UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MASTER DEGREE IN SIGNAL THEORY AND COMMUNICATIONS

MASTER THESIS

NETWORK ANOMALY DETECTION USING MACHINE LEARNING TECHNIQUES

PILAR SCHÜMMER BENGOA

2024

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MASTER DEGREE IN SIGNAL THEORY AND COMMUNICATIONS

MASTER THESIS

NETWORK ANOMALY DETECTION USING MACHINE LEARNING TECHNIQUES

Author
PILAR SCHÜMMER BENGOA

Tutors and Supervisor

ALBERTO DEL RÍO PONCE
JAVIER SERRANO ROMERO
LUIS ALFONSO HERNÁNDEZ GÓMEZ

2024

Resumen

Durante la última década diferentes científicos han presentado múltiples propuestas para definir valores atípicos en un conjunto de datos, sin que hasta la fecha se haya alcanzado un consenso claro. A pesar de esta ambigüedad en la definición de las anomalías, toda la sociedad trata de evitar la presencia de éstas en cualquier ámbito o escenario. Un tema de actualidad son las anomalías de red, los cuales son datos que se encuentran fuera de la norma, y no es posible dar una definición estricta debido a la gran variedad de anomalías que pueden presentarse. Mediante el uso de sistemas de detección de anomalías, los administradores de red son capaces de resolver anomalías en un tiempo considerablemente menor que sin el uso de estos sistemas. La detección temprana y eficaz de anomalías en la red es un campo de investigación que la mayoría de las empresas e instituciones públicas necesitan para evitar fallos o averías en el servicio. Por este motivo, un sistema adecuado de detección de anomalías es de vital importancia en cualquier ámbito, pero especialmente en las redes de comunicaciones.

Para solventar este problema y aportar investigacion a la comunidad cientifica, se desarrollan diferentes algoritmos de aprendizaje automático que permitan detectar de forma objetiva y precisa anomalias de red. Para lograr este objetivo, se diseñó una topología de red adecuada para su caracterización y se implementó en un emulador que permitiera generar tráfico de red. Posteriormente, se generó un conjunto de datos mediante el envío de contenido multimedia en un entorno controlado, recopilando información sobre el estado de la red y las métricas obtenidas a través de una sonda de red.

A continuación, se realizó un análisis exploratorio de los datos y se desarrollaron varios modelos de aprendizaje automático para implementar un sistema de detección de anomalías. Para los modelos supervisados, se estableció un umbral para la etiquetación de las anomalías y se entrenaron los modelos para identificar los datos anómalos. Además, se desarrollaron técnicas para explicar las predicciones de los modelos y técnicas no supervisadas, como el clustering, que no requieren datos etiquetados.

Finalmente, se compararon los resultados obtenidos con ambas técnicas. Mediante el uso de algoritmos como Logistic Regression, Random Forest, Support Vector Machine, se lograron valores de precisión del 91% al 97%. Esto demuestra que, a través de un análisis y entrenamiento preliminar de la red, es posible establecer patrones efectivos para la detección de anomalías. Los resultados obtenidos validan que el sistema desarrollado proporciona detecciones objetivas, precisas y automáticas, contribuyendo significativamente al campo del monitoreo y seguridad de redes.

Palabras clave: Anomalía, Aprendizaje automático, Detección, Inteligencia Artificial, Predicción, Tráfico de red.

Abstract

During the last decade different scientists have presented multiple proposals to define outliers in a dataset, without reaching a clear agreement to date. Despite this ambiguity in the definition of outliers, society attempts to avoid the presence of outliers in any environment or scenario. A topical issue is network anomalies, which are data that are outside the norm, and it is not possible to give a strict definition due to the wide variety of anomalies that can occur. By using anomaly detection systems, network administrators are able to resolve anomalies in considerably less time than without the use of these systems. Early and effective detection of network anomalies is a field of research that most companies and public institutions need in order to avoid service failures or breakdowns. For this reason, an adequate anomaly detection system is of vital importance in any field, but especially in communications networks.

To solve this problem and contribute research to the scientific community, different machine learning algorithms are developed to objectively and accurately detect network anomalies. To achieve this purpose, a network topology suitable for its characterization was designed and implemented in an emulator to generate network traffic. Subsequently, a dataset was generated by sending multimedia content in a controlled environment, collecting information about the network status and metrics obtained through a network probe.

Then, an exploratory analysis of the data was performed and several machine learning models were developed to implement an anomaly detection system. For the supervised models, a threshold for anomaly labeling was established and the models were trained to identify anomalous data. In addition, techniques were developed to explain model predictions and unsupervised techniques, such as clustering, that do not require labeled data.

Finally, the results obtained with both techniques were compared. By using algorithms such as Logistic Regression, Random Forest, Support Vector Machine, accuracy values from 91% to 97% were achieved. This shows that, through a preliminary analysis and training of the network, it is possible to establish effective patterns for anomaly detection. The results obtained validate that the developed system provides objective, accurate and automatic detections, contributing significantly to the field of network monitoring and security.

Keywords: Anomaly, Machine learning, Detection, Artificial intelligence, Prediction, Network traffic.

Acknowledgement

I would first like to thank my tutors, Alberto del Río Ponce, Javier Serrano Romero and David Jiménez Bermejo, for their help throughout the entire master thesis and for their willingness to help me at any time so that this project achieved the best results. Secondly, I would like to thank Luis Alfonso Hernández Gómez, supervisor of this project and professor of the master, for the knowledge he has provided to me.

I would also like to thank my family and my partner, especially in these last months for their patience, encouragement and kindness in every moment, but especially when I was overwhelmed and when things were challenging.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AI: Artificial Intelligence.

CLI: Command Line Interface.

CP: Change Points.

CPU: Central Processing Unit.

DTW: Dynamic Time Warping.

IP: Internet Protocol.

IOS: Internetwork Operating System.

ISP: Internet Service Provider.

IT: Information Technology.

LSTM: Long Short-Term Memory.

MEC: Multi-access edge computing.

ML: Machine Learning.

NAB: Numenta Anomaly Benchmark.

PCA: Principal Component Analysis.

SHAP: SHapley Additive exPlanations.

SSH: Secure Shell.

TCP: Transmission Control Protocol.

t-SNE: t-Distributed Stochastic Neighbor Embedding.

UDP: User Datagram Protocol.

Chapter 1

Introduction

Data networks have changed the way modern communications operate nowadays. Connectivity and speed are essential, having an efficient networking data system has become a must [1]. A data system refers to the technological infrastructure that enables communication and information exchange between different devices. This includes computers, servers and other connected devices as shown in Figure 1.1.

Therefore, there is a need to ensure that all data movements are correct, and if this is not the case, to detect the problem and solve it as soon as possible. It is because of this need that anomaly detection systems arise, capable of helping with this task. The anomaly detection strategy starts with identifying the key performance indicators, as well as understanding the characteristics of the data [2]. How is it streaming into the network? Is it continuous or in batches? Which data points are being tracked?

In general, detection systems are based on Machine Learning (ML). These models need a source of information to be trained and therefore capable of detecting anomalies. In the same way that data networks have revolutionized communications, anomaly detection systems based on ML are transforming the way anomalies are identified and tackled, using advanced algorithms to detect problems, ensuring the integrity of the environment.

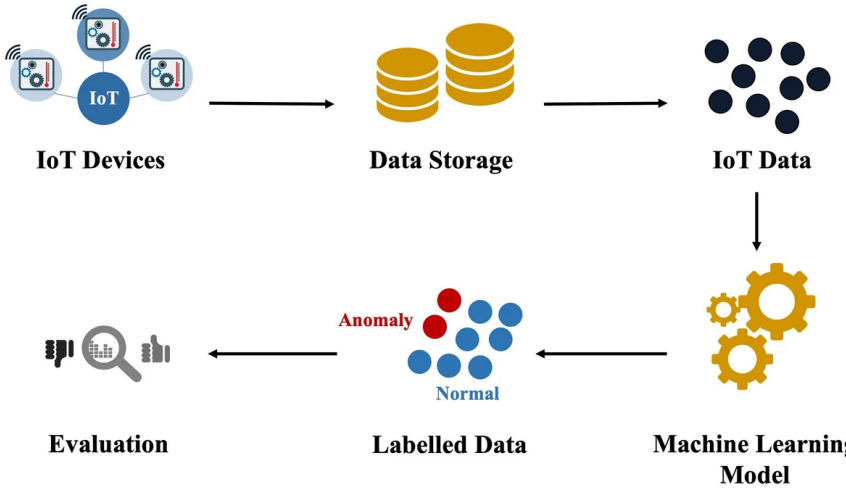


Figure 1.1: Machine learning workflow for anomaly detection (extracted from [3]).

The prediction of anomalies is the next step in preventing errors that may occur in the future. By forecasting unusual patterns, proactive measures can be taken to avoid potential issues. Looking for them helps to prevent damage to a system before it occurs. For example, a hospital that does not know what an attack will be like can benefit from predictability. With predictability, the hospital can design rules to prevent the threat, protect sensitive information and its infrastructure.

1.1. Background and motivation

In network infrastructures, anomaly detection is a fundamental role in the preservation and maintenance of systems and environments. Every day that passes, information technology (IT) professionals must face new threats and challenges due to the continuous evolution of Internet networks and communications systems. Because of this, it is essential to develop and further research the creation of efficient and advanced anomaly detection systems that can identify anomalies, allowing IT professionals to respond effectively in the resolution of these anomalies, avoiding collateral damage.

Anomalies can manifest themselves in a variety of manners, including unauthorized access, network crashes or unusual traffic patterns. Numerous research teams carry out new investigations every day to solve this issue, addressing it through the use of different techniques. Research papers described anomaly detection techniques used, for instance, to intrusion systems [4], unusual patterns [5] or interruption of a service by overloading its infrastructure due to traffic avalanches [6].

The above solutions achieved high satisfactory theoretical results, but when we started the project, the existing detection systems did not categorize the network or the data used. Therefore, during this master thesis, it was wanted to develop an anomaly detection system that would consider the network environment, the different situations in the traffic and the variables used for the creation of the dataset. An essential aspect of this project is to develop a prototype that can be utilized in a real security system. In order to make this possible, it was decided to implement anomaly detection algorithms that could operate autonomously and integrate them into a data application to facilitate its utilization by users and its subsequent deployment.

1.2. Objectives

The purpose of this master thesis is to design and implement Artificial Intelligence (AI) algorithms for network anomaly detection. This thesis aims to analyze anomalies in network transmissions, seeking to develop a system capable of identifying anomalous patterns and behaviors.

For this project, several tools will be used to simulate anomalies in different network scenarios. The algorithms for the development of the detection system will be carried out using ML techniques due to its numerous advantages and the efficiency it provides, as well as accurate results.

The system should provide an objective, accurate and automatic evaluation of

network traffic parameters for decision making, avoiding network crashes. This will enable early detection and resolution of network anomalies, essential in the current society. For the development of the automatic anomaly detection system, the following steps were followed:

- Review of the different existing software and applications for the detection of anomalies, specially in communication networks.
- Development of a dataset for anomaly detection in an emulated network environment and network categorization.
- Development of different algorithms based on Machine Learning whose purpose is the detection of network anomalies.
- Design of a software that allows the classification of data into usual or anomalous, and the validation and analysis of the automatic system for the detection of network anomalies by performing the measurement of various metrics.
- Development of prediction models capable of predicting the data by training them with metrics from the generated dataset.
- Implementation of an interactive web application that combines the ML models developed, enabling the insertion of new data by the user and its subsequent detection of anomalies.

1.3. Document structure

This master thesis has been structured as follows:

- Chapter 1: introduction explaining the importance of this master thesis, definition of anomalies, different types of anomalies in networks, the relevance of the research and the detection of anomalies by different methods. It also describes the worldwide known metrics used for network traffic evaluation and the objective of this master thesis.
- Chapter 2: the main existing tools for anomaly detection are presented, as well as the available datasets for the development of Machine Learning based algorithms for anomaly detection. An exhaustive search of algorithms and frameworks focused on network anomaly detection is performed.
- Chapter 3: this section explains the procedure followed for the implementation of this master thesis. It details the steps from the installation of the virtual machine, the probe to perform the network traffic metrics to the development of the network topologies and scenarios for its emulation in the simulated scenario.
- Chapter 4: the emulated scenario generated and the results obtained are presented, as well as their interpretation for a real environment.
- Chapter 5: the conclusions of the project are detailed, possible projections and future lines of this master thesis.

Chapter 2

State of the art

This Chapter describes a variety of tools of different levels of complexity and implementation for the analysis and detection of network anomalies.

2.1. Definition and types of anomalies

Anomalies correspond to the behavior of a system which does not conform to its expected or normal pattern [7]. Prior to creating a model capable of identifying anomalies, understanding the types of anomalies involved and their characteristics is essential for effective detection and analysis. There is no standardization of the types of anomalies common to all researchers, but there is a general idea of their classification. Two main types of anomalies can be distinguished (see Figure 2.1).

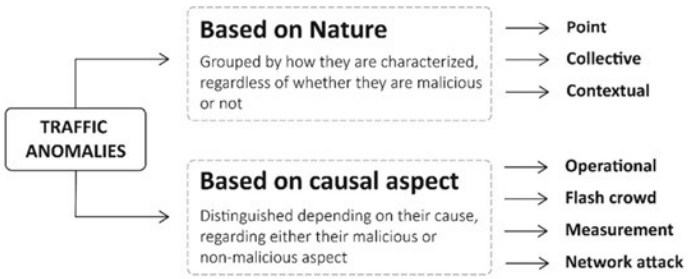


Figure 2.1: Network data types categorization (extracted from [8]).

Anomalies based on their nature are grouped according to how they are characterized, regardless of whether they are malicious or not. There are three categories within this division of anomalies [8].

- **Point anomaly:** is a deviation of an individual data that is outside its normal behavior [9]. This type of anomaly is the most basic because it does not depend on other characteristics of the dataset. For example, it is expected in a service to have a packet loss close to 0%, standard baseline. So, if we have a packet loss of 50% during several minutes, the system will detect it and will establish this drop as anomalous.

- **Collective anomaly:** is a deviation of the behavior of a set of data from the total dataset [10]. In a collective anomaly, individual anomalous behaviors are not considered individually anomalous, but their collective occurrence is considered an anomaly. For example, observing an increase in the use of a Central Processing Unit (CPU) and at the same time observing a gradual increase in the response time, even if the data is normal, we could attribute the increase in the response time to the high use of the CPU, thus establishing an association between the two services and therefore an anomaly.

- **Contextual anomaly:** is also known as conditional anomaly, i.e. they are considered as anomalies depending on the context [11]. For instance, the volume of packages in what is called a low period from 00:00 to 07:00 hours would be different from the daytime period, i.e. from 07:00 to 23:59, so a value that may seem very high or very low will always depend on a previous context. Another example would be the increase of Bizum transactions on December 25 or 31 due to all that is economically involved in these days [12].

Anomalies based on their nature they are distinguished on the basis of their cause, whether they are malicious or not. Within this division there are four categories of anomalies [8].

- **Operational events:** are non-malicious issues that occur either due to hardware failure, human error or misconfigurations. Examples of these types can be server crashes, traffic congestion or incorrect configuration. These types of anomalies can be detected as abrupt changes in data flow.

- **Flash crowds:** are anomalies in which data floods occur due to a large increase in the number of user accesses to a particular website. Although it is not a malicious anomaly, if there is not enough reaction time it can lead to complete server failure.

- **Measurement anomalies:** are not anomalies like the previous ones related to network infrastructure, but occur due to the data collection process.

- **Network abuse anomalies:** known as network attacks, which are malicious actions to degrade or destroy services and information. Examples of this type of anomalies are email spam or service intrusion attacks. These anomalies that limit or destroy resources can be caused by different approaches, such as social engineering techniques, identity fraud, software vulnerabilities or different implementations, as well as service congestion to produce a failure.

2.2. Anomaly detection methods

Different techniques can be implemented for anomaly detection depending on the particular characteristics of each system. Anomaly detection techniques can be classified into supervised techniques, which require labeled data to train the models,

and unsupervised techniques, which can detect anomalies without labeled data. Depending on the situation and the availability or not of labeled data, one or the other techniques are chosen for the development of anomaly detection systems.

Similarly, techniques can be based on specific rules that allow the detection of anomalies by deviation from their usual behavior, or techniques based on automatic learning, which are capable of learning more complex patterns. In the same way, it is possible to analyze not only one variable, but several variables simultaneously to identify anomalies that may not be noticeable with a single variable.

There are numerous techniques and models that can be developed for anomaly detection, which provide good performance as well as satisfactory results. The methods for anomaly detection can be classified into the following techniques (see Figure 2.2):

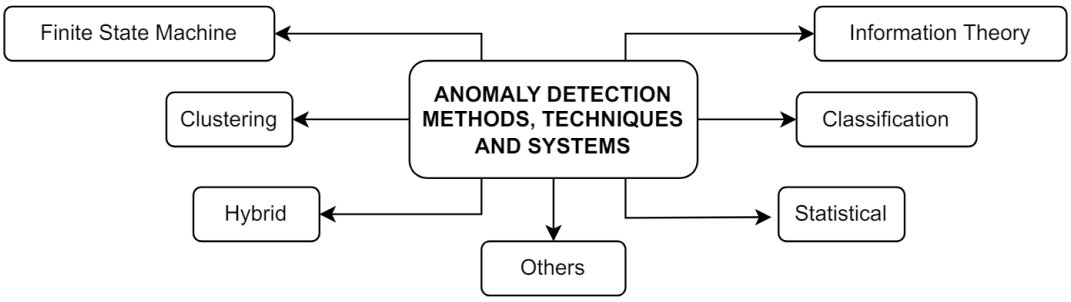


Figure 2.2: Anomaly detection methods.

- **Statistical methods:** are well known and widely used because they are based on probabilistic models [13].
  - **Wavelet analysis:** focuses on modeling non-stationary data series.
  - **Principal Component Analysis (PCA):** widely used because it allows a reduction of the dimensionality of the data, which makes it possible to separate traffic measurements into normal and anomalous subspaces.
  - **Covariance matrix:** has become a powerful method because it allows to find which variable best labels network anomalies and, therefore, to improve the detection of these anomalies.
  - **Thresholding:** this technique consists of classifying the anomalies in those data that deviate from a certain measure according to their mean and deviation.
- **Clustering methods:** this Machine Learning technique allows the detection of outliers by identifying those values that are not grouped within any cluster, i.e. those that deviate from the majority of the data. A well-known algorithm is K-nearest neighbor (KNN) clusters [14].
- **Finite State Machine:** mathematical models that model behavioral patterns, which are made up of states, transitions and actions, simulating the problem

[15]. They provide a solid analytical technique because they allow representing multiple situations by simulating inputs and outputs.

- **Classification-based methods:** technique that builds a classifier in the training phase to learn from labeled data. Followed by the inference phase in which this classifier is responsible for classifying new instances into normal data or anomalies.
  - **Naive Bayesian:** classifier based on probabilistic theory, which uses prior information to perform statistical inference [16].
  - **Support vector machines (SVM):** supervised learning technique based on the concept of feature vectors [17].
  - **Artificial Neural networks (ANN):** techniques that allow a minimum intervention of people due to their own capacity to look for patterns and optimization of the problems [18].
  - **Ensemble approach:** consists of the combination of classifiers which creates a new method that obtains better results compared to an individual approach.
- **Information theory:** technique based on the quantification of information and redundancy analysis, as well as the use of statistical properties.
  - **Entropy:** widely known measure that indicates the uncertainty of a problem [19]. For anomaly detection, entropy enables anomaly detection, because the higher the entropy, the more unpredictable the system and therefore the more likely it is to be an anomaly.
  - **Kullback-Leibler Distance or Divergence (KLD):** allows the difference between two probability distributions to be measured [20]. This technique is used to detect anomalies by analyzing two time periods and identifying possible anomalies.

2.3. Network anomalies

Network anomalies are considered to be all those situations in which the network traffic modifies its usual pattern through significant deviations, taking into account that network traffic inevitably undergoes slight variations due to the increase or decrease of users or diversity of services and applications in distinct temporal periods.

Network anomalies can occur due to two main causes. The first is physical, such as the failure of a node in the network structure or a drastic increase in streaming content, which causes an extremely rapid growth in the number of users and the consequent overloading of the network. The second possible cause is due to deliberate attacks that are intentionally malicious such as service intrusion.

To further understanding and analyzing the behavior of a network, it is necessary to measure a series of metrics that provide information on data flow, resource utilization and network performance, as well as network security [21]. The main measures for obtaining network information are described as follows (see Figure 2.3).

- **Bandwidth:** the bandwidth of a network is defined as the maximum transfer capacity of a network.
- **Throughput:** is the average rate in delivering a packet over a communication channel.
- **Jitter:** refers to the difference in packet delay (i.e. the difference between the arrival of the first packet and the next).
- **Latency:** latency is the time it takes for a packet to get from source to destination.
- **Congestion:** is the amount of traffic occupancy on an interface. For example, congestion at 100% occurs when a network carries or exchanges data at maximum its capacity. As a result, the network may experience delays in processing information or packet loss.
- **Packet-loss:** the information is splits into small packets to break it up for faster transmission. The packets are labeled with header information so that the information can be reconstructed when it reaches the target. When a significant portion of these packets is corrupted or lost in transit, packet loss occurs. Transmission Control Protocol (TCP) will recognize the loss and identify the lost packet to retransmit the information.

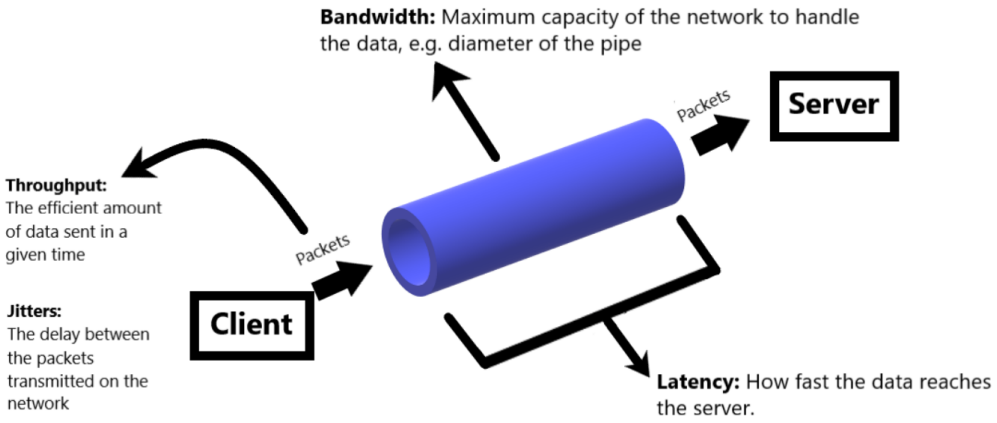


Figure 2.3: Network traffic metrics (extracted from [21]).

Knowing all the metrics mention above is crucial to the performance of any network. When people use programs or software, they want their requests to be received and responded to in a timely manner. Maximizing system efficiency is critical to ensuring user satisfaction and uninterrupted workflow

2.4. Open source datasets

The development of systems capable of detecting network anomalies heavily rely on the availability of high-quality datasets, which simulate real network traffic scenarios. In recent years, there has been an increase in the publications of open datasets due to the importance of research by the scientific community in this field [22].

Each service has a different use case to apply to, so this variety of scenarios and situations means that there is no specific dataset that serves a general purpose. Therefore, the datasets presented in the following sections have been used to analyze the state of the art of anomaly detection, realizing the particular necessity of creating a specific dataset for this master thesis due to the different features that compose the open datasets.

Within the datasets described in the following sections, there are datasets focused on intrusion anomaly detection, which is of vital importance in the communications industry, as well as datasets that are more related to the objective of this master thesis, focused on anomaly detection in online applications and streaming content.

Non-Linearly Separable Knowledge Discovery in Databases (NLS-KDD) [23] is one of the few open-source datasets well known in the world of anomaly detection, in particular, in intrusion anomaly field. It is made up of 41 features, which makes it the dataset with the highest number of parameters in the sector. Among the features are the number of failed logins, connection duration and connection protocol (see Figure 2.4). The analysis of the dataset shows that 82% of the protocols are tcp, followed by 12% udp and the remaining 7% icmp. It was also observed that it is a dataset with 53% of data labeled as no anomalies and 47% as anomalies, which is a good feature for Machine Learning techniques because it avoids class imbalance. Anomalies are labeled in the dataset based on detailed analysis of types of network attack by experts.

F#	Feature name	F#	Feature name	F#	Feature name
F1	Duration	F15	Su attempted	F29	Same srv rate
F2	Protocol type	F16	Num root	F30	Diff srv rate
F3	Service	F17	Num file creations	F31	Srv diff host rate
F4	Flag	F18	Num shells	F32	Dst host count
F5	Source bytes	F19	Num access files	F33	Dst host srv count
F6	Destination bytes	F20	Num outbound cmds	F34	Dst host same srv rate
F7	Land	F21	Is host login	F35	Dst host diff srv rate
F8	Wrong fragment	F22	Is guest login	F36	Dst host same src port rate
F9	Urgent	F23	Count	F37	Dst host srv diff host rate
F10	Hot	F24	Srv count	F38	Dst host serror rate
F11	Number failed logins	F25	Serror rate	F39	Dst host srv serror rate
F12	Logged in	F26	Srv serror rate	F40	Dst host rerror rate
F13	Num compromised	F27	Rerror rate	F41	Dst host srv rerror rate
F14	Root shell	F28	Srv rerror rate	F42	Class label

Figure 2.4: Features NLS-KDD (extracted from [24]).

The Numenta Anomaly Benchmark (NAB) [25] is another dataset containing data for the analysis of data detection systems in streaming or online applications. The data that make up this dataset come from different sectors such as financial, network and

cloud services, among others. In addition, the dataset stores the data with its temporal label, i.e., the evolution over time can be easily seen as shown in Figure 2.5. This open source dataset is composed by 58 files real world data each with 1,000 up to 20,000 instances which is designed to provide data for streaming anomaly detection research. The NAB dataset is manually labeled, following a documented procedure. Then, an algorithm uses the labels assigned by different human annotators to determine a final classification, using methods that weight the agreement between the different labels to generate the ground truth labels. Because of this manual labeling process in which the inclusion of real-world data with anomalies for which we know the causes, makes it one of the most accurate and useful datasets. However, it involves an extensive and extremely costly process to collect all the accurately labeled set of anomalous data instances.

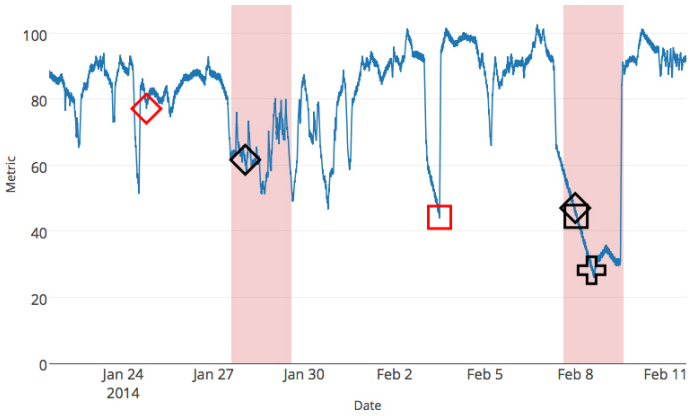


Figure 2.5: Example anomaly detection NAB (extracted from [26]).

2.5. Other softwares for anomaly detection

An analysis of the current approaches used for network anomaly detection is presented in this section. Description of a detailed review of a monitoring system widely used in current practice is presented, along with an innovative approach to network anomaly detection based on images.

SMARTxAC is a monitoring and analysis system that enables gigabit speeds without packet loss. In addition, it integrates a web-based graphical interface that provides numerous online traffic statistics. The current measurement scenario involves monitoring the Anella Científica network [27], which connects approximately fifty universities and research centers in Catalonia, since July 2003. SMARTxAC continuously monitors the link connecting the Anella Científica to RedIRIS, which constitutes its main connection to the Internet.

The SMARTxAC system consists of three main components: the capture system, the traffic analysis system and the results visualization system. Each of these components runs on a different computer for performance reasons. The capture system receives passive copies of traffic from Gigabit Ethernet links and initiates traffic flows.

In contrast, the traffic analysis system is responsible for aggregating the collected flows into classified flows, while calculating various traffic statistics. This classification allows obtaining detailed statistics per institution and destination network, as well as valuable information about the nature of the traffic (see Figure 2.6).

SMARTxAC uses an anomaly detection method based on traffic thresholds, where upper and lower traffic limits are established per institution. Anomalies are considered to be those situations in which an institution's traffic exceeds a predefined threshold.

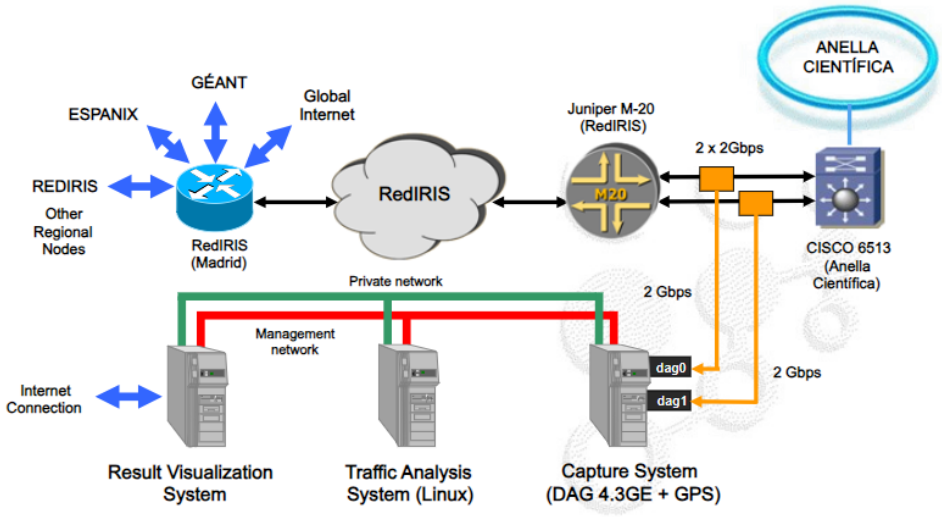


Figure 2.6: SMARTxAC platform overview (extracted from [28]).

In contrast to SMARTxAC, which uses a threshold based anomaly detection method, the following software is designed for image based detection. The network traffic anomaly detection approach using images is an innovative paradigm presented at a conference in Argentina [29]. This approach can provide advantages such as the ability to visualize anomalies in an intuitive way, as well as the extraction of invariant features from images.

The researchers describe the following steps for the implementation of this technology. First, data capture is performed using traditional form tools. Second, the network traffic data is converted into images, in which the color intensity of each pixel indicates a greater or lesser amount of data traffic. The third step consists of extracting features from the images and comparing these features with situations without anomalies and situations with anomalous patterns.

2.6. Open source tools

To perform network traffic monitoring and analysis, the following tools are useful to simulate a virtual environment, as well as to measure different network metrics on a specific port and interface. Network environments or emulators may be empty of services, so it is optimal to look for services to generate traffic on the network, either

through generic traffic, video streaming or web request. Taking this into account, this section describes different technologies that allow the monitoring of network traffic, as well as others that allow the construction of a simulated network environment.

2.6.1. Network measurement software

Internet Performance (Iperf) is an open source tool designed to measure the bandwidth between two hosts on a network. It allows to generate Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) traffic between nodes to evaluate network performance [30]. It works as a platform client-server, with versions available for Linux, macOS, Windows and even Android.

Users can use JPerf, a graphical interface to iPerf written in Java. It is worth mentioning that there exist Python wrappers of iperf [31], which are functions that allow encapsulating other functionalities to extend the behavior of iperf, facilitating the execution of testing. iPerf is useful for testing on Ethernet, Wi-Fi or Internet Service Provider (ISP) networks, providing accurate measurements of data transfer speed [32].

To find out or check the maximum achievable bandwidth of a computer, the necessary iperf commands are as follows. First of all, if the Linux (Ubuntu) computer does not have the necessary dependencies, it must be installed:

```
apt install iperf3
```

Second, one machine should be used as the server and the other machine as the client, with the following commands on each host:

```
iperf3 -s # Start server on host terminal
iperf3 -c <ip_device> # Send packages from host terminal 2
   to terminal 1
```

Once the two devices have been linked, a detailed report is obtained per terminal with data on the amount transferred in each time period, with the specific bitrate.

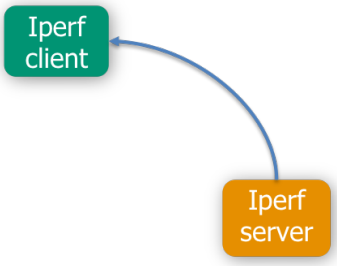


Figure 2.7: Iperf tool (extracted from [33]).

Networking Performance (netperf) is an open source tool that allows to measure several aspects of the network, among which stands out the ability to act as a one-way traffic generator, both UDP and TCP, as well as to measure the maximum bandwidth as iperf does [34]. The main advantage over iperf is that it can send request-response traffic and therefore evaluate the response of different devices. In addition to the aforementioned, it is also possible to modify the parameters to be returned by default to obtain new statistical data to characterize the network and its traffic.

Wireshark is a more advanced tool that allows to analyze network traffic with different protocols. This software supports physical layer protocols, link protocols, network protocols, transport layer and also application layer. Wireshark allows the capture of network traffic, storing all packets entering and leaving the network, storing this data offline and allowing the analysis of them at any time. At an educational level it is a very useful tool because it allows the detailed analysis of network traffic, as well as the connected devices.

2.6.2. Customized socket-based network probe

Within the field of open-source software, some developers create applications by integrating various techniques of the same nature. One of them is a development of a Python-based network probe, which uses socket communication for network performance measurement in real time [35].

This technology uses socket communication, which generates a connection between two machines. The probe is executed by command line interface (CLI), allowing to customize the execution environment. It is a very versatile probe because it allows the configuration of various parameters that allow setting exactly what metrics are desired, such as bandwidth, throughput, latency, packet loss rate, jitter, congestion and network interface statistics (see Figure 2.8).

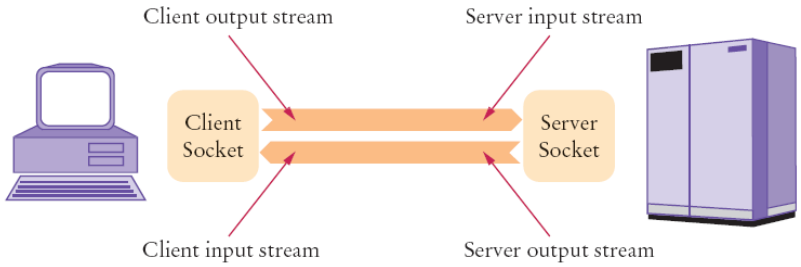


Figure 2.8: Socket communication (extracted from [36]).

As prerequisites to run the network probe, it is necessary to configure a web server like Apache or Nginx, because port 80 is the default port for HTTP traffic. To do this, the following steps must be followed. First, Apache2 must be installed using the apt package manager:

```
sudo apt update
sudo apt install apache2
```

The second step consists of turning on and enabling Apache:

```
sudo systemctl start apache2
sudo systemctl enable apache2
```

And finally, it is necessary to check the firewall to allow incoming HTTP traffic and verify with the following command that Apache is listening correctly on port 80:

```
sudo ufw allow 80/tcp
sudo ss -tuln | grep :80
```

Once the prerequisite is done, the next step is the installation of the software, which is done by cloning the GitHub repository to the local machine and installing the required dependencies:

```
git clone https://github.com/user/network-performance-probe
    .git
cd network-performance-probe
pip install -r requirements.txt
```

Once the software is installed, the probe is easy to use, measuring network performance, using the following command:

```
python network_probe.py --host <target_host> [options]
```

2.6.3. FFmpeg

FFmpeg is a command line tool used for video and image processing, which allows among other things to convert, encode and play different multimedia formats [37]. FFmpeg can be used directly from the CLI to perform various tasks related to the management, modification and analysis of multimedia files [38]. This open-source framework allows users to perform common and complex tasks without relying on third-party programs such as video editors. In addition, it offers similar functionalities to those of audio and video programs, but free of charge.

The workflow of sending content from one server to another by FFmpeg commands would be as follows. First, the video to be sent is captured on the source server.

Next, the video is encoded on H.264 using a format suitable for transmission, which may include compression of the video or conversion to an efficient format for transmission over the network. Once encoded, the video is sent over the network using a transmission protocol. The destination server receives the video stream and this server can simply store it or can retransmit it to other servers. Finally, the video is decoded at the target device.

The following command is an illustrative example to show the variety of commands and configurations provided by ffmpeg. The following command starts the ffmpeg program, the input file is specified, in this case a .mp4 video, the libx264 codec is used for video encoding, as well as the aac codec for audio encoding. The output format is specified to be compatible with the RTMP protocol and the url of the destination server where the data stream will be sent is defined.

```
ffmpeg -i input.mp4 -c:v libx264 -c:a aac -f flv rtmp://ip/
    app/stream
```

2.6.4. Network simulators

The complexity and scaling of networks is constantly increasing and therefore requires advanced testing and implementation methods. Network simulators and emulators become indispensable in this situation because it allows engineers to create and experiment with different network topologies in a controlled environment without the need of expensive hardware. These tools also offer the possibility to test new configurations, identify possible failures and analyze the performance of the network before deploying it in a real environment.

Cisco Packet Tracer is a network simulator developed by Cisco Systems [39], aimed primarily at students with various learning courses. Packet Tracer allows to create simple to complex network topologies using a drag-and-drop interface with multiple devices, such as routers or switches. It is widely used due to its comfortable user interface that allows learning the tool at high speed, as well as its support for Cisco devices. The main drawback is that being a simulator, it is not able to provide the same detail and accuracy as emulators would.

The next tool found is Graphic Network Simulation (GNS3) [40], which is a graphical network simulator that allows designing network topologies and simulating network scenarios. This tool allows, in the same way as the others, the creation of the network through a simple interface with multiple devices. GNS3 runs in a Windows environment and the interface itself allows to easily add the device images that want to be executed.

The Emulated Virtual Environment Next Generation (EVE-NG) is an emulator that has gained popularity over the last few years due to its numerous advantages [41]. It offers a full-featured network virtualization solution with many capabilities that allows users to create, configure and manage complex network topologies in a virtualized environment quickly and easily. In addition to all this, it is highly

accessibility that despite operating on Linux, by means of a virtual machine such as VMware, it is accessible to any operating system, which, with very few resources, offers a large number of possibilities. Among the key features provided by EVE-NG is a user interface that makes it easy for IT professionals to use the platform without the use of additional tools. It also supports emulation of a wide range of network devices and can even connect to real networks for more intensive testing.

Figure 2.9: EVE-NG simulator (extracted from [42]).

With the emergence of innovative Internet technologies, there are more and more services and devices with diverse characteristics that are interconnected through communication networks and require greater computing resources [43]. To address this problem, the cloud computing solution emerged, which relies on delegating the heaviest tasks to a central server with high computational power. At first glance, this approach solves the problem, but since these data centers are usually located far from the target devices, new problems arise, such as the increase in latency. To overcome this drawback, Multi-access Edge Computing (MEC) was developed [44]. It is similar to cloud computing, but which uses servers located closer to the users, i.e. at the edge of the access network, instead of remote servers. Due to the improvements it provides, several applications use the MEC architecture, such as robotics or virtual reality [43].

Network emulation software are fundamental tools in this context. They allow developers to test how their applications and services perform in different network scenarios prior to their actual deployment. By emulating various conditions, developers can optimize their applications to run efficiently on MEC architectures. By simulating a variety of network environments, emulators make it easy to identify potential problems, ensuring optimal performance in a MEC architecture, where low latency and proximity to the user are the main advantages. To provide a better understanding of the advantages of MEC architecture [45], a comparison between MEC and the main alternative, cloud computing, is presented in the following Table 2.1:

	Cloud computing	Multi-access Edge Computing
Latency	High	Low
Availability	High	High
Capacity	High	Medium
Flexibility	High	Very high
Security	Medium	High

Table 2.1: Comparison between Cloud computing and MEC

2.6.5. Streamlit

Streamlit is an open source Python framework that facilitates the creation of websites, in which to implement machine learning models with ease due to the simple commands without the need to delve into web development with HTML, CSS or JavaScript.

With just the following commands, the necessary dependencies to use the Streamlit tool in a Python environment are installed. The next steps consist of the creation of a customized environment, as well as the following desired application scenarios.

```
pip install streamlit
streamlit hello
```

The Streamlit tool provides a more user-friendly interface than a code executed by terminal, as well as the advantage of being able to deploy the AI models so that through a simple web address, any user can interact with the detection system. Due to the creation of an interactive data application with Streamlit, it is possible to containerize and deploy the system app on Google Kubernetes Engine, among others (see Figure 2.10).

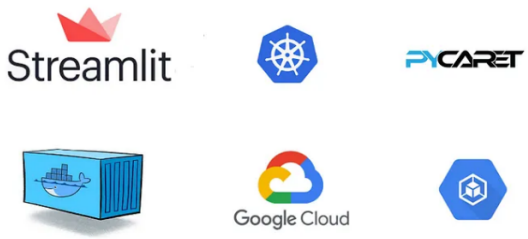


Figure 2.10: Streamlit deployment (extracted from [46].)

Chapter 3

Methodology

Once the investigation of existing software, algorithms and applications for the detection of anomalies in the network was completed, it was concluded that it was necessary to develop a suitable and optimal dataset for this master's thesis. Therefore, on a simulated network scenario, the dataset is developed using the sockets probe network, creating various conditions in the emulated network with the transmission of content in the form of videos, which simulate services consuming bandwidth.

3.1. Requirement definitions

The design and development of a network anomaly detection system requires a precise specification of the functional and non-functional requirements that must define its implementation. Functional requirements are statements about how the system should behave. It defines what the system must do to meet the needs or expectations of the user. Within these requirements, the following functional requirements are defined:

- The system must be able to collect real-time network traffic data using a network probe and subsequently clean and transform the collected data to provide a suitable input to the detection model.
- The system must allow the configuration of anomaly detection thresholds and label the dataset accordingly.
- The system must implement ML algorithms to detect anomalies in network traffic and perform model predictions.

On the other hand, non-functional requirements describe characteristics that the system must fulfill, such as performance, security or availability. The following are defined:

- The system must process large amounts of data quickly and efficiently using the Python programming language and its libraries, such as Pandas or NumPy.
- The system must be light and highly available, minimizing unproductive periods.
- The user interface should be intuitive and easy to use, allowing network administrators to interact with the system.

3.2. Project set up

For anomaly detection, a complete and structured workflow from beginning to end is essential. This process includes several stages, from project design to project implementation. The detailed procedure to be followed in the development of this master's thesis is illustrated in Figure 3.1.

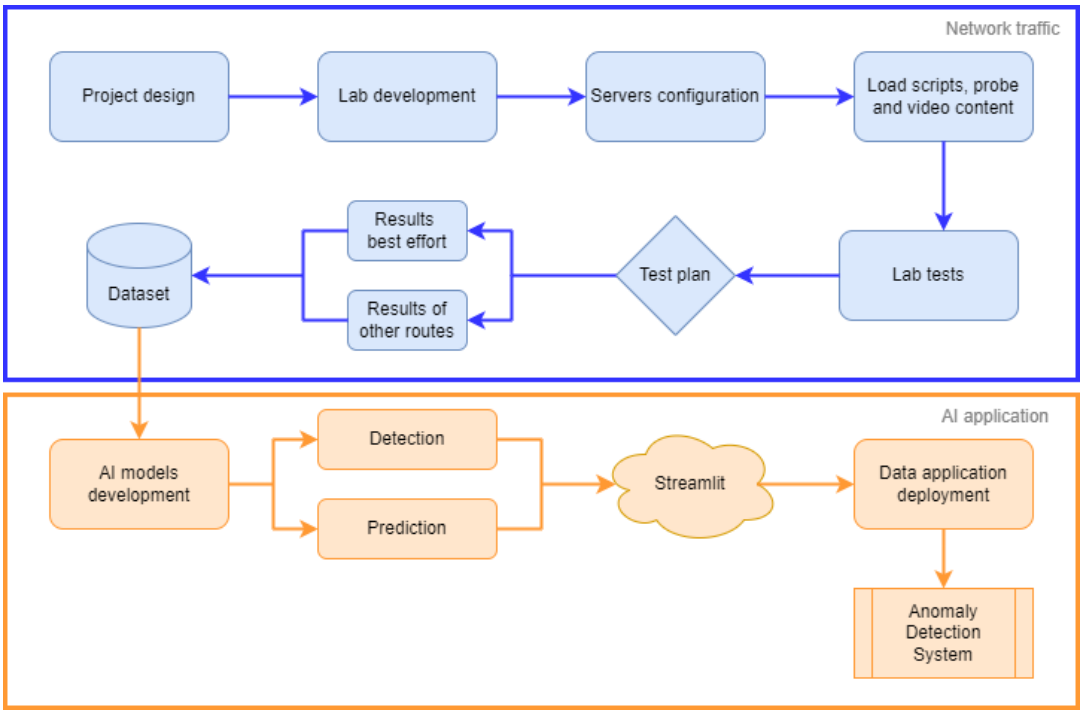


Figure 3.1: Project workflow.

It begins with the design of the project, which includes the research and investigation of software and tools necessary for the implementation of a system that allows the detection of anomalies. The next step consists of the development of a simulated scenario with the emulator tool that allows the creation of a laboratory for the generation of network traffic. EVE-NG is the emulator selected because of its capabilities. It allows one to create several network maps with different types of images, as well as the ability to modify the location of routers and to alter the links, either by pulling or creating them. Once the emulator has been developed, the next step is to configure the laboratory servers, to which all the necessary dependencies for traffic simulation must be installed, as well as the development of scripts in Python language to measure network traffic and the content to be transmitted, which in this case is multimedia content.

The next step is to determine a test plan to observe different scenarios in the network, both in base state and overloaded. After the traffic generation in the different

scenarios, data is collected from the two generated routes, the shortest available route, which takes the least time, called "best effort" and others routes that must redistribute the network traffic due to the lack or non-existence of routers. With all these data, the objective dataset of this master thesis is generated. With the dataset creation, the network traffic part is concluded, moving on to the second phase of the project, which focuses on the application of AI algorithms, specifically ML for the detection and subsequent prediction of anomalies.

In the second phase of the project, different ML models are developed to allow the detection of anomalies. Due to the three types of anomalies that exist, which are very diverse, techniques are developed to enable the detection of different anomaly types, such as global and contextual anomalies. With the models developed, an interactive web is developed using the Streamlit tool, which is a Python-based software that allows transforming data scripts into shareable web apps. Finally, with the development of the web, in which the developed AI models are mounted, the development of the anomaly detection system is accomplished.

3.3. Network traffic implementation

As previous steps to the development of this master thesis, it was decided to carry out local tests for an initial contact to learn how the probe and other network tools work.

As the first step of this master's project, the Oracle VM VirtualBox machine was installed together with the Linux operating system in a local environment. The purpose of this step is to create an environment in which to be able to generate network traffic in a simple way and to have a first contact with network anomalies.

Once the basic test simulation environment was created, the Python file of the probe was used to measure different metrics in the generated traffic. In order to obtain diverse metrics, traffic increases and decreases to characterize the scenario under various conditions. Among the commands used for the implementation of the probe, it is worth mentioning the following:

```
python network_probe.py --host 192.168.1.47 --port 80 --
    live --delay 15 --bandwidth --latency --congestion --
    packet-loss --throughput --jitter --verbose
```

Once the system setup of the probe in the emulated scenario is finished, several tests are performed to analyze the environment between the two deployed devices. With the previous command the probe is launched through the correct interface and starts to be displayed the metrics by terminal as shown in Figure 3.2. As can be seen from the terminal, numerous metrics are obtained in real time by increasing or decreasing the generated network traffic. In addition to the results of the metrics, the IP address of the target device is specified, as well as the port on which the measurements are being taken. It is worth to highlight that modifications were made

to the Python file to store the information displayed on the terminal in a .csv file for later analysis with ML techniques.

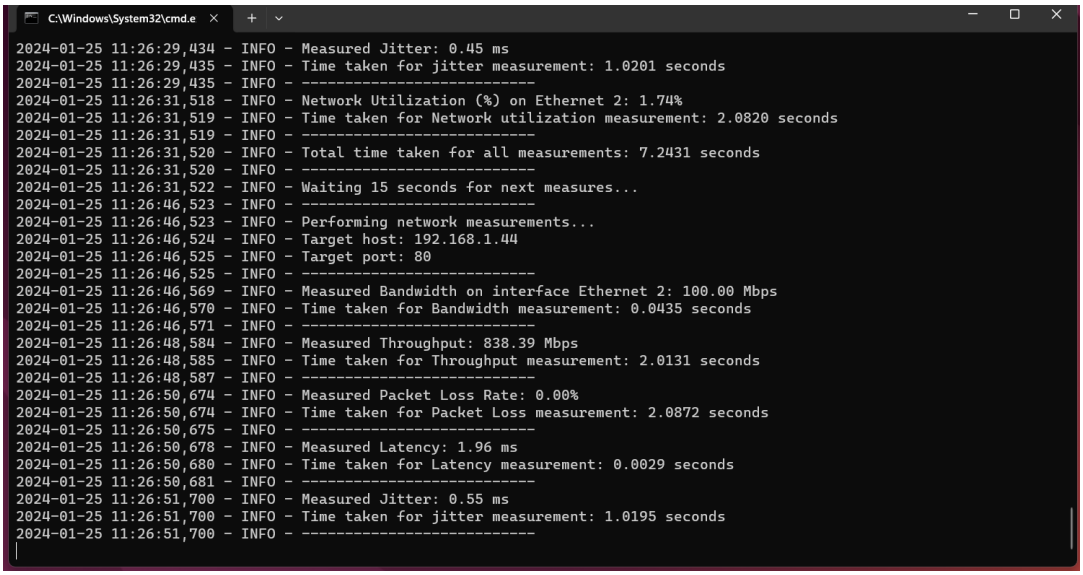


Figure 3.2: Sockets probe in virtual machine environment.

For the generation of network traffic in the emulated laboratory using the EVE-NG tool, the automated transmission of multimedia content from one server to another server is performed, with the launching of the probe simultaneously to obtain network data.

Before the transmission of multimedia content, iperf commands are launched for network categorization, in order to know the maximum bandwidth of the network, so that network traffic occupying different bandwidths can be simulated. As shown in Figure 3.3, one machine must be the server, listening on the indicated port, and another machine must be the client, as shown in Figure 3.4.

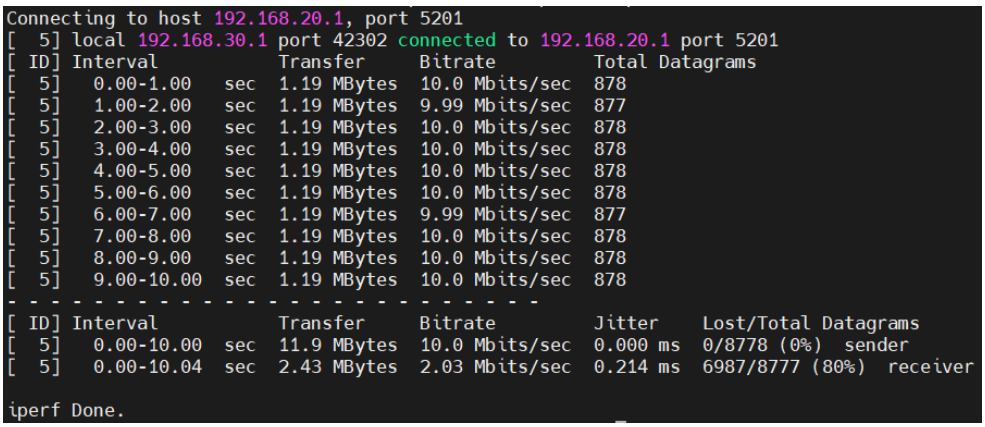


Figure 3.3: Iperf server.

```
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.30.1, port 57142
[  5] local 192.168.20.1 port 5201 connected to 192.168.30.1 port 49314
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-1.00   sec   313 KBytes  2.56 Mbits/sec  0.126 ms  611/836 (73%)
[  5]   1.00-2.00   sec   235 KBytes  1.93 Mbits/sec  0.154 ms  709/878 (81%)
[  5]   2.00-3.00   sec   236 KBytes  1.94 Mbits/sec  0.194 ms  708/878 (81%)
[  5]   3.00-4.00   sec   235 KBytes  1.93 Mbits/sec  0.182 ms  709/878 (81%)
[  5]   4.00-5.00   sec   236 KBytes  1.94 Mbits/sec  0.204 ms  708/878 (81%)
[  5]   5.00-6.00   sec   235 KBytes  1.93 Mbits/sec  0.183 ms  708/877 (81%)
[  5]   6.00-7.00   sec   241 KBytes  1.97 Mbits/sec  0.140 ms  0/173 (0%)
[  5]   7.00-8.00   sec   248 KBytes  2.03 Mbits/sec  0.161 ms  709/887 (80%)
[  5]   8.00-9.00   sec   248 KBytes  2.03 Mbits/sec  0.143 ms  708/886 (80%)
[  5]   9.00-10.00  sec   248 KBytes  2.03 Mbits/sec  0.186 ms  709/887 (80%)
[  5]  10.00-10.05  sec  15.3 KBytes  2.75 Mbits/sec  0.193 ms  708/719 (98%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-10.05  sec  2.43 MBytes  2.03 Mbits/sec  0.193 ms  6987/8777 (80%)  receiver
```

Figure 3.4: Iperf client.

It is observed that the maximum bandwidth is 2 Mbps, which indicates that network traffic below and above this bandwidth value should be simulated to observe the network behavior in different scenarios. The low bandwidth value is due to the fact that the software images loaded in the EVE-NG emulator are control plane, not data plane, which are used to perform infrastructure management. Once the network is categorized, a video is searched for subsequent transmission in the laboratory scenario simulated at different bitrates. It is worth mentioning that the low bandwidth value is due to the fact that the simulated scenario is of the control plane type, which is designed to perform infrastructure management operations.

The chosen video has a size of 1MB with a total duration of 28 seconds, therefore to categorize the network the following ffmpeg command is used to obtain the different videos with the specific bitrate for the network congestion at different values (see Table 3.1):

```
ffmpeg -i video8Mb.mp4 -b:v 167k video4Mb.mp4
```

Video (Mb)	Bitrate (kbps)	Network capacity rate
Video4Mb	167	10%
Video8Mb	334	20%
Video20Mb	835	50%
Video32Mb	1336	80%
Video40Mb	1670	100%
Video48Mb	2004	120%

Table 3.1: Multimedia content to perform network traffic.

3.4. Lab development

It is essential to have laboratory environments that allow experimentation and testing of proposed solutions. Considering that we are capable of generating and analyzing network traffic data, as well as detecting network anomalies through ML techniques, the need to develop a laboratory environment that reflects the network's underlying complexities and challenges arises. Therefore, the study and development of various network topologies for their subsequent implementation in an emulated laboratory environment provided by Telefónica was started.

First of all, the network topology of the Figure 3.5 is designed, which is a simple topology, the main requirement to be able to categorize the network. The network is formed by 3 servers, which are the traffic generators. This traffic generation is done by launching videos at certain times. The other components of the network are XRv9k routers, which are cloud-based Cisco routers that are deployed in simulators [47]. This Internetwork Operating System (IOS) XRv 9000 router [48] provides traditional edge services from the provider in a virtualized design as is the case in this master thesis.

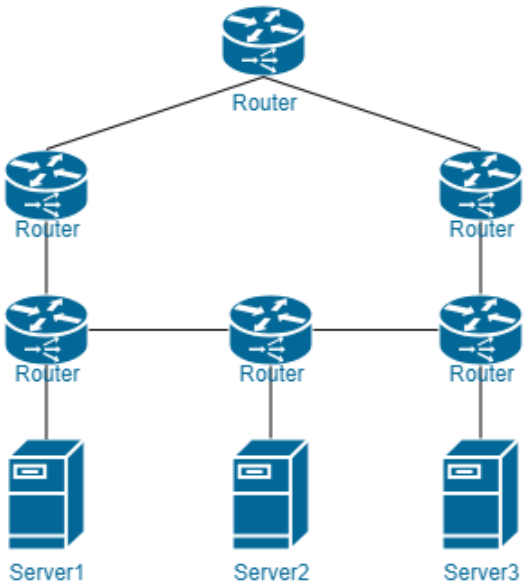


Figure 3.5: Network topology.

The lab developed by Telefónica is based on the concept of MEC architecture, which moves the computing of traffic and services from a centralized cloud to the edge of the network, near to the user. Telefónica operates as a MEC, being the service provider of the simulated lab on its servers, which are accessed via ssh. This structure reduces latency and ensures highly efficient network operation.

The emulated network scenario is composed of three servers, each of which is capable of generating network traffic, as well as the six routers that will allow the creation of various scenarios, by creation or deletion of links between them. As a summary, details are shown in Table 3.2.

Nº nodes	Nº servers	Nº routers
9	3	6

Table 3.2: Summary of network topology.

3.5. Requirement installation

For the correct generation of network traffic and transmission of multimedia content, the simulated laboratory servers must have all the necessary dependencies installed. First of all, conda is installed, which is a system that allows to manage Python environments in a simple way, allowing the installation of different packages in each one of the environments, without installing dependencies and libraries on the host machine. To install conda, it is necessary to install Anaconda first [49]:

```
sudo apt-get update
cd /tmp
apt-get install wget
wget https://repo.anaconda.com/archive/
Anaconda3-2022.05-Linux-x86_64.sh
bash Anaconda3-2022.05-Linux-x86_64.sh
```

Once Anaconda is installed, the next step is to activate the environment configuration using the following command:

```
source ~/.bashrc
```

Once it has been verified that conda is ready, an environment is created on each of the three servers using the following command:

```
conda create --name
```

After that, the probe is installed and the files contained in the probe are modified to save the data of the metrics generated in a .csv file, which will be necessary for the development of algorithms using ML techniques. For the operation of the probe, the interface to be measured must be specified. To achieve this objective, the following

interfaces of the emulated scenario servers are analyzed by executing the command ifconfig through the CLI (see Figure 3.6). The presence of the broadcast address 192.168.159.255 in the output indicates that the interface *ens3* is enabled to send broadcast messages to all devices on the same subnet. Therefore, it is noted that the probe should be measured at the interface *ens3* to collect the network traffic in the scenario.

```
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.159.243  netmask 255.255.255.0  broadcast 192.168.159.255
        inet6 fe80::f816:3eff:fede:53fe  prefixlen 64  scopeid 0x20<link>
        ether fa:16:3e:de:53:fe  txqueuelen 1000  (Ethernet)
        RX packets 7032170  bytes 1333215735 (1.3 GB)
        RX errors 0  dropped 1559  overruns 0  frame 0
        TX packets 923833  bytes 575265376 (575.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 3.6: Network interface.

3.6. Test plan and dataset development

To study the behavior of the simulated network, a test plan is created with different modifications of the scenario in order to have as much data as possible to be able to categorize the different possible events. The scenario modifications vary from router removal, transmission of content from different servers, as well as forcing network stress. The launching of the videos on the servers is done in a simultaneous or random way at different times to categorize the network and to simulate uncontrolled traffic.

Scenario	Planned route	Measure
Base scenario	Best effort	Server1 → Server2
Base scenario	Best effort	Server1 → Server3
Video transmission	Best effort	Server1 → Server2
Video transmission	Best effort	Server1 → Server3
Simultaneous video transmission	Best effort	Server1 → Server2
Simultaneous video transmission	Best effort	Server1 → Server3
Base scenario	Redistribution	Server1 → Server3
Video transmission	Redistribution	Server1 → Server3
Simultaneous video transmission	Redistribution	Server1 → Server3
Increase/decrease transmission	Redistribution	Server1 → Server3

Table 3.3: Test plan.

As shown in Table 3.3, the scenarios can be classified into two main groups, those that send network traffic via the best effort route, which is the shortest route, i.e. the route with the least number of hops between routers.

The other group of scenarios, on the other hand, is designed to send network traffic over the longest route based on routing algorithm, having turned off the links between some routers.

This network modification allows us to better understand how the network behaves in the presence of various alterations. In order to perform all these scenarios, the following commands must be used through the Teletype Network (Telnet) protocol on the routers to be modified, either to establish or drop links:

```
telnet <ip> <port>
up:     configure
        interface gigabitEthernet 0/0/0/1
        no shutdown
        commit
down:   configure
        interface gigabitEthernet 0/0/0/1
        shutdown
        commit
```

Once the different scenarios are defined, the specific network configuration of each scenario and the network traffic is generated, being this the different videos with each specific bitrate. The dataset is generated in a .csv format with the following features, which contain information about the time of each data, the network metrics, as well as the configuration of the devices in the network and the traffic generated between them:

Feauture	**Description**
Timestamp	Date and time of data acquisition
Bandwidth	Network metric measured in Mbps
Throughput	Network metric measured in Mbps
Congestion	Network metric measured in percentage
Packet loss	Network metric measured in percentage
Latency	Network metric measured in ms
Jitter	Network metric measured in ms
Routers	Linked or connected routers
Planned route	Path followed by traffic
Network measure	Server where the probe is launched
Network target	Server where the probe is measuring
Video target	Server where the multimedia content is sent
Number of videos	Number of simultaneous videos
Percentage video occupancy	Utilized capacity of bandwidth
Bitrate	Number of bits per second sent per video

Table 3.4: Dataset description

3.7. AI models

ML allows the analysis of data due to its properties such as flexibility, speed, efficiency and efficient handling of large and complex volumes of data. The objective is to develop a model which can detect possible anomalies based on data received from the network. With this input, the model must infer whether or not there is an anomaly. In addition, with ML, if there is a certain trend in the data, the model could identify the anomaly in advance and detect it ahead of time.

The different ML techniques that will be applied to the data obtained with Telefónica's simulator are listed below (see Figure 3.7).

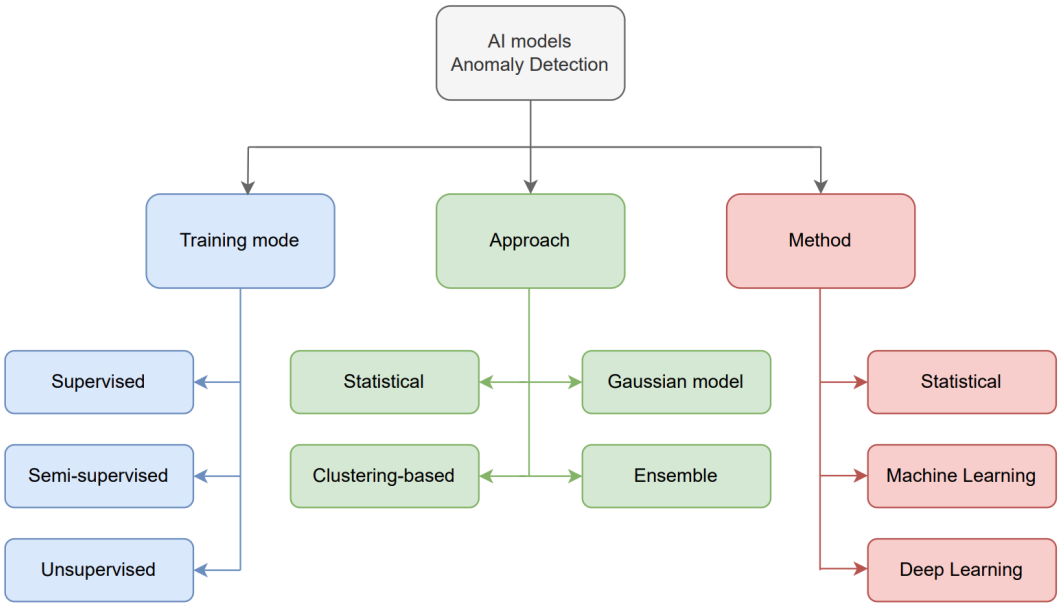


Figure 3.7: AI models development.

- **Supervised learning**: models are trained using labeled data, i.e., data for which we know predefined responses. To apply this type of ML, the data is first labeled as anomalies using the previously described statistical techniques.
- **Semi-supervised learning**: is a hybrid technique that uses both labeled and unlabeled data. This technique is used in situations where obtaining a sufficient amount of labeled data is either difficult or expensive, while large unlabeled data is simple to collect.
- **Unsupervised learning**: models are trained using unlabeled data, i.e. we do not know the tags or preset categories. Instead, the model looks for patterns, structures or groupings inherent in the data without any external guidance. This approach allows us to discover unknown patterns within the data and identify groups, which could be anomalies or not.

Using the aforementioned techniques, an anomaly detection system will be developed to characterize the network in order to know when an anomaly occurs. Furthermore, by means of ML and time series techniques, the model could be able to detect these anomalies ahead of time due to the previous data information.

It is important to mention that in certain situations, it is important to apply dimensionality reduction by means of principal component analysis (PCA) [50], which is a technique that allows to project the data in a lower dimensional space (2D or 3D), allowing discovering patterns in large datasets or with a large volume of features. PCA only finds linear relationships in the data, whereas t-Distributed Stochastic Neighbor Embedding (t-SNE) [51] is another technique that also reduces the dimensionality of the data but is capable of analyzing complex polynomial relationships in addition to linear ones. In addition, in order to understand the outcome of the predictions made by the models once trained, it can be very useful to use SHapley Additive exPlanations (SHAP) [52], which is a method that allows to increase the explainability of ML models. As a result of this technique, the features of the dataset that have the greatest influence on the outcome of the model, are known, which allows to better understand why some predictions are made and not others.

Once the ML models have been evaluated, they are usually deployed in frameworks such as Google Cloud or Amazon Web Services, but for ease and rapid deployment, Streamlit is used to convert the Python code into a data interactive application. For this master thesis, Streamlit will be used so that once the models are developed, the user enters new network traffic data and is provided with results in an interactive and fast way. Depending on the type of data to be entered, the application will display the results to the user, either by text or in a more visual way with graphs and analytics.

Chapter 4

Results and Discussion

This chapter presents and explains the results obtained according to the objectives of this master thesis.

4.1. Simulated lab environment

To implement the anomaly detection system, the first step was the creation of the simulated network scenario illustrated in Figure 4.1.

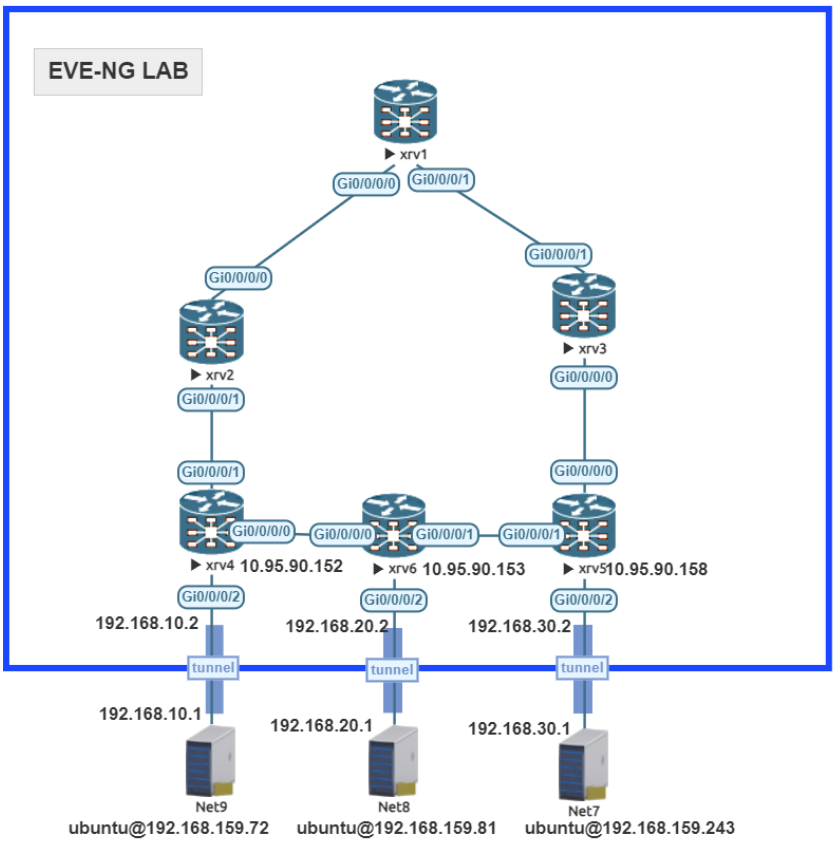


Figure 4.1: Network environment.

The network topology consists of three servers, which are connected to the network environment by means of a GRE (Generic Routing Encapsulation) tunnel. The tunnel makes it possible to redirect traffic from the virtual machines, in this case the servers, to the laboratory network configured in EVE-NG. This encapsulation protocol allows wrapping a packet with a new header, so that it can be transmitted over a network incompatible with the original protocol.

The different routers have been placed in this topology to allow network traffic to be sent from different routes, from the Best effort route, which has the least router hopping, to the Redistribution route, which involves the longest path. In Table 4.1 the name of the servers to be used in the following sections are specified together with the IP address of the three corresponding virtual machines.

Server	
S1	192.168.159.72
S2	192.168.159.81
S3	192.168.159.243

Table 4.1: Servers IP.

To create the dataset used for the development of the anomaly detection system, the following scenarios, specified in Table 4.2, were followed. It is worth mentioning that the scenarios of the test plan can be divided into two main scenes, the one in which the multimedia content traffic is sent through the Best effort route, which involves the least number of hops due to the fact that the interfaces of the xrv6 router link, the router that enables the server 2 to communicate with the other elements of the network, are turned on. The second network scenario is designed to send network traffic through routers xrv1, xrv2 and xrv3, by turning off the link interface of router xrv6. With these two paths, as specified in the Test plan, the network traffic measurements are performed on various servers, as well as the launch of multimedia content from several nodes.

	Routers	**Planned route**	**Measure**	**Video**
1. Base scenario	up xrv6	Best effort	S1 -> S2	None
2. Base scenario	up xrv6	Best effort	S1 -> S3	None
3. Video transmission	up xrv6	Best effort	S1 -> S2	S1
4. Video transmission	up xrv6	Best effort	S1 -> S3	S1
5. Simultaneous video transmission	up xrv6	Best effort	S1 -> S2	S1, S3
6. Simultaneous video transmission	up xrv6	Best effort	S1 -> S3	S1, S2
7. Base scenario	up xrv1,2,3	Redistribution	S1 -> S3	None
8. Video transmission	up xrv1,2,3	Redistribution	S1 -> S3	S1
9. Simultaneous video transmission	up xrv1,2,3	Redistribution	S1 -> S3	S1, S2
10. Increase/decrease transmission	up xrv1,2,3	Redistribution	S1 -> S3	S1, S2

Table 4.2: Test plan.

4.2. Exploratory data analysis

Prior to the implementation of any ML model that allows to develop the network anomaly detection system, it is necessary to perform an Exploratory Data Analysis (EDA), which is a set of techniques that allow to analyze and explore datasets in order to understand their main properties, by using data visualization methods. With the use of EDA, the best way to manipulate the data to obtain the optimal answers to the problem is identified. The main objective of the EDA is the analysis of the data before assuming anything about the data. This allows us to identify difficult errors, as well as to better understand patterns within the data and find interesting relationships between any variables.

The dataset used for this master thesis is obtained by performing the described test plan in Table 4.2. From each scenario, measurements of the generated network traffic are collected at an interval of 15 seconds, until a total of 100 measurements are obtained. As a result, a dataset with 1,000 rows is generated, each one of them with metrics from the network probe, as well as information about the network status and the generated multimedia content.

Table 4.3 shows the description of the variables in the generated dataset. It can be noticed that the dataset is made up of a total of 1,000 rows and 15 columns. The table 4.3 shows the summary of the rows with numerical features. It is worth mentioning that the bandwidth variable always has the value of 2 Mbps since it is the maximum bandwidth of the emulated environment. Regarding throughput, the average value is 1.91 Mbps, with a minimum value of 0.90 Mbps reaching up to 7.14 Mbps in some of the test plan scenarios. The congestion metric ranges from 0.03% in situations in which no multimedia content traffic is sent over the network to maximum values of 134.37% at times when the network is overloaded due to heavy network traffic. As for packet loss, an average of 5.43% loss is observed, reaching values of approximately 50% loss, which indicates that something is not performing as expected. It should be noted that the packet loss values in most situations are close to 0% because data below the third quantile (75%) do not experience packet loss. The same analysis is performed for the subsequent variables, showing that latency ranges from 4.48 ms to highly anomalous values such as 3,051 ms, which is a strange behavior in some of the scenarios.

	Throughput	Congestion	Packet loss	Latency	Jitter	Occupancy	Bitrate	Videos
count	1000	1000	1000	1000	1000	1000	1000	1000
mean	1.91	23.86	5.43	54.75	0.86	28.98	483.98	1.18
std	0.90	32.14	9.65	275.62	0.89	37.07	619.15	1.11
min	0.05	0.03	0	4.48	0	0	0	0
25%	1.43	0.09	0	6	0.48	0	0	0
50%	1.98	10.74	0	7.55	0.65	10	167	1
75%	2.46	40.89	7.50	10.29	0.94	50	835	2
max	7.14	134.37	52.5	3,051.58	10	120	2,004	6

Table 4.3: Summary dataset

Continuing with the analysis, for jitter, it is observed that it is the value with the lowest standard deviation, which indicates that it is the metric that suffers less variations throughout the test plan scenarios. The last columns of Table 4.3 describe that the network percentage overhead has been varied from 0, 10, 20, 50, 50, 80 and 100 to 120%. This has been done by sending videos at different bitrates. As can be seen in the last column of the table, the number of videos sent to generate network traffic through multimedia content varies from 0 to 6 simultaneous videos, depending on each scenario of the test plan.

Moving forward in the analysis of the dataset, it is convenient to see how the characteristics behave among them. For instance, the correlation matrix in Figure 4.2, which provides information on how closely the variables are related to each other, allowing to find patterns between them. A value greater than 0 indicates that there is a positive correlation between two variables, with the closer this value is to 1, the stronger the association. On the other hand, a value less than 0 indicates a negative correlation, with the closer this value is to the value -1 end, the stronger the negative correlation. The correlations between the different features of the dataset of this master thesis are detailed below, as well as the significance of these correlations.

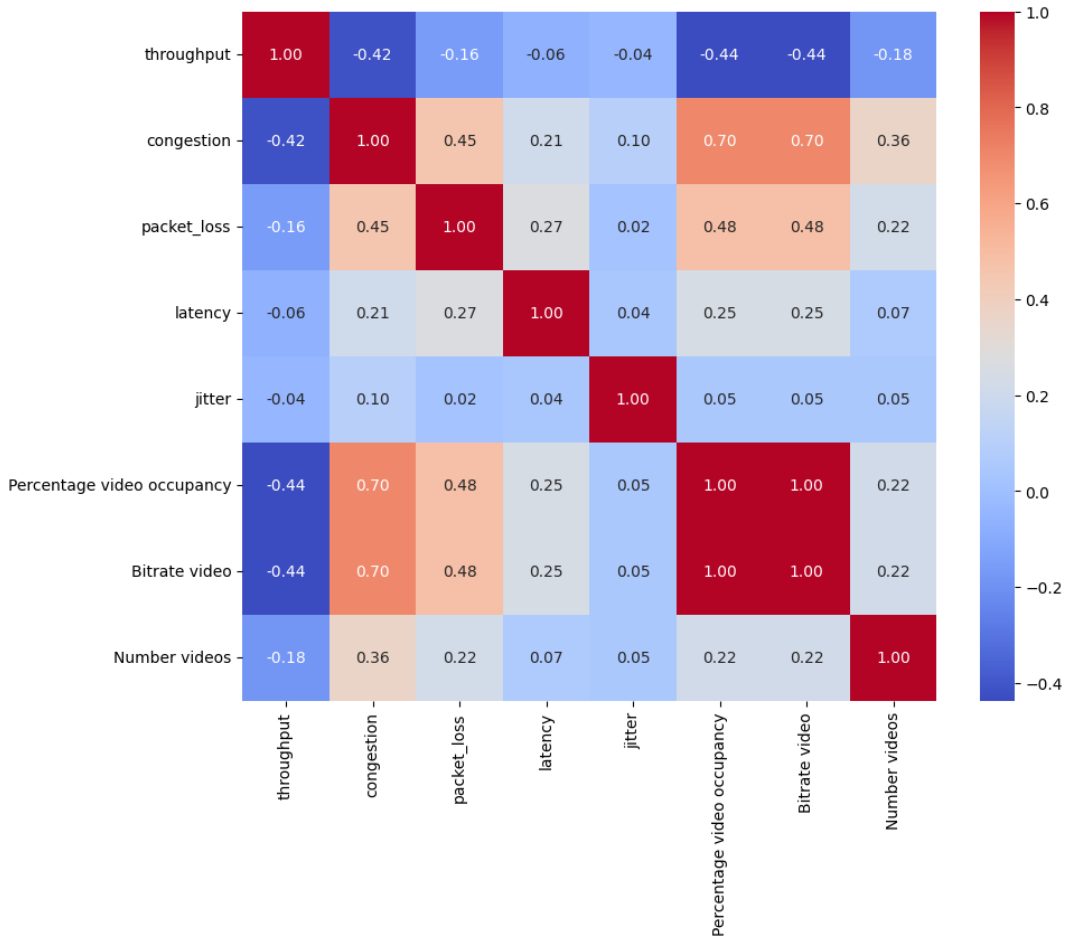
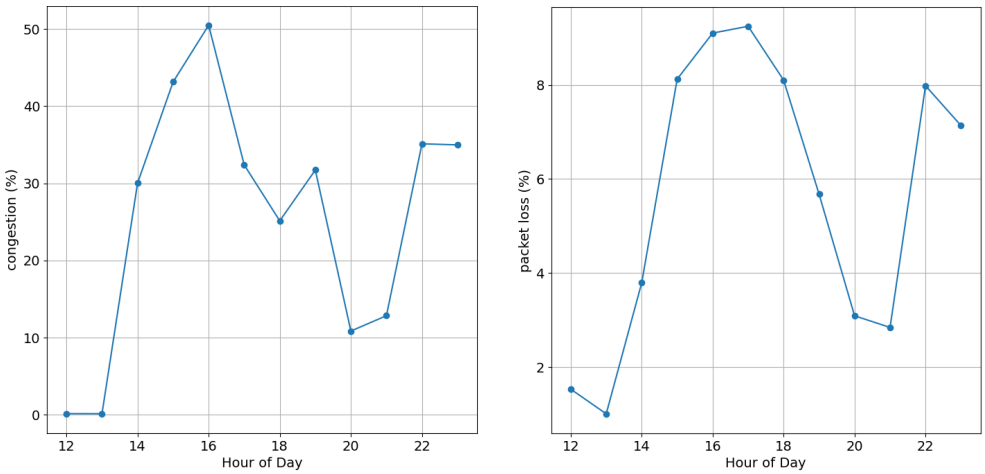


Figure 4.2: Correlation matrix.

Analyzing the correlation matrix, it can be noticed that the higher the throughput, the lower the congestion and packet loss, as well as the lower the bitrate of the multimedia content, which generates a lower network occupancy percentage. Analyzing the correlations of the congestion variable, it is observed that the higher the congestion, the lower the throughput and the higher the other variables except from jitter. It is worth mentioning that the jitter variable is not correlated with any other column of the dataset because the value is very close to zero, which indicates that jitter is not correlated. This may be because fluctuations in network conditions can affect jitter without influencing other metrics.

To examine in greater detail the relationships revealed by the correlation matrix, the following graphs are obtained, showing the average of the variables over the test plan implementation time, in which data varies every hour, since a different scenario is carried out every hour. Figure 4.3 shows the evolution of congestion and packet loss in the different scenarios. It can be seen that the two variables change their behavior in a similar way, i.e., when congestion increases, packet loss also increases, which is consistent with the positive correlation shown by these two variables in the correlation matrix. The lowest data were observed from 14.00 to 16.00, which correspond to the base scenarios of the first route, and from 20.00 to 21.00, corresponding to the base scenario of the second route, in which no network traffic was generated.



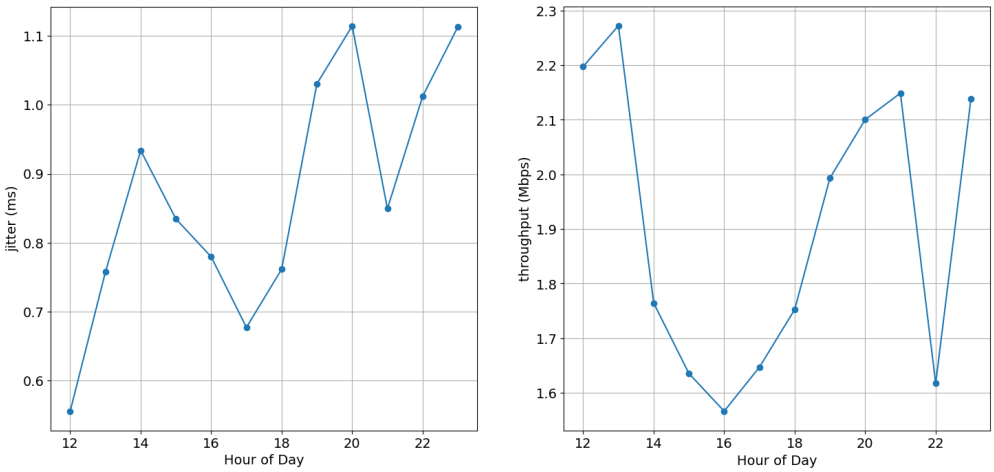
(a) Mean congestion.

(b) Mean packet loss.

Figure 4.3: Evolution of congestion and packet loss values.

The next graph to be analyzed corresponds to the jitter parameter, which was found with the correlation matrix not to be related to any other variable, with the exception of a slight correlation with congestion of 0.10. The jitter follows a different pattern from the rest of the metrics, observing this evolution in the Figure 4.4. Specifically, 3 escalating rises are observed, which are increasing as the hours progress. The first one is observed from 12.00 to 17.00, the second from 18.00 to 21.00 and the third from 21.00 to 23.00.

Based on theoretical information, fluctuating latency motivates an increase on jitter [53]. However, as observed in this study, jitter does not always increase in the same manner.



(a) Mean jitter. (b) Mean throughput.

Figure 4.4: Evolution of jitter and throughput values.

The Figure 4.5 shows an analysis of the data for each of the 10 scenarios of the test plan. It shows the variation of the throughput values, noticing that the lower the network traffic (as in the cases of base scenarios 1, 2 and 7), the throughput reaches its highest values. The probe fills the channel with traffic, so if there is secondary traffic, it will not be able to fill the capacity as much as it could if there were no other services.

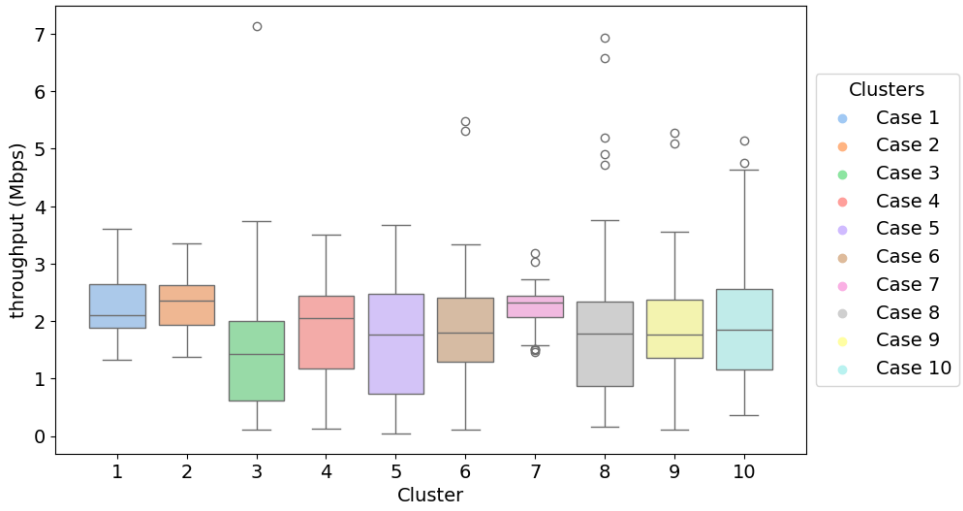


Figure 4.5: Boxplots throughput scenarios.

It can also be seen that the throughput is not conditioned by the type of route, since both cases 1 to 6, which correspond to the best effort route, and cases 7 to 10, which follow the extended route, maintain similar throughput values.

Figure 4.6 shows the different latency values in the different simulated scenarios. Extracting information from the figure, we estimate that each router introduces a latency of 2 ms, having for the best effort route, a latency between 6 and 7 ms. On the other hand, for the second path, network traffic has a latency between 10 and 12 ms. Similarly, depending on the server on which the traffic was generated, a latency 2 ms higher or lower is observed, which is consistent as a router is introduced or removed.

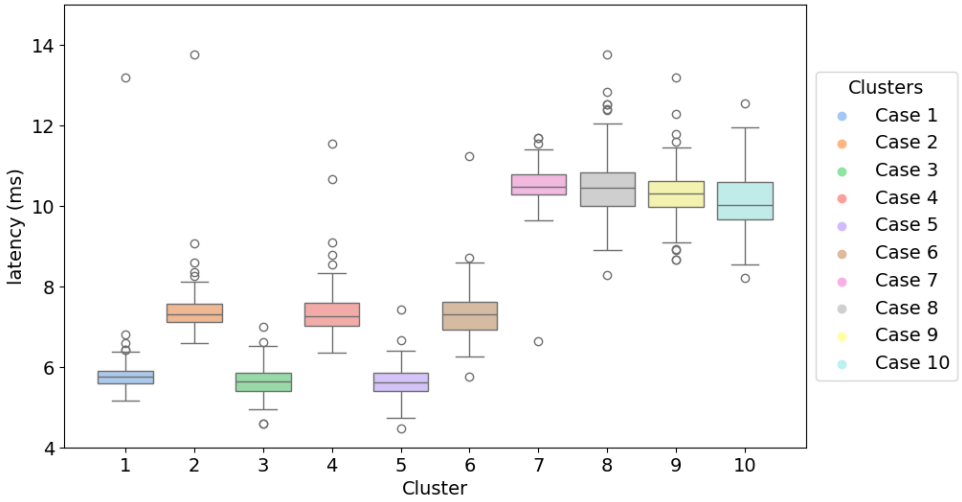


Figure 4.6: Boxplots latency scenarios.

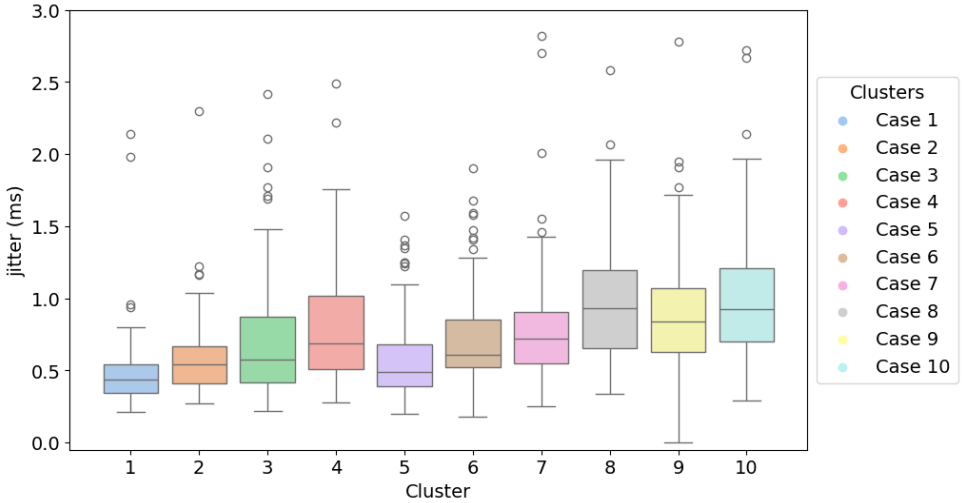


Figure 4.7: Boxplots jitter scenarios.

Figure 4.7 shows the behavior of the jitter variable in the different 10 scenarios of the test plan. As discussed in the previous sections, the correlation matrix shows that jitter has no correlation with any of the other variables. Similarly, the mean jitter plot showed three significant increases at different times. As seen with the boxplots, the jitter increases significantly in scenario 4 and subsequently, in scenario 8, in the same way as the rises in the mean jitter plot. With Figure 4.7 it is observed that the increase in jitter occurs in scenarios with a higher number of hops, concluding that the more nodes on the route, the greater the probability of occurrence of fluctuations in the delay of data transmission, increasing the value of jitter. Each node that a data packet passes through introduces the possibility of additional delays.

4.3. Dimensionality reduction

The number of characteristics of the dataset is high, so in order to make a 2D representation, it is convenient to use dimensionality reduction techniques such as the t-Distributed Stochastic Neighbor Embedding (t-SNE). This technique allows to reduce the dimension of the data in order to analyze the distribution of the data and relationships between them. By visualizing the data, anomalies such as those data points that are out of aggregations can be identified. Therefore, before applying ML algorithms, t-SNE can be useful to explore and better understand the structure of the data.

Figure 4.8 shows the data grouped into different, quite well separated groups. The different groups are assembled in terms of similar patterns. In this case, it can be seen that the data with the highest throughput values are in the groups in the lower left area of the graph, while those with lower throughput values are grouped in the upper area of the graph.

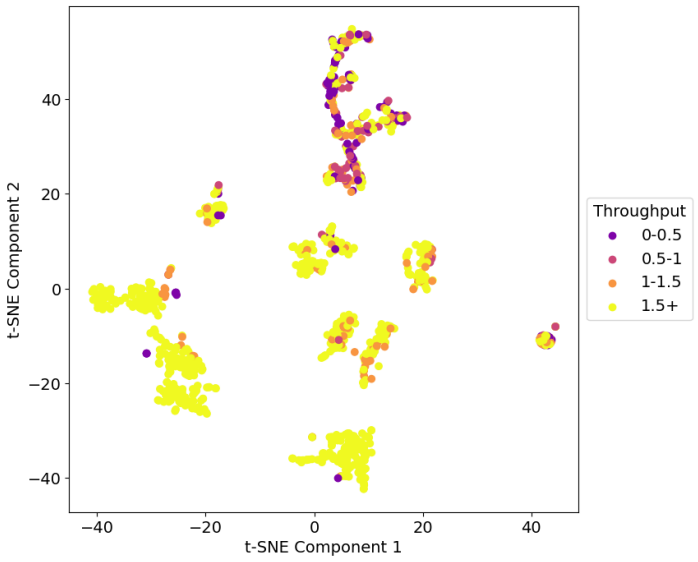
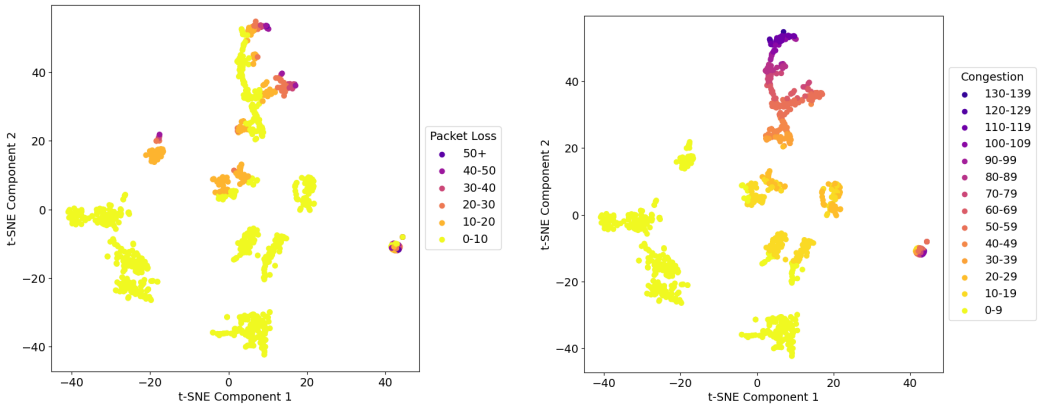


Figure 4.8: t-SNE throughput.

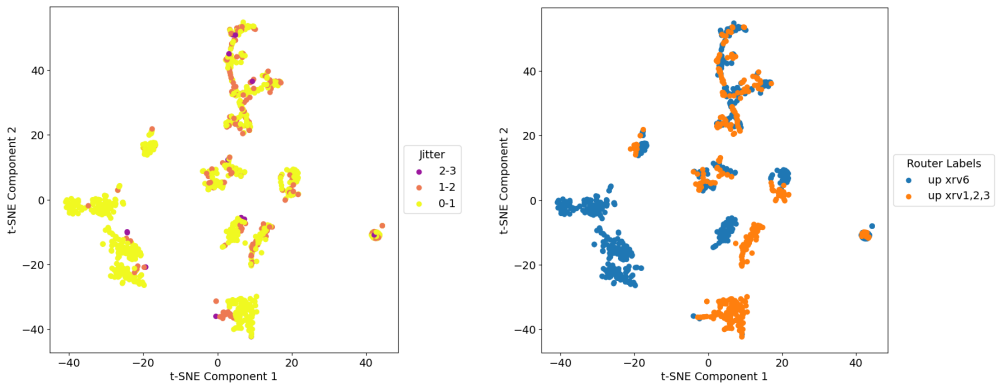
Figure 4.9(a) shows the visualization of the data in the reduced dimension by t-SNE for the packet loss variable. As expected, since the values with the lowest throughput in the previous graph were located in the upper area, the values with the highest packet loss are also located in this area, leaving the packet loss values below 30% in the lower left area.



(a) t-SNE packet loss. (b) t-SNE congestion.

Figure 4.9: t-SNE packet loss and congestion values.

Figure 4.9(b) displays data with high network congestion values in the upper area of the graph. This is also consistent with the low throughput values in this area, as well as high packet loss.



(a) t-SNE jitter. (b) t-SNE different routes.

Figure 4.10: t-SNE jitter compared to the routes values.

Finally, the last characteristic to observe can be seen in Figure 4.10(a), measuring the jitter. The distribution of high jitter values does not follow the same pattern as the previous plots, because this variable has no correlation with any variable in the

dataset. The conclusion obtained with the boxplots is that jitter tends to increase with a higher number of nodes. To perform this check, Figure 4.10(b) is obtained, which displays the grouping of the data according to the two routes of the dataset, the best effort route in blue and the longest route in orange. As can be seen, the jitter values are generally higher in the areas of the route with a higher number of routers in the route, which is consistent with the previous argument.

4.4. Anomaly detection

An anomaly is one or more observations that are outside a data set. In the scientific community, there is no formal definition among scientists as to what an anomaly is. Because of this, in the corporate environment, due to the multiple situations and problems that a company deals with, each organization specifies particular thresholds for the detection of anomalies.

There are many methods for defining anomalies. The best known are the statistical methods, among which is a simple technique for the identification of anomalies in one-dimensional data following a normal distribution, based on the mean and standard deviation [54]. This method labels as anomalies the points that deviate from the mean plus 3 standard deviations. However, because the dataset of this master thesis does not follow or resemble a normal distribution, this simple method is discarded.

For the detection of global outliers, it is decided to use nonparametric techniques. The use of nonparametric techniques is due to the fact that they make minimal assumptions about the underlying distribution of the data, focusing on the Tukey method [55], which establishes as outliers, the data that exceed the following threshold:

$$\text{Outlier} = \begin{cases} \text{True}, & \text{if } x_i < Q_1 - k \cdot \text{IQR or } x_i > Q_3 + k \cdot \text{IQR} \\ \text{False}, & \text{otherwise} \end{cases}$$

where:

- x_i is the value of the data point.
- Q_1 is the first quartile.
- Q_3 is the third quartile.
- IQR is the interquartile range $(Q_3 - Q_1)$.
- k is the threshold, in this case 1.5.

By setting the threshold, the following time series are obtained with their corresponding detected global outliers labeled as red dots. Figure 4.11 displays the throughput outliers,where both very high or low values reflect irregular situations in the network.

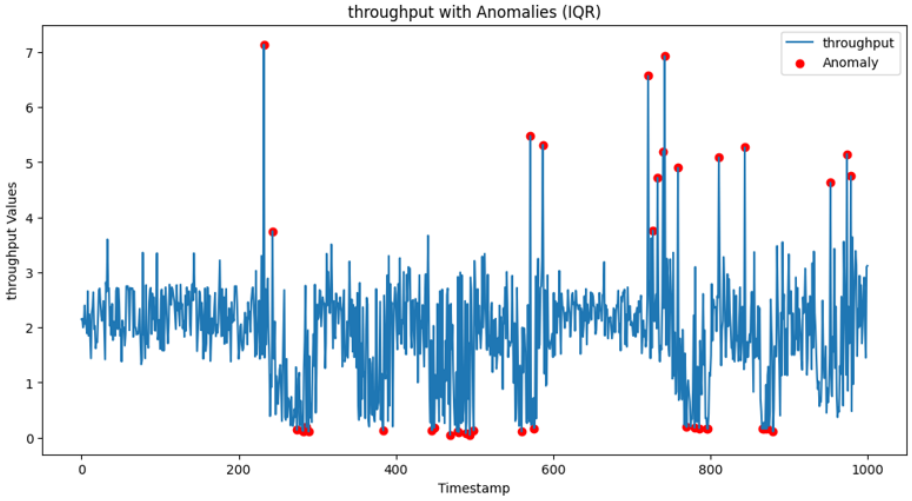


Figure 4.11: Throughput global outliers.

Subsequently, in the Figure 4.12 shows the outliers detected for the congestion variable. The positive correlation obtained in the correlation matrix is verified, due to the fact that the outliers identified in the congestion plot are located in similar areas as the throughput outliers.

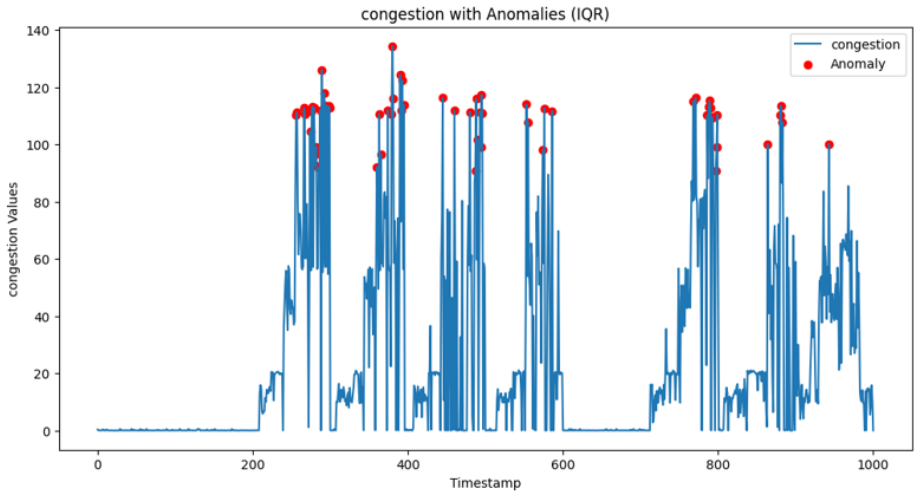


Figure 4.12: Congestion global outliers.

Figure 4.13 illustrates the global outliers identified in the packet loss metric. This graph shows that even in the base scenarios, in which no multimedia content traffic was generated, i.e. from 0 to 200 (scenarios 1 and 2) on the x-axis and from 600 to 700 (scenario 7), packet losses also occur, with peaks of around 20% loss. This indicates the usual network fluctuations.

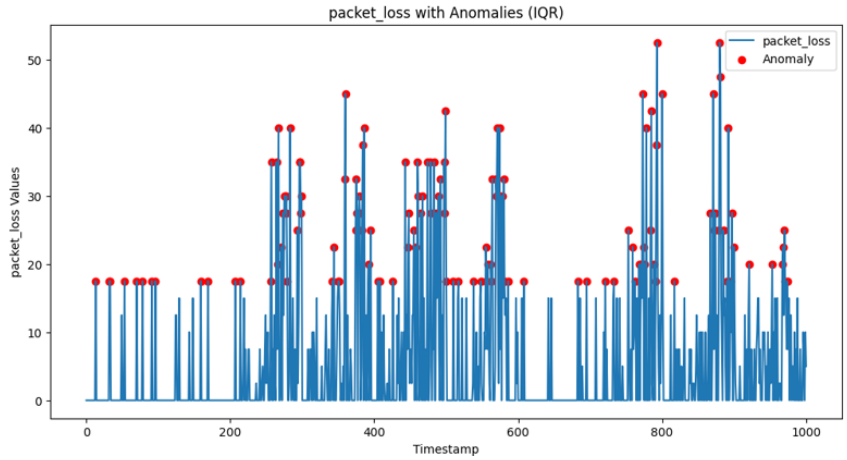


Figure 4.13: Packet loss global outliers.

Lastly, the outliers detected for the jitter variable are illustrated in the Figure 4.14, which has no direct correlation with any of the dataset variables, except for the slight association with congestion. Furthermore, as was observed in the box plots, an increase in jitter values is observed in those scenarios with a higher number of hops.

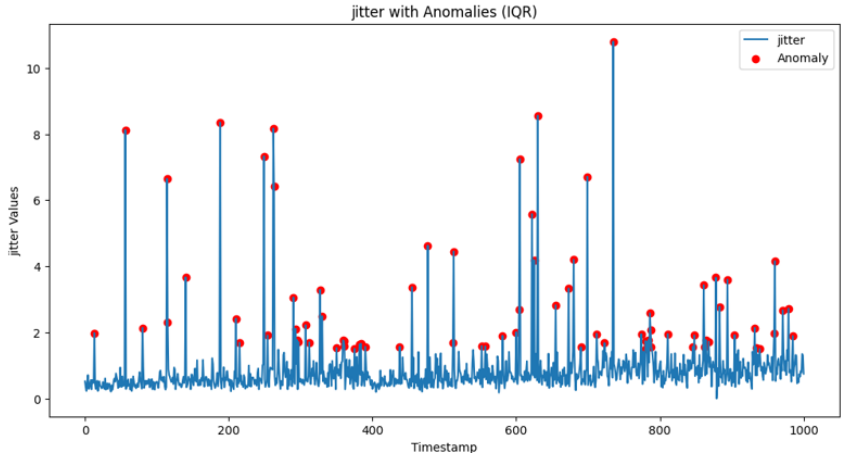


Figure 4.14: Jitter global outliers.

Once the global outliers of each of the metrics have been detected, the following condition is established for the labeling of anomalies. If a row of the dataset has at least two variables detected as outliers, this data becomes an anomaly. After the labeling process, the generated dataset contains a percentage of anomalies of 8.29%:

$$\text{Anomaly} = \begin{cases} \text{True}, & \text{if } \sum_{i=1}^{n} \text{Outlier}_i \geq 2 \\ \text{False}, & \text{otherwise} \end{cases}$$

This labeling method also allows to classify collective anomalies, because a feature's values may be within the usual range, and therefore there are no outliers, but the other features may have anomalous data, becoming it a collective anomaly.

4.4.1. Supervised models

Supervised models are those in which labeled data is provided to the model and the system learns the patterns in order to be able to classify this data. The following three ML models have been implemented, to detect anomalies in the dataset, labelled as class 1 and class 0 being usual data.

For the training of the ML models, due to the imbalance of classes, being the class with anomalies the minority one, Synthetic Minority Over-sampling Technique (SMOTE) is applied [56], which identifies the instances of the minority class in the dataset and creates synthetic samples by interpolating between the instances of the minority class and its close neighbors. With this technique, a balanced dataset with better performance for the ML model is achieved. Secondly, feature scaling is performed because each feature has different units and magnitudes of measurement.

After this data preprocessing, the data is divided into training sets, being this the 80th % and test the 20th % of the remaining data. Cross validation is also applied to avoid inaccurate results. With all these techniques, the following models are trained: Logistic Regression, Random Forest and Support Vector Machine.

Logistic Regression: simple model that allows binary classification. This approach facilitates the interpretation of the results because it provides a probability of belonging to a class. However, its main disadvantage is that it does not handle nonlinear relationships between variables properly. As seen in the following results in Table 4.4, Logistic Regression initially shows promising results, having an accuracy of 0.91, however, the confusion matrix reports that almost 10% of the data are incorrectly classified as false positives (17) and false negatives (16). This misclassification must be due to the model not being able to learn nonlinear relationships that are present in the dataset.

Confusion Matrix:

$$\begin{bmatrix} 167 & 17 \\ 16 & 168 \end{bmatrix}$$

	Precision	**Recall**	**F1-score**	**Support**
Class 0	0.91	0.91	0.91	184
Class 1 (Anomaly)	0.91	0.91	0.91	184
Accuracy			0.91	368
Macro avg	0.91	0.91	0.91	368
Weighted avg	0.91	0.91	0.91	368

Table 4.4: Logistic Regression Classification Report

Random Forest: ML model that builds decision trees and combines them to perform classifications. In particular, this model is able to learn nonlinear relationships, which is a great advantage for network anomaly classification. The results shown in Table 4.5 indicate a higher accuracy around 0.97% and a lower number of misclassifications about 0.03% as false positives (6) or false negatives (5).

Confusion Matrix:

$$\begin{bmatrix} 178 & 6 \\ 5 & 179 \end{bmatrix}$$

	Precision	**Recall**	**F1-score**	**Support**
Class 0	0.97	0.97	0.97	184
Class 1 (Anomaly)	0.97	0.97	0.97	184
Accuracy			0.97	368
Macro avg	0.97	0.97	0.97	368
Weighted avg	0.97	0.97	0.97	368

Table 4.5: Random Forest Classification Report

Support Vector Machine: ML model that searches for the hyperplane in the space that best separates the two classes. It is less intuitive models and depending on the volume of the dataset, it may be less efficient than Random Forest. As shown in Table 4.6, the accuracy of the model is 0.97%, being able to correctly classify data without anomalies of class 0, penalizing class 1, by classifying 0.06% of the data as false positives (12).

Confusion Matrix:

$$\begin{bmatrix} 173 & 12 \\ 0 & 183 \end{bmatrix}$$

	Precision	**Recall**	**F1-score**	**Support**
Class 0	1.00	0.94	0.97	185
Class 1 (Anomaly)	0.94	1.00	0.97	183
Accuracy			0.97	368
Macro avg	0.97	0.97	0.97	368
Weighted avg	0.97	0.97	0.97	368

Table 4.6: Support Vector Machine Classification Report

As a summary of the results of the anomalies detected by each of the three ML models, the following Table 4.7 is presented. It is observed that the model that best classifies the anomalies of the dataset is Random Forest. However, all three trained

models lead to a slight overestimation of the anomalies when detecting a sample as false positives. To improve this slight overestimation of anomalies and thus avoid false alarms even more, strategies can be employed to reduce the number of false positives in a model. These include adjusting the decision threshold, further penalizing false positive errors during training, or regularization techniques to prevent overfitting, which could reduce false positives.

ML model	**Detected Anomalies(%)**
Logistic Regression	15.89%
Random Forest	8.89%
Support Vector Machine	11.98%
Actual anomalies of the dataset	8.29%

Table 4.7: Percentage of detected anomalies.

Once the supervised models have been trained, techniques such as SHAP values are used to explain the decisions made by the models. A series of examples and the conclusions drawn are presented in the following sections.

Figure 4.15 shows in order those metrics that have the greatest influence on the model's predictions. The first variable is congestion, in which high values of this parameter have an impact of almost 0.5 on the prediction. The next most influential metric is high values of packet loss, as well as high values of latency. As a fourth variable, there are high jitter values and low throughput rates. Finally, the maximum bandwidth has no influence on the prediction because its value does not vary throughout the dataset.

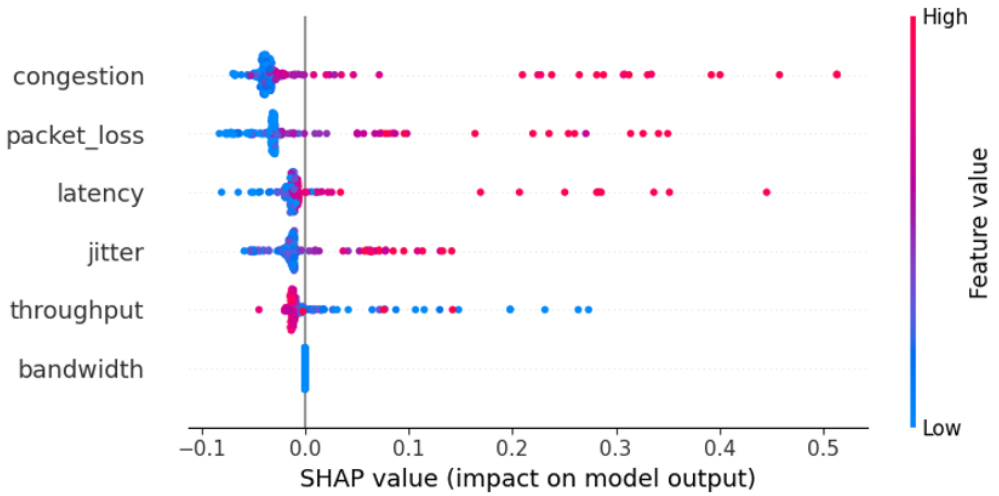


Figure 4.15: SHAP values.

Figure 4.16 shows the force plot result, which shows exactly which features had the most influence on the model's prediction for a single observation, which in this case is an anomalous observation. In this case, it is observed that the feature with the greatest influence is congestion, which pushes the prediction up, followed by the value of packet loss and throughput. Otherwise, a low value of jitter pushes the model prediction down, followed to a minor extent by latency.

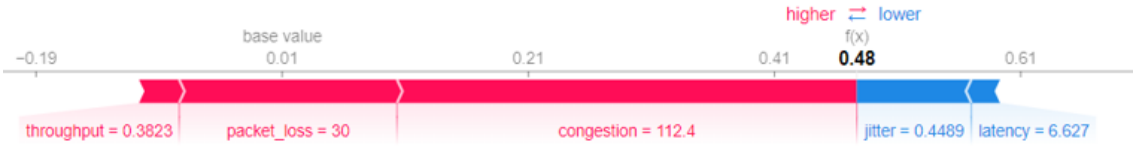


Figure 4.16: SHAP example anomaly data.

In contrast, Figure 4.17 shows an example of a non-anomaly observation, in which the value of 0% packet loss greatly pushes the model down towards class 0, which it classifies as non-anomaly.

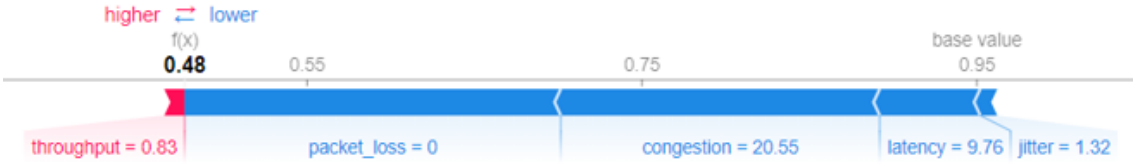


Figure 4.17: SHAP example normal data.

4.4.2. Unsupervised models

Unsupervised models are those that without being provided by labeled data, are able to discover patterns and insights without any explicit guidance or instruction. For the development of the unsupervised model two techniques are applied [57].

Firstly, change points by using the Dynp tool [58] that by dynamic programming divides the time series into subproblems and analyzes the statistical properties of these segments to detect change points (CP). This technique allows to analyze the data and detect possible temporal anomalies based on the CP in the time series.

Secondly, a clustering technique is applied for the detection mechanism, which allows a more detailed analysis of the data. Clustering methods encompass the choice of clustering algorithm and the selection of an appropriate distance measure. As a clustering algorithm, k-means is chosen because it is relatively simple to implement and easy to interpret. On the other hand, for the distance measure, l2-norm, which is the Euclidean distance, is discarded because it is not capable of capturing nonlinear

relationships and handling outliers.

Because of this, Dynamic Time Warping (DTW) similarity is selected as the distance measure of the k-means clustering because it provides a robust measure in cluster development [59]. DTW addresses this challenge by allowing for flexible alignment between two time series. It finds the best way to adjust the sequences, minimizing the distance between the corresponding points.

DTW is robust to temporal distortions, such as time shifts within the sequences. This is particularly important in applications where the timing of events may vary or when comparing sequences of different lengths. Unlike simpler distance measures, DTW considers all possible alignments between the sequences because it dynamically adjusts the alignment to find the best match, even in the presence of outliers. DTW ability to capture hidden variations and complex patterns makes it well suited for detecting temporal anomalies.

As results, the next graphs are obtained with these two techniques for the network metrics obtained from the dataset. Figure 4.18 shows the time series of the throughput variable along the different scenarios of the test plan, continued below by the same time series with the CP marked as dashed red lines, as well as the 3 clusters obtained with k-means.

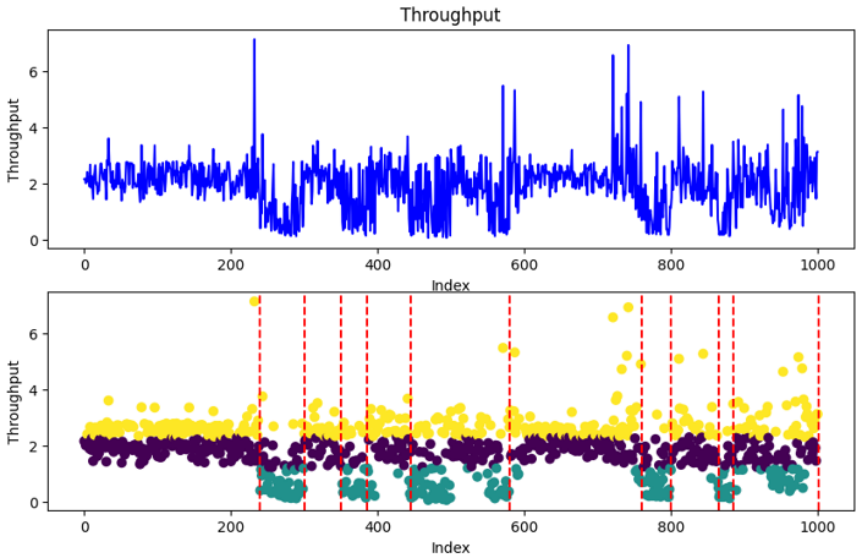


Figure 4.18: Throughput time series.

The change points detected occur at times when the throughput goes to low values indicated by the green cluster from 0 to 2 Mbps, which are indicative of possible errors. Furthermore, these change points are similar to the outliers detected with the interquartile range threshold. The following Figure 4.19 shows the congestion CPs, which occur when the values exceed 60% congestion, as indicated by the yellow cluster. Note that the yellow dots at the end of the time series are not detected as change points because they are isolated, not together like the data from index 0 to 800.

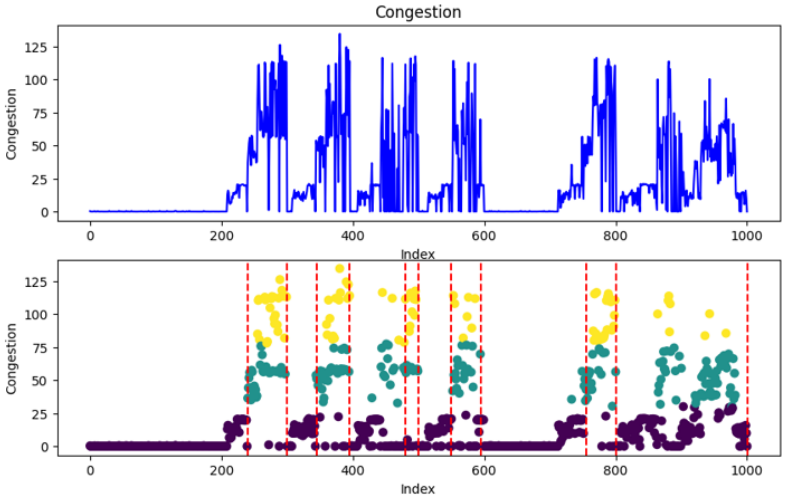


Figure 4.19: Congestion time series.

The packet loss metric is shown in Figure 4.20, in which the possible anomalies detected in the time intervals in which the packet loss rises above 20%, the cluster with the green dots, are quite visual. These green dots could be identified as severe anomalies, leaving the yellow cluster dots as slight anomalies.

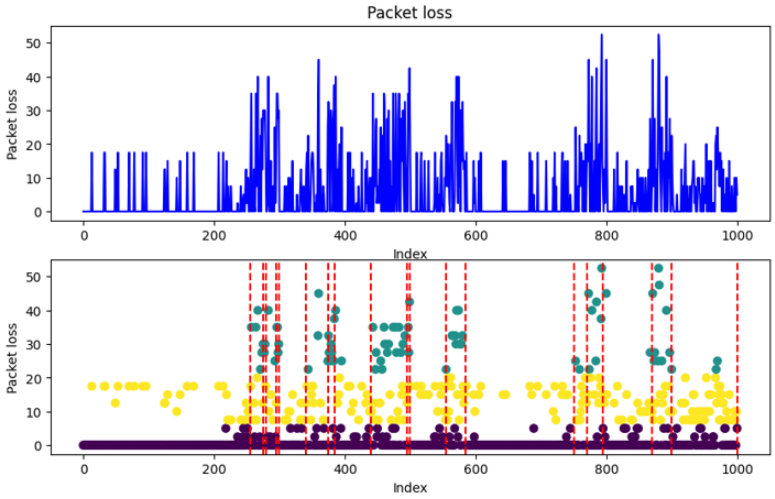


Figure 4.20: Packet loss time series.

4.5. Anomaly prediction

Network anomaly detection is continually evolving, making it possible to identify and address problems. However, the next challenge is the prediction of these anomalies, preventing service failures and malfunctions. This predictive capability is crucial to guarantee an optimal user experience, ensuring continuity and quality of service

at every moment. The implementation of anomaly prediction systems requires a comprehensive approach that involves analyzing historical data patterns and using advanced ML techniques.

The Long Short-Term Memory (LSTM) network structure is implemented to process the data patterns and provide predictions [60]. To achieve this, the generated dataset is loaded and the data are prepared for the LSTM by scaling them. Then, the training and testing data are created, the LSTM structure is defined, trained and evaluated by the test set. The network structure has been defined by using two LSTM layers, followed by a dense layer that has a single neuron because it provides the prediction. The use of two LSTM layers, instead of a single one, is done so that the model has a higher learning capacity, allowing the collection of more complex patterns and relationships.

Figure 4.21 shows the LSTM result for the dataset congestion metric. It shows the predicted congestion values in orange, versus the actual blue time series data of the test set, which corresponds to 20% of the samples, because the other 80% was used for LSTM training. At first glance, the LSTM makes a reasonable prediction of congestion, however, the predicted values do not reach the actual values of the highest peaks because the model is more conservative and tends not to overestimate the values. This conservative pattern is also observed for other metrics of the dataset.

These neural networks can sometimes become too conservative. To mitigate this behavior and make the model more sensitive, it is essential to adjust various aspects of the architecture and training process. For example, increasing the number of LSTM units and layers, implementing regularization techniques, using learning rate adjustment or early stopping. In addition to these strategies, increasing the diversity and amount of training data can help the model to learn more complex patterns.

The use of LSTM for prediction of the entire set of metrics in the dataset may not provide accurate results due to this conservative pattern. However, the prediction approach could be valuable to obtain the measured values of an individual metric based on the data obtained from the other variables in the dataset.

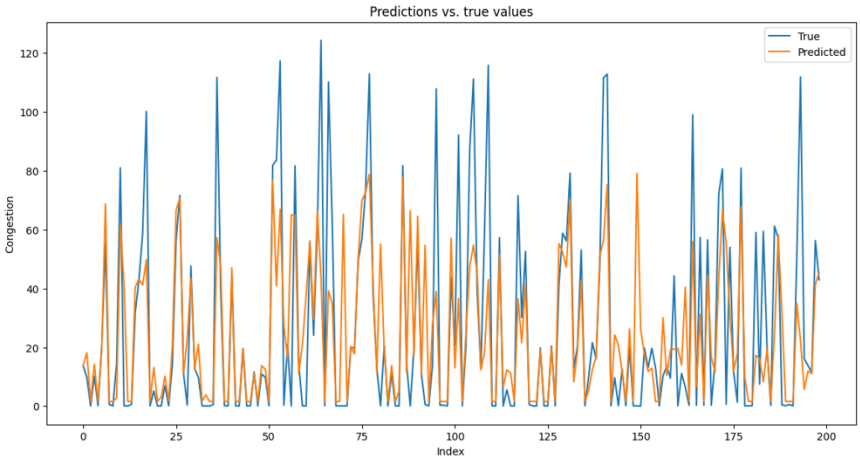


Figure 4.21: LSTM congestion.

4.6. Final system

Once the anomaly detection algorithms have been developed, they are deployed in Streamlit, a platform designed for the creation of interactive web applications with the Python language. It was decided to implement this functionality so that the user can interact with the ML models, improve the experience and the visual interface of the algorithms, as well as the deployment of these into services.

The interactive web application developed with Streamlit can be deployed locally through the localhost address or deployed remotely, which is more practical. Therefore, a localtunnel is implemented, which allows to share the web service of the local development machine without the need to disable or modify the firewall settings or DNS troubles. This localtunnel provides each execution a unique publicly accessible URL that will proxy all requests from the locally running web server.

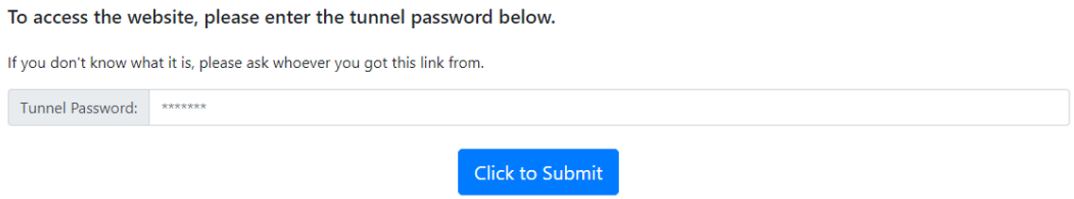


Figure 4.22: Streamlit localtunnel.

To develop the interactive web application, the necessary NodeJs library is installed, which is an engine that is used to execute JavaScript code on the server side. Once installed, the Python script app.py containing the network anomaly detection algorithms is prepared. It should be noted that the Python script was performed using the Streamlit documentation API commands. Once developed, the following command is executed to start application on the chosen port:

```
streamlit run app.py & npx localtunnel --port 8501
```

After executing the command, Streamlit provides a URL, which is a unique address, in which for increased security, the password of the created tunnel must be entered as shown in Figure 4.22.

Once the password is entered, the user has the possibility to perform two operations as seen in Figure 4.23, anomaly detection using the developed ML models and prediction of future values with the implemented LSTM.

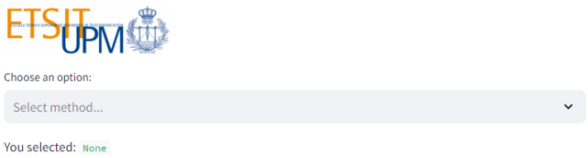


Figure 4.23: Anomaly Detection System front page.

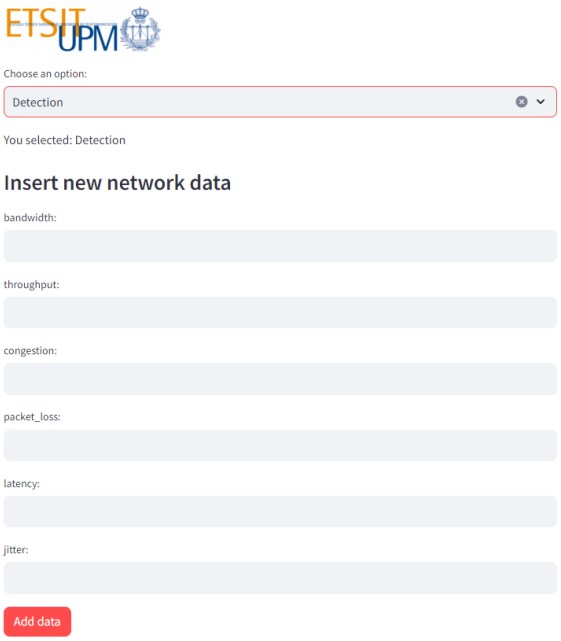


Figure 4.24: Anomaly Detection System detection.

The detection process begins with the input by the user of new data of network values as Figure 4.24, which are then labeled as anomaly or not depending on the threshold established in this project. Then, the trained supervised models perform the prediction of the new data and its subsequent classification as anomaly or not, showing both the results of the new data (Figure 4.25) with the corresponding updated dataset which could be downloaded and the evaluation metrics of the ML models.

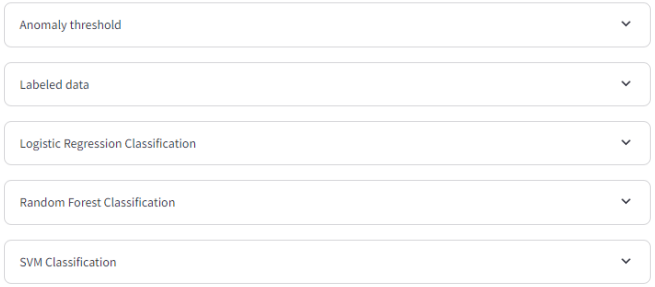


Figure 4.25: Anomaly Detection System results.

The prediction process consists of the uploading of the dataset generated in this project and the subsequent training of the LSTM neural network, which provides as results the predicted values versus the real ones as Figure 4.26.

Figure 4.26: Anomaly Detection System prediction.

Chapter 5

Conclusions

5.1. Discussion

The importance and the wide possibilities offered by ML techniques to solve any real-world problem, in this case the detection of anomalies, have been highlighted. The models developed with ML provide us with additional resources, as well as a greater capacity for data analysis, allowing a detailed analysis of the data, and the discovery of intrinsic patterns.

On the one hand, the supervised ML models developed have allowed the detection of global anomalies according to a previously established threshold. After simulating network traffic through multimedia content in the different scenarios of the test plan, an extensive and detailed dataset is obtained, which includes the characteristics of each scenario. In addition, real-time metrics are collected using the network probe, covering from throughput to maximum network capacity. This approach enables a complete representation of the network environment, which is essential for the development of accurate anomaly detection models. Once the dataset is generated, the development of the supervised ML models starts with the generation of synthetic samples using programming libraries to correct the class imbalance due to the existence of the majority class of data without anomalies versus the minority class. Once a balanced dataset is obtained, data scaling is performed to avoid model misperformance, as well as the implementation of cross validation to avoid biases.

With the mentioned techniques, the development of the Logistic Regression model is started, which is observed to obtain the worst results because it is not able to learn non-linear patterns from the data. Because of this, the training of Random Forest and Support Vector Machine, models that obtain good results, highlighting Random Forest by performing the lowest number of instances classified as false positives, decreasing the number of false alerts.

On the other hand, the application of techniques to detect changes in the time series and unsupervised clustering techniques, provides another point of view in the detection of anomalies since it is not required to label the data as anomalies, avoiding the need to establish a threshold, which varies for each situation and problem. These techniques visually provide the relationships between normal and anomalous data, indicating the exact moment at which the statistical time series data changes due to

the state of the network and the measured multimedia content traffic.

5.2. Conclusions

In this study, an exhaustive comprehensive overview of ML algorithms for the development of a network anomaly detection system has been conducted. According to the objectives of Chapter 1, the different existing software and applications for the detection of anomalies, in particular, network anomalies, have been reviewed. A successful dataset was developed for anomaly detection in an emulated environment, which has enabled extensive analysis of the network. Through this analysis, the relationship between the network metrics in the different scenarios was identified. Among them, the increase of packet loss when increasing congestion, or the low correlation of jitter with the other variables of the dataset are highlighted. The sophisticated t-SNE dimensionality reduction technique allowed the study of the network behavior in the different scenarios due to the separation of the data into groups. By establishing a threshold with a non-parametric technique, the data were labeled into anomalous and non-anomalous data.

Successful ML models were developed, highlighting the Random Forest supervised model with an accuracy of 97% with the lowest false positive rate, followed by Support Vector Machine with an accuracy of 97% and the weakest one, Logistic Regression with an accuracy of 91% and a high false alarm rate. The sophisticated SHAP technique provided insights about the model predictions, concluding that the most influential variables in anomaly classification are congestion and packet loss. Unsupervised models such as k-means were also implemented to discover patterns and detect temporal anomalies with CPs. In addition, a prediction model was developed using LSTM neural network capable of predicting the data by training it with metrics from the generated data. Finally, an interactive web application was implemented that combines the ML models developed, allowing the insertion of new data and its subsequent anomaly detection in a satisfactory performance, fulfilling the objectives presented in Chapter 1.

Considering the results presented in this master thesis, it is concluded that the anomaly detection system developed for the network topology of Figure 4.1 allows the classification of the data both quantitatively and qualitatively, in anomalous data and normal data in an objective and precise way. Furthermore, through the development of a LSTM, the possibility of predicting network metrics data with the created dataset is observed, allowing the possibility of improving and developing anomaly prediction systems through recurrent neural network techniques used in deep learning to process and predict data sequences.

In conclusion, the development of this master thesis can be of great use for network administrators, allowing them to detect anomalies in network environments in an objective manner. This will facilitate the analysis of new strategies for the categorization and detection of anomalous network data in the research community.

5.3. Future research lines

This master thesis has achieved the development of ML algorithms for the detection of network anomalies in the simulated network scenario automatically and accurately. Several models have been developed for anomaly detection, both supervised and unsupervised, as well as a recurrent neural network for the prediction of new data. In order to improve the quality of the developed system, the following future lines are proposed:

- Prediction of a network metric based on the data of the other network variables collected in order to launch a network probe that performs the measurement of fewer parameters and therefore, consumes less bandwidth avoiding overloading the network.
- Detect anomalies based on whether they are collective or contextual depending on the time and the different events that occur during the 365 days of the year. Currently, the system is able to detect global anomalies based on the set threshold, which will improve detection by including more types of anomalies.
- Search and implement a network scenario with higher bandwidth and node capacities, for the consequent generation of a larger dataset. Currently, the scenario created with the emulator contains control plane images, which are used for the management of the network itself, not intended for sending multimedia content.
- Creation of a simulated scenario with different network maps, as well as a larger number of connected devices. Designing and implementing several network topologies could improve the simulation by performing realistic real-world scenarios.
- Capability to perform manipulation of the network links to introduce changes in the generated network traffic, by introducing modifications in a given router. Altering parameters at specific nodes can simulate various conditions and observe their impact on the overall network traffic.
- Consider throughput as a percentage of bandwidth rather than a specific value. This approach will allow scaling the results more effectively to other systems with different bandwidth capacities, facilitating the comparison and generalization of the results.
- Extension of the microservice to a new network, so that a new data extraction is performed, and subsequently comparison.

References

[1] The importance of an efficient data network in an enterprise - ultimate technology. https://ultimate.com.co/red-datos-eficiente/ [Access date: 10-03-2024].

[2] What is anomaly detection? - anomaly detection in machine learning explained - aws. https://aws.amazon.com/es/what-is/anomaly-detection/ [Access date: 10-03-2024].

[3] Nusaybah Alghanmi, Reem Alotaibi, and Seyed M. Buhari. Machine learning approaches for anomaly detection in iot: An overview and future research directions. *Wireless Personal Communications*, 122:2309–2324, 2 2022.

[4] J. Jabez and B. Muthukumar. Intrusion detection system (ids): Anomaly detection using outlier detection approach. *Procedia Computer Science*, 48:338–346, 1 2015.

[5] Len Feremans, Boris Cule, and Bart Goethals. Efficient pattern-based anomaly detection in a network of multivariate devices, 2023.

[6] Basheer Husham Ali, Nasri Sulaiman, Syed Abdul Rahman Al-Haddad, Rodziah Atan, Siti Lailatul Mohd Hassan, and Mokhalad Alghrairi. Identification of distributed denial of services anomalies by using combination of entropy and sequential probabilities ratio test methods. *Sensors (Basel, Switzerland)*, 21, 10 2021.

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection. *Encyclopedia of Machine Learning and Data Mining*, pages 1–15, 2016.

[8] Gilberto Fernandes, Joel J.P.C. Rodrigues, Luiz Fernando Carvalho, Jalal F. Al-Muhtadi, and Mario Lemes Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems 2018 70:3*, 70:447–489, 7 2018.

[9] Tianyuan Lu, Lei Wang, and Xiaoyong Zhao. Review of anomaly detection algorithms for data streams. *Applied Sciences 2023, Vol. 13, Page 6353*, 13:6353, 5 2023.

[10] Stefania Russo, Michael D. Besmer, Frank Blumensaat, Damien Bouffard, Andy Disch, Frederik Hammes, Angelika Hess, Moritz Lürig, Blake Matthews, Camille Minaudo, Eberhard Morgenroth, Viet Tran-Khac, and Kris Villez. The value of human data annotation for machine learning based anomaly detection in environmental systems. *Water Research*, 206:117695, 11 2021.

[11] Semantic metadata annotation for network anomaly detection. https://www.ietf.org/archive/id/

draft-netana-opsawg-nmrg-network-anomaly-semantics-01.html [Access date: 10-03-2024].

[12] Bizum guarantees that there will be no drop in christmas purchases. https://www.elconfidencialdigital.com/articulo/dinero/bizum-garantiza-que-habra-caidas-compras-navidad/20231216000000688889.html [Access date: 10-04-2024].

[13] Prabhaker Mishra, Chandra Pandey, Uttam Singh, Amit Keshri, and Mayilvaganan Sabaretnam. Selection of appropriate statistical methods for data analysis. *Annals of Cardiac Anaesthesia*, 22:297, 7 2019.

[14] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in classification. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2888:986–996, 2003.

[15] Mordechai Ben-Ari and Francesco Mondada. Finite state machines. *Elements of Robotics*, pages 55–61, 2018.

[16] Vikramkumar, Vijaykumar B, and Trilochan. Bayes and naive bayes classifier. 4 2014.

[17] Theodoros Evgeniou and Massimiliano Pontil. Support vector machines: Theory and applications. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2049 LNAI:249–257, 2001.

[18] Samuel Schmidgall, Jascha Achterberg, Thomas Miconi, Louis Kirsch, Rojin Ziaei, S. Pardis Hajiseyedrazi, and Jason Eshraghian. Brain-inspired learning in artificial neural networks: a review. 5 2023.

[19] Marko Popovic. Researchers in an entropy wonderland: A review of the entropy concept.

[20] Shuyi Ji, Zizhao Zhang, Shihui Ying, Liejun Wang, Xibin Zhao, and Yue Gao. Kullback-leibler divergence metric learning. *IEEE Transactions on Cybernetics*, 52:2047–2058, 4 2022.

[21] What is speed in networking 2023? https://theoriesbyvicky.in/networking/what-is-speed-in-networking/ [Access date: 10-03-2024].

[22] Github - elisejiuqizhang/ts-ad-datasets: Public datasets for time series anomaly detection. https://github.com/elisejiuqizhang/TS-AD-Datasets [Access date: 15-03-2024].

[23] Ralf C. Staudemeyer and Christian W. Omlin. Extracting salient features for network intrusion detection using machine learning methods. *South African Computer Journal*, 52, 6 2014.

[24] The 41 features provided by the kdd cup '99 datasets. | download table. https://www.researchgate.net/figure/The-41-features-provided-by-the-KDD-Cup-99-datasets_tbl1_263274883 [Access date: 20-11-2023].

[25] Chao Meng, Xue Song Jiang, Xiu Mei Wei, and Tao Wei. A time convolutional network based outlier detection for multidimensional time series in cyber-physical-social systems. *IEEE Access*, 8:74933–74942, 2020.

[26] Numenta anomaly benchmark evaluates anomaly detection techniques for real-time data | business wire. https://www.businesswire.com/news/home/20151110006297/en/Numenta-Anomaly-Benchmark-Evaluates-Anomaly-Detection-Techniques\-for-Real-time-Streaming-Data [Access date: 20-12-2023].

[27] Jordi Domingo, Pere Barlet Ros, and Josep Solé Bonet. Smartxac: Sistemas de monitorización y análisis de tráfico para la anella científica. *RedIRIS: boletín de la Red Nacional de I+D RedIRIS, ISSN 1139-207X, Nº. 66-67, 2003, págs. 27-33*, pages 27–33, 2003.

[28] Advanced broadband communications center (ccaba) universitat politècnica de catalunya (upc) smartxac: A passive monitoring and analysis system for high-speed. - ppt download. https://slideplayer.com/slide/5046612/ [Access date: 02-01-2024].

[29] Mercedes Barrionuevo, Mariela Lopresti, Natalia Carolina Miranda, and María Fabiana Piccoli. Un enfoque para la detección de anomalías en el tráfico de red usando imágenes y técnicas de computación de alto desempeño. 2016.

[30] Optimizar nuestra red: La herramienta indispensable para medir el rendimiento. https://rcg-comunicaciones.es/optimizar-nuestra-red-mi-herramienta/ [Access date: 02-01-2024].

[31] iperf3 python wrapper — iperf3 0.1.10 documentation. https://iperf3-python.readthedocs.io/en/latest/ [Access date: 15-03-2024].

[32] Cómo medir el ancho de banda con iperf | oastic. https://oastic.com/es/posts/how-to-test-bandwith-with-iperf/ [Access date: 02-01-2024].

[33] Bandwidth testing using iperf – ojit. https://www.ojit.co.uk/bandwidth-testing-using-iperf/ [Access date: 02-01-2024].

[34] Manuel Vidal Vázquez. Comparativa de herramientas libres de generación de tráfico de red.

[35] Alberto Del Río. Github - kaiser-14/network-performance-probe. https://github.com/Kaiser-14/network-performance-probe [Access date: 20-11-2023].

[36] [programación de servicios y procesos]. https://psp.codeandcoke.com/apuntes:sockets [Access date: 02-01-2024].

[37] Tutoriales ffmpeg para principantes - muylinux. https://www.muylinux.com/2010/03/25/tutoriales-ffmpeg-para-principantes/ [Access date: 02-02-2024].

[38] ¿qué es ffmpeg y por qué provoca un uso intensivo de la cpu? | qnap (es). https://www.qnap.com/es-es/how-to/faq/article/qu%C3%A9-es-ffmpeg-y-por-qu%C3%A9-provoca-un-uso-intensivo-de-la-cpu [Access date: 02-02-2024].

[39] Cisco packet tracer - networking simulation tool. https://www.netacad.com/es/courses/packet-tracer [Access date: 10-03-2024].

[40] Qué es eve-ng? el «nuevo» emulador para redes | info++. https://cesarcabrera.info/que-es-eve-ng-un-nuevo-emulador-para-redes/ [Access date: 20-11-2023].

[41] Eve emulated virtual environment | blog vmware vsphere & cloud. https://www.josemariagonzalez.es/vmware-nsx/eve-emulated-virtual-environment.html [Access date: 02-02-2024].

[42] Eve-ng. https://www.eve-ng.net/ [Access date: 02-02-2024].

[43] John Bosco Ssemakula, Juan Luis Gorricho, Godfrey Kibalya, and Joan Serrat-Fernandez. Deployment of future services in a multi-access edge computing environment using intelligence at the edge. *Journal of Network and Systems Management*, 31, 10 2023.

[44] Anzola Rojas, Durán Barroso, Ramón José, Miguel Jiménez, Ignacio de, Parra Domínguez, Camilo Anzola-Rojas, Javier Parra-Domínguez, and André Chaves. Multi-access edge computing: Características y aplicación en entornos rurales de baja densidad de población.

[45] Sonia Shahzadi, Muddesar Iqbal, Tasos Dagiuklas, and Zia Ul Qayyum. Multi-access edge computing: open issues, challenges and future perspectives. *Journal of Cloud Computing*, 6:1–13, 12 2017.

[46] Deploy machine learning app built using streamlit and pycaret on google kubernetes engine | by moez ali | towards data science. https://towardsdatascience.com/deploy-machine-learning-app-built-using-streamlit-and-pycaret-on-google-kubernetes-engine-fd7e393d99cb [Access date: 10-04-2024].

[47] Cisco xrv9k - containerlab. https://containerlab.dev/manual/kinds/vr-xrv9k/ [Access date: 10-03-2024].

[48] Cisco ios xrv 9000 router installation and configuration guide - cisco. https://www.cisco.com/c/en/us/td/docs/routers/virtual-routers/configuration/guide/b-xrv9k-cg.html [Access date: 1-02-2024].

[49] Anaconda | the operating system for ai. https://www.anaconda.com/ [Access date: 22-03-2024].

[50] Ian T. Jollife and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 374, 4 2016.

[51] Jyoti Pareek and Joel Jacob. Data compression and visualization using pca and t-sne. *Lecture Notes in Networks and Systems*, 135:327–337, 2021.

[52] Shap documentation — shap latest documentation. `https://shap.readthedocs.io/en/latest/` [Access date: 10-03-2024].

[53] Lukas Wüsteney, David Hellmanns, Markus Schramm, Lukas Osswald, René Hummen, Michael Menth, and Tobias Heer. Analyzing and modeling the latency and jitter behavior of mixed industrial tsn and detnet networks. *CoNEXT 2022 - Proceedings of the 18th International Conference on emerging Networking EXperiments and Technologies*, pages 91–109, 11 2022.

[54] Łukasz Wawrowski, Andrzej Białas, Adrian Kajzer, Artur Kozłowski, Rafał Kurianowicz, Marek Sikora, Agnieszka Szymańska-Kwiecień, Mariusz Uchroński, Miłosz Białczak, Maciej Olejnik, and Marcin Michalak. Anomaly detection module for network traffic monitoring in public institutions. *Sensors (Basel, Switzerland)*, 23, 3 2023.

[55] Tukey fences for outliers. `https://community.ibm.com/community/user/ai-datascience/blogs/moloy-de1/2021/03/23/points-to-ponder` [Access date: 15-03-2024].

[56] Hui Han, Wen Yuan Wang, and Bing Huan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. *Lecture Notes in Computer Science*, 3644:878–887, 2005.

[57] Sotiris Skaperas, Lefteris Mamatas, and Vassilis Tsaoussidis. A link-quality anomaly detection framework for software-defined wireless mesh networks. *IEEE Transactions on Machine Learning in Communications and Networking*, 2:495–510, 2024.

[58] Dynamic programming - ruptures. `https://centre-borelli.github.io/ruptures-docs/user-guide/detection/dynp/` [Access date: 2-04-2024].

[59] Ali Javed, Byung Suk Lee, and Donna M. Rizzo. A benchmark study on time series clustering. *Machine Learning with Applications*, 1:100001, 9 2020.

[60] Benjamin Lindemann, Timo Müller, Hannes Vietz, Nasser Jazdi, and Michael Weyrich. A survey on long short-term memory networks for time series prediction. *Procedia CIRP*, 99:650–655, 1 2021.

[61] How much energy a server consumes and how much co2 it emits - news, events and articles in vigunu.com. `https://vigunu.com/eventsnews/2020/10/03/cuanta-energia-consume-un-servidor-y-cuanto-co2-emite/` [Access date: 1-04-2024].

[62] Transparency portal | universidad politécnica de madrid. https://transparencia.upm.es/personal/retribuciones [Access date: 1-04-2024].

[63] Pre-doctoral and post-doctoral researcher salaries | transparency portal. https://www.ucm.es/portaldetransparencia/retribuciones-investigador-predoctoral-posdoctoral [Access date: 1-04-2024].

Appendix A

Ethical, economic, social and environmental issues

Once this master thesis was accomplished, an analysis of the impact of this project on ethics, the economy, society and the environment was carried out.

A.1. Introduction

Network anomaly detection is of vital importance for the correct development of communications systems. Creating a dataset in a simulated environment allows an exhaustive analysis and knowledge of the network, for the subsequent development of AI models with the data collected. Once the detection system has been developed with predictive algorithms, anomalies are able to be identified accurately, without human intervention, allowing the anticipation and correction of the network in an anomalous situation. Each of the ethical, economic, social and environmental impacts of this master thesis project are discussed in more detail as follows.

A.2. Description of relevant impacts related to the project

- **Ethical implications**: Ethics in the field of computer engineering, especially in projects focused on the detection and prediction of anomalies, required to establish a balance, fairness and respect for the privacy of the individuals. This implies careful consideration of the patterns that might emerge, as well as the realization of backups to protect the integrity of sensitive data.

  In addition, it is crucial to establish mechanisms to address ethically situations, such as the handling of personal information and the management of results, in order to maintain trust and transparency at all stages of a project involving user data. In this master thesis, user data is not employed directly because content delivery services are simulated on the network using several tools.

- **Economic impact**: The analysis of the economic impact of the project reveals a number of situations that go beyond monetary losses due to service interruptions

or cyber-attacks. There will be significant losses, as well as indirect costs such as loss of customer confidence and damage to the reputation of the communications service provider. Therefore, the ability to detect and prevent anomalies as quickly as possible becomes very important, as it can help to mitigate these risks and improve operational efficiency.

When evaluating the cost-benefit of implementing anomaly detection systems, it is essential to consider not only the immediate costs of the technology, but to take into account the future benefit that a robust anomaly detection system provides.

- **Social implications**: By providing tools that enhance security at the individual and global level, protection against potential threats is achieved, building a safer and more reliable digital environment for all users of the communications system.

- **Environmental impact**: Servers consume a significant amount of energy and resources, which can have negative consequences for the environment [61]. Some important aspects to take into account for this project include optimization mechanisms in the developed algorithms for the decrease of necessary resources since the technology has the following disadvantages with respect to the environment.

  Servers require a constant constant amount of energy to operate, both to power the internal components and to maintain an adequate temperature in the data centers. This energy consumption contributes to greenhouse gas emissions and the consumption of non-renewable resources. Servers leave a huge carbon footprint, so the use of renewable energy to reduce this impact is vital importance.

  Secondly, server cooling is another important factor to consider. Data centers need efficient cooling systems to maintain an optimal temperature, which often involves additional energy consumption. The use of more efficient cooling systems and management technologies can help reduce this impact.

  Besides, as servers become obsolete or are upgraded, large amounts of e-waste are generated. This discarded often contain hazardous materials that can contaminate soil and water if not properly managed. Adopting recycling and reuse practices can help mitigate this environmental impact.

  Also, it is worth mentioning, that the location of data centers can also have an impact on the environment. For example, building data centers in areas areas with access to renewable energy can reduce the carbon footprint of the server.

A.3. Conclusions

As mentioned in this appendix, the automatic anomaly detection system developed in this master thesis generates great benefits in several areas. The project not only focuses on technological and scientific progress, but also considers ethical, economic, social and environmental aspects.

Appendix B

Financial budget

This master thesis has been carried out with the collaboration of the Group of Visual Telecommunications Application (GATV) of the Universidad Politécnica de Madrid and Telefónica, using the resources provided to me. In this appendix the budget is broken down into two blocks, the cost associated with personnel and material costs. Finally, the total budget derived from this master thesis is presented.

- **Personnel**: salaries have been established according to sources obtained from public organizations in the Community of Madrid, including the Universidad Politécnica de Madrid [62] and the Universidad Complutense [63]. With the salaries mentioned above, the associated personnel costs have been estimated in Table B.1.

	Hourly rate (€)	**Hours**	**Total (€)**
Project director 1	60	40	2,400
Project director 2	80	20	1,600
Engineering student	25	480	12,000
Telefónica employee	40	20	800
TOTAL			**16,800**

Table B.1: Personnel costs.

- **Material resource costs**: the cost of all the resources used for this master thesis is described, taking into account the associated expenses due to the amortization of each of the elements (see Table B.2).

	Lifetime (years)	**Units**	**Cost (€)**	**Amortization (€/month)**	**Usage (months)**	**Total (€)**
Personal computer	5	1	850.00	14.17	9	127.50
TID Servers	4	3	1,053.28	17.55	6	315.90
EVE-NG professional license	1	0	0	0	6	1,017.60
Probe and network tools	1	0	0	0	9	0
TOTAL						**1,461.00**

Table B.2: Material resource costs.

Once the budget has been detailed and classified into personnel costs and material resource costs, the total sum of the expenses shown above is indicated as follows (see Table B.3).

	Cost
Personnel costs	16,800€
Material costs	1,461.00€
Subtotal	18,261.00€
IVA	3,834.81€
Total	**22,095.81€**

Table B.3: Total costs.

Appendix C

GitHub repository

The network anomaly detection system can be found in the following Github repository `https://github.com/pilarschummer/AnomalyDetectionSystem.git`. The scripts are organized by system functionalities, including exploratory data analysis, anomaly labeling according to the desired threshold, training and prediction of new data, as well as training of an LSTM for variable prediction.

The execution of the Python scripts is done in a customized way by means of arguments. The user can specify the desired dataset, as well as different parameters such as the classification algorithm to be used (–logistic_regression, –random_forest, –svm), the new data to be introduced to detect whether it is an anomaly or not (–new_data) and the desired metric to be predicted in the prediction script using a LSTM. The requirements and steps to execute the code are as follows:

1. **Python installation:** the Python programming language must be installed to run the scripts. The latest stable version can be downloaded directly from the web site. During installation, the option "Add Python to PATH" should be checked to facilitate command line usage.

2. **Open command terminal:** Once the terminal is open, navigate to the destination directory where the code is to be downloaded:

```
cd path/directory/target
```

3. **Download the code:** to perform this step you can run the following command which clones the repository:

```
git clone https://github.com/pilarschummer/
   AnomalyDetectionSystem.git
```

4. **Run the code:** depending on the desired Python script to be executed, the commands are as follows:

```
python labelAnomaly.py dataset.csv #Threshold based
    anomaly labeling
python classification.py logistic_regression dataset.
    csv --new_data 1.0 2.0 3.0 4.0 5.0 6.0 #Detection
python predictionLSTM.py dataset.csv throughput #
    Prediction
```

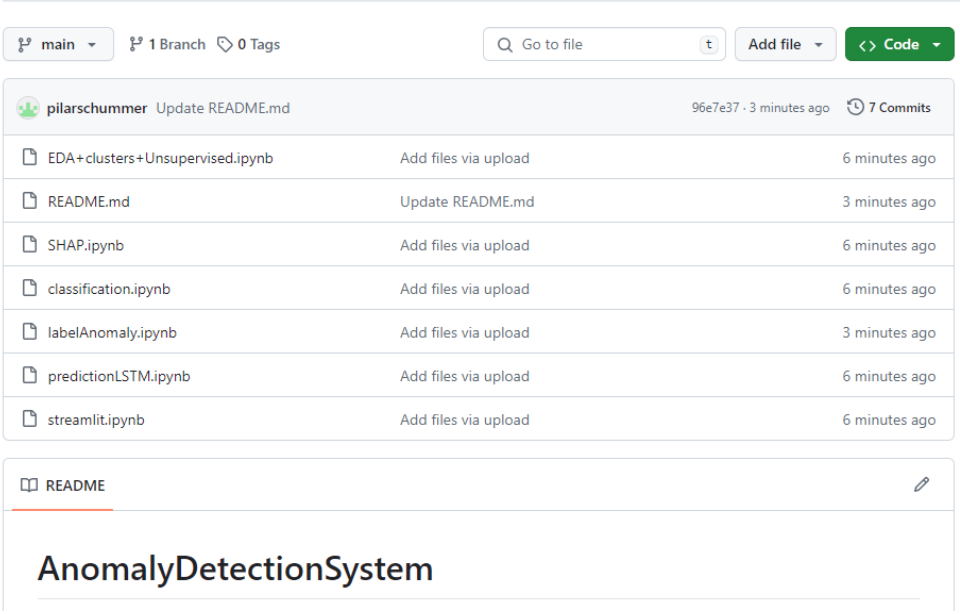


Figure C.1: GitHub repository.