traffic-generator-analyzer.html. [Accessed 2019].

[67] N. Inc., *TCPCopy Manual,* 2013.

[68] M. Ghobadi, G. Salmon, Y. Ganjali, M. Labrecque and J. G. Steffan, "Caliper: Precise and Responsive Traffic Generator," in *IEEE 20th Annual Symposium on High-Performance Interconnects*, 2012.

[69] "vEPC 1.1 User Manual," Bombay, 2016.

[70] J. S. Rojas, A. R. Gallon and J. C. Corrales, "Personalized Service Degradation Policies on OTT Applications Based on the Consumption Behavior of Users," *Springer International Publishing AG, part of Springer Nature 2018,* p. 15, 2018.

[71] J. S. Rojas, Personalized Service Degradation on OTT Applications, Popayan, 2018.

[72] "Wireshark," [Online]. Available: https://www.wireshark.org/index.html#aboutWS. [Accessed 2019].

[73] C. I. f. Cybersecurity, "UNB," [Online]. Available: https://www.unb.ca/cic/research/applications.html. [Accessed 2019].

[74] "ntop," [Online]. Available: https://www.ntop.org/products/traffic-analysis/ntop/. [Accessed 2019].

[75] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer and R. Wirth, CRISP-DM 1.0 Step-by-step data mining guide, 2000.

[76] D. C. Corrales, J. C. Corrales and A. Ledezma, "How to Address the Data Quality Issues in Regression Models: A Guided Process for Data Cleaning," *Symmetry,* vol. 10, no. 99, p. 20, 2018.

[77] "RStudio," [Online]. Available: https://www.rstudio.com/products/rstudio/features/. [Accessed 2019].

[78] B. McNeese, "SPC for EXCEL," February 2016. [Online]. Available: https://www.spcforexcel.com/knowledge/basic-statistics/are-skewness-and-kurtosis-useful-statistics#kurtosis. [Accessed 2019].

[79] . D. M. Kelmansky , "ANÁLISIS DE DATOS," 2008. [Online]. Available: http://www.dm.uba.ar/materias/analisis_de_datos/2008/1/teoricas/Teor5.pdf. [Accessed 2019].

[80] J. E. Gentle, Conputational Statistics, Springer, 2009.

[81] J. D. Gibbons and S. Chakraborti, Nonparametric Statistical Inference, Fourth Edition: Revised and Expanded, 2014.

[82] M. Shafiq, X. Yu, A. A. Laghari, L. Yao, N. Kumar Karn and F. Abdessamia, "Network Traffic Classification Techniques and Comparative Analysis Using Machine Learning Algorithms," in *2nd IEEE International Conference on Computer and Communications,* 2016.

[83] G. Al-Naymat and M. Alkasassbeh, "Classification of VoIP and non-VoIP traffic using machine learning approaches," *Journal of Theoretical and Applied Information Technology,* vol. 92, no. 2, p. 13, 2016.

[84] I. H. Witten, E. Frank and M. A. Hall, DATA MINING: Practical machine learning tools and techniques, Elsevier, 2011.

[85] D. Opitz, 24 August 1999. [Online]. Available: https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume11/opitz99a-html/node3.html. [Accessed 2019].

[86] S. Kumar Patra and S. C. Santosh Prasad, "A Survey of different classification techniques and their comparison using Mc Nemar’s Test," Rourkela, 2013.