Politecnico di Torino

Politecnico di Torino

Cybersecurity

A.a. 2024/2025

Graduation Session December 2025

Predicting User Quality of Experience to Identify Network Issues in Live Streaming services: An ISP Perspective

Supervisors:

Marco Mellia

Danilo Giordano

Candidate:

Giuseppe Giordano

To my parents...
Whithout you none of this would have been possible.
Thank you for all your sacrifices
which have allowed me to achieve this dream

Abstract

In today's world, video streaming accounts for a significant share of Internet traffic. Internet Service Providers (ISPs) strive to provide these services while ensuring the best possible Quality of Experience (QoE) for users. One of the biggest challenges for ISPs is that, unlike content providers who can directly access users' QoE metrics, they can only derive them from network measurements such as throughput, latency and packet loss. This limitation is mainly due to end-to-end encryption, which limits the possibility of deep packet inspection, by enhancing user privacy while limiting insight into application-level metrics. The goal of this thesis is to develop methods that allow ISPs to accurately estimate user QoE relying on passive network traffic measurements that can be collected without breaking end-to-end encryption. This approach also makes it possible to identify problematic users with poor QoE while also allowing ISPs to determine whether service degradation originates from their infrastructure or is limited to specific users. To this end, I use machine learning techniques to predict the quality of video streaming and identify stall events based solely on passive network traffic measurements. The models are trained and evaluated on a dataset composed of traffic measurements describing video streaming content collected in a controlled environment and then tested on real-world traffic. In particular, this thesis analyses the streaming of DAZN videos, a popular sports streaming service. The real-world traffic was collected from an Italian Internet Service Provider. The personal data were properly anonymised to protect the privacy of the users. To identify QoE problems, the aim of this thesis is to predict changes in video resolution and stall events. For this purpose, starting from network logs that describe the data flows exchanged between a user and the content provider with passive measurements, I process millions of flows to extract the features required for the classifiers. Predictions are performed in 10-second time windows, which enables the identification of users with a high rate of resolution changes or rebuffering events, as well as users with consistently low video quality. Users with low QoE are further investigated by analysing additional network metrics such as RTT, the number of open flows and the amount of data exchanged to identify the main causes of QoE degradation. To visualise the results, a dashboard was developed that provides both aggregated and detailed views of QoE over time, while also allowing the examination of individual users and the associated network metrics. The results show that the proposed method can identify video resolution and stall events with high accuracy. After applying the developed method to the ISP data, I identified several problematic users where the ISP confirmed that they had connectivity issues.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In today's world, video streaming accounts for a significant share of Internet traffic. Internet Service Providers (ISPs) aims to deliver these services while ensuring the best possible Quality of Experience (QoE) for their users. Guarantee an optimal QoE means providing users a video streaming experience that is smooth, without interruptions, and with high video quality.
In the past years, the consumption of live streaming content has grown exponentially. Services such as Dazn, Amazon Prime and other live streaming platforms are now part of everyday digital life, generating a significant portion of Internet traffic. In particular, live streaming has gained increasing importance thanks to the growing popularity of real-time events or sports broadcasts. This evolution has introduced new challenges for ISPs, who must ensure stable connectivity and consistent performances. The Quality of Experience (QoE) is a measure of the delight of a customer's experiences with a service. In the context of video streaming, QoE is strongly influenced by factors such as video quality, playback stability, the presence and duration of rebuffering events. Understanding and quantifying QoE is fundamental to determine whether a user is experiencing a service in a satisfactory manner. From an ISP perspective, being able to monitor and predict user QoE is essential. It allows ISPs to proactively manage network resources, identify potential issues, and optimize the delivery of streaming content. The idea behind this thesis is to develop methods that allow ISPs to accurately estimate user QoE for DAZN live streaming events by relying exclusively on passive network traffic measurements, which can be collected without breaking end-to-end encryption. Such an approach can enable ISPs to monitor the user-perceived quality, identify potential network-related degradations, and improve the overall service quality offered to end users.

1.1 Motivation

End users expect to be able to enjoy video content smoothly and without interruptions, regardless of the device used or the network they are connected to. Internet Service Providers (ISPs), on the other hand, face the challenge of maintaining a high quality of service even in the presence of increasing network traffic. Monitoring and improving user Quality of Experience (QoE) is crucial for ISPs for both `business` and `technical` reasons. From a business perspective, monitoring and improving user Quality of Experience (QoE) is important because a poor viewing experience can lead to customer dissatisfaction with an increasing number of churn and damage to brand reputation. From a technical point of view, being able to assess QoE provides ISPs a deeper insights into network behavior and how it impacts end users. Traditional Quality of Service (QoS) metrics – such as throughput, latency and packet loss – are useful for evaluating the performance of the network infrastructure but do not always correlate with the actual experience perceived by the user. As a result, relying only on QoS measurements can lead to a misleading understanding of user satisfaction. However, accurately measuring QoE poses significant challenges, particularly in modern encrypted Internet environments. The widespread adoption of end-to-end encryption, such as HTTPS, has made it increasingly difficult for ISPs to access application-layer information. As a results, ISPs can no longer rely on Deep Packet Inspection (DPI) or other techniques to analyze traffic at the application level. Given these limitations, ISPs must rely only on passive network measurements to estimate user QoE so it is important to have tools and methods that allow them to extract useful information from network measurements to identify how the end user is experiencing the service. The idea is to leverage machine learning techniques to infer QoE from observable network traffic by extract indicators that can be correlated with user-perceived quality such as video quality or rebuffering events. These methods would allow ISPs to monitor QoE in a non-intrusive and privacy-preserving way, enabling continuous assessment of user-perceived quality across millions of live sessions.

1.2 Proposed Methodology

In this this thesis, we propose methods that allow ISPs to estimate user QoE in order to identify not only users with poor QoE but also the potential root causes of it. The idea is to extract from millions of network logs, those related to DAZN streaming traffic, in order to extract features needed as input for classifiers. These classifiers are designed to predict both the video quality with which the user is enjoying the service and the presence of stall events. The output of the model is then used to identify users with poor QoE and, in general, also the overall network

performances. Specifically the process is the following:

a) `DAZN flows identification`: live DAZN flows are identified from the network logs provided by the ISP and separated from the rest of the traffic.

b) `Feature extraction`: live DAZN network flows are divided into a 10-second windows from which statistical and temporal features are extracted. Those windows will be used as input for the classifiers.

c) `Adding user network info`: other usefull metrics are added to the 10-second quality windows, used to describe the user's bheavior during those 10-second interval, such as the number of opened connections or the amount of data downloaded/upload for both TCP and UDP traffic.

d) `Data classification and consolidation`: 10-second windows are used as input for two classifiers: one to estimate video quality and the other to identify stall events. Classifiers' result are then used also to coumpute the number of resolution changes and the number of stall event in a 5 minutes interval for each user.

To easily visualize the classification results, a dashboard has been developed. This dashboard allows ISPs to monitor user QoE over time and identify potential issues affecting the streaming experience. Thanks to this tool, ISPs can identify not only users experiencing poor QoE, but also the eventual precence of network problems that may be affecting the overall service quality.

1.3 Thesis Structure

The thesis is structured as follows:

- Chapter 2 provides an overview of the background and related work in the field of QoE estimation for video streaming services.
- Chapter 3 presents the proposed methodology for estimating user QoE from passive network measurements.
- Chapter 4 discusses the results obtained from the experiments conducted to evaluate the proposed approach.
- Chapter 5 concludes the thesis and outlines potential future work.

Chapter 2

Background

When it comes to evaluating network performance and user QoE, different types of measurement tools and protocols have been developed. However, they differ in how they work. Solutions can be divided into two primary categories: active and passive.

2.1 Passive Measurements

Passive measurement refers to the process of monitoring network traffic without injecting new traffic or affecting the existing one [1]. They rely on collecting packet traces or flow-level statistics directly from network interfaces, routers, or monitoring probes. Typical use cases include traffic classification, anomaly detection, and QoE estimation. In the context of video streaming, passive measurements can be used to infer application-level metrics (e.g., bitrate, stall events, video quality) by analyzing the characteristics of packet sequences and timing patterns. The main advantage of this approach lies in its scalability and transparency. However, it also presents challenges, such as the need for accurate labeling and difficulties in obtaining ground truth, especially with encrypted traffic. For these reasons in this thesis, we use tstat [2], a widely adopted passive network traffic analyzer, to collect traffic traces and extract relevant features for our QoE estimation models.

2.1.1 Tstat

`Tstat` is a tool used to process `.pcap` files[1] and reconstruct the characteristics of the internet traffic at the network, transport and application level. For the purpose of this thesis, we are interest in the ability of `Tstat` to reconstruct the TCP/UDP

[1]pcap (Packet Capture) is a common format for storing packet captures

flows, specifically we are interested on the recostruction of TCP flows to develop the classifier and on the reconstruction of both TCP and UDP flows to analyse the users' behavior. The output of tstat is a set of `.txt` files in a SSV format (Space Separated Values), each row corresponds to a different flow and each column is associated to a specific measure. If necessary the measures are provided for both directions of the flow (C2S and S2C). The useful files generated by tstat are:

a) `log_TCP_complete`: it provides a complete set of all the opened connection that ended correctly. A TCP connection is identified when the first SYN segment is observed and ends when the RST or FIN/ACK segments are observed. This log provides a wide range of volumetric and temporal metrics like the total amount of downloaded/uploaded bytes, the duration of the connection, the min, max and average rtt observed during the connection lifetime.

b) `log_UDP_complete`: reports every tracked UDP flow pair. Since UDP is a connectionless protocol an UDP flow pair is identified when the first UDP segment is observed for a UDP socket pair[2], and is ended when no packet has been observed in a specific time interval. In this case we can only have volumetric metrics because there are no ACKs or other control messages that can be used to compute temporal metrics.

c) `log_{TCP/UDP}_periodic`: these logs capture the temporal evolution of the connection by dividing it into bins of at most one second.

For both TCP and UDP flows tstat provide also, if able, the servername of the service used.

2.2 Active Measurements

A broad set of passive solutions exist that collect different statistics from network devices. These include network information such as traffic statistics like the amount of download or uploaded data, round trip time or dropped packets. However, while there is some relation to QoS, there is no possibility to use them alone to infer user's QoE. For this reason, active measurements are often used to complement passive measurements. Active measurements involve injecting synthetic traffic into the network to evaluate its performance. This method involves the active generation of test traffic or probes by dedicated tools or software. Active monitoring typically provides real-time insights into network availability, response times, and application

[2]a socket pair is the 4-tuple (src ip, dst ip, src prt, dst prt) that uniquely identify a network connection

performance. For the scope of this thesis, we use `streambot` [3] and `throttle` [4]-a JavaScript library-to perform active measurements. Specifically we use `streambot` to simulate video streaming sessions and collect application-level QoE metrics, while `throttle` is used to emulate different network conditions (e.g., bandwidth limitations) in order to simulate rebuffering events. These active measurements are crucial for obtaining ground truth data, which is essential for training and validating our QoE estimation models.

2.2.1 Streambot

`Strembot` is a lightweight JavaScript tool that automates browser actions to simulate user behavior on video streaming platforms. It performs packet captures using `Whireshark` and it also trace HTTP messages at the application layer. Streambot can be configured to simulate the streaming of a set of different streaming event for a specific duration. Thanks to its ability to interact with the web page DOM, Streambot can also retrieve the state of the video player. Once the execution is completed, Streambot outputs:

a) `log_net_complete`: a `.pcap` packet trace.

b) `log_har_complete`: a HTTP Archive (`.har`) file that contains all the HTTP messages exchanged during the streaming session in a json format.

c) `log_bot_complete`: a `.txt` file that contains the start time and the end time of the streaming session.

d) `log_reb_complete`: a `.txt` file that trace the state of the video player during the streaming session at each second (Table 2.1).

t_i	state
...	...
1752936816868	live
1752936817868	dead
1752936818868	live
...	...

Table 2.1: Example of `log_reb_complete`

2.3 Related Work

Before the adoption of traffic encryption, network QoE monitoring solutions were based on deep packet inspection (DPI) to extract information about video quality

parameters such as streamed resolution, buffer state, and bitrate [5], [6]. With the introduction of encryption, such approaches became unfeasible, this opened up new research related to QoE estimation based on the analysis of encrypted video traffic. Two main approaches have been proposed to address this challenge: `session-modeling-based` (SM) and `machine-learning-based` (ML) methods. SM solutions require knowledge of the underlying streaming protocol exploiting the session reconstruction to obtain QoE-related Key Performance Indicators (KPIs). One of this solution was eMIMIC [7]. The system reconstructs the chunk-based delivery sequence of a video session from packet traces, which is used estimate average bitrate, rebuffering ratio, bitrate switches, and startup time. However, adapting this system to work with QUIC traffic may reduce the accuracy of chunk detection, on which the system heavily depends. Other SM approaches instead, do not attempt to estimate QoE directly but rather focus on predicting the client buffer conditions [8]. However, given the problem dimensionality resulting from the differents type devices, platforms, streaming services, apps from which the content is accessed, finding analytical solutions for a wide range of cases becomes extremely complex. For such reasons, recent studies have shifted towards ML techniques for QoE/KPI estimation. This happened because this type of approaches are potentially more flexible and sustainable in long term scenarios [9], [10], [11], [12], [13]. In both SM and ML approaches, the ground-truth is required for model training. While this is relatively easy for some platforms, such as YouTube, it can be more challenging for others like DAZN or Prime due to the lack of publicly available APIs. In SM-based approaches, collecting a smaller dataset is typically sufficient to characterize the streaming protocol. In contrast, ML-based methods require large and diverse datasets—both in terms of application-layer performance and network conditions—to effectively train their models. The general structure of ML-based QoE estimation systems proposed is quite the same in each architecture (Figure 2.1). Once network traffic traces are collected, relevant traffic features are extracted and paired with corresponding QoE metrics derived from application-level data. The resulting dataset is then used to train ML models to estimate QoE/KPIs solely based on network traffic features.

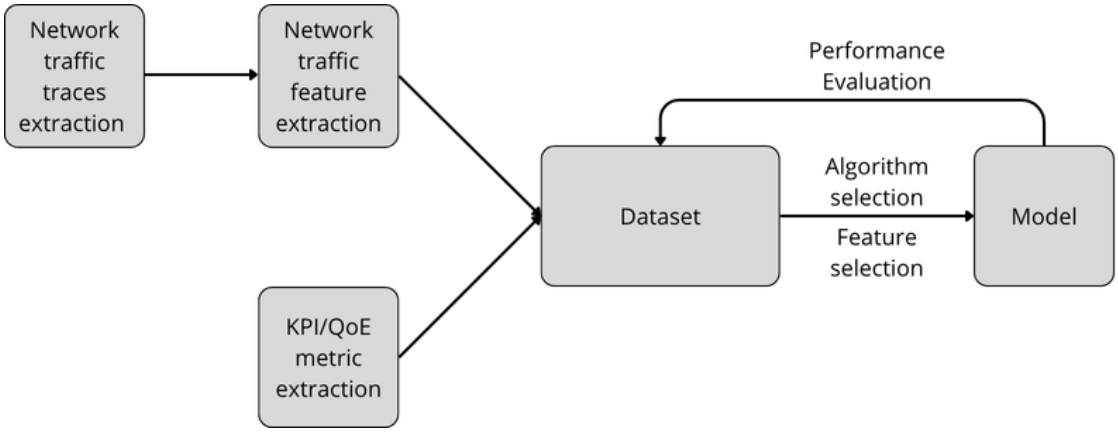


Figure 2.1: General architecture of ML-based QoE/KPI estimation systems

Chapter 3

Methodology

As mentioned in section 1.2, the pipeline proposed to compute the users' QoE and the overall network performance consists of five main steps:

a) `DAZN flows identification`: live DAZN flows are identified from the network logs provided by the ISP and separated from the rest of the traffic.

b) `Feature extraction`: live DAZN network flows are divided into a 10-second windows from which statistical and temporal features are extracted. Those windows will be used as input for the classifiers.

c) `Adding user network info`: other usefull metrics are added to the 10-second quality windows, used to describe the user's bheavior during those 10-second interval, such as the number of opened connections or the amount of data downloaded/upload for both TCP and UDP traffic.

d) `Data classification and consolidation`: 10-second windows are used as input for two classifiers: one to estimate video quality and the other to identify stall events. Classifiers' result are then used also to coumpute the number of resolution changes and the number of stall event in a 5 minutes interval for each user.

e) `Dashboard`: results will be displayed in a dashboard that allows to visualize the user QoE with different granularity (daily, hourly, per user).

In this section, all the steps carried out to develop the proposed pipeline are described. Specifically, the chapter is organized as follows:

a) Section 3.1 describes the analysis of the network logs provided by the ISP in order to identify DAZN live flows and consolidate the regular expression used for the filtering process.

b) Section 3.2 presents the development process of the classifier designed to identify stall events from network logs.

c) Section 3.3 describes the process of feature extraction and the retrieval of all useful information regarding user behavior from the network logs provided by the ISP, followed by the classification phase.

d) Section 3.4 illustrates the development of the dashboard that enables the visualization of the obtained results.

3.1 ISP data analysis

The first step in implementing the proposed pipeline is the analysis of the data provided by the ISP

3.1.1 ISP data structure

The network logs provided by the ISP are processed using `tstat`. The output of `tstat` is organized in a directory tree as shown in the following figure:

```
/ISP Folder
├── 2024/
├── 2025/
│   ├── 01-Jan/
│   ├── 02-Feb/
│   │   ├── 2025_02_01_00_24.out/
│   │   │   ├── log_tcp_complete.gz
│   │   │   ├── log_tcp_periodic.gz
│   │   │   ├── log_udp_complete.gz
│   │   │   ├── log_udp_periodic.gz
│   │   │   └── ...
│   │   ├── 2025_02_01_01_24.out/
│   │   ├── 2025_02_01_02_24.out/
│   │   └── .../
│   ├── .../
│   └── 12-Dec/
└── rrd/
```

Each `*.out` file contains the data collected in a 1 hour interval and is named as `YYYY_MM_DD_HH_mm.out` where `YYYY` is the year, `MM` is the month, `DD` is the day, `HH` is the hour and `mm` is the minutes when the logging started. In each `*.out` folder there are several compressed files, each one containing informations about the network flows observed during that hour.

3.1.2 Dazn flows Identification

Thanks to previous experiments [14] we know that there is the possibility to identify DAZN live flows by filtering them using the regular expression `(?=.*live)(?=.*dazn)`. The first step is to use this regular expression to filter DAZN live flows from the ISP data, in order to understand the transport protocol used (TCP or UDP) by DAZN to stream video content. To do this we analyse the log tcp and udp provided by the ISP in different days (those corresponding to some important Serie A matchday). For each day we:

- Select all the Dazn live flows from the log tcp and udp complete.
- Count the number of users that used tcp or udp and the total amount of data downloaded.

Doing this analysis on different days we can understand if DAZN uses a specific protocol to stream video content.

3.1.3 DAZN live flows analysis

Once the transport protocol used by DAZN to stream video content is identified, we can proceed with the analysis of DAZN live flows.

Regular expression consolidation

The second step is to consolidate the regular expression `(?=.*live)(?=.*dazn)` used to filter DAZN live flows. We want to understand if the regular expression is able to capture all the DAZN live flows or if there are other flows related to DAZN streaming that are not captured by this regular expression. In this case the idea is to concentrate the analysis on the time range when the event is played. In order to perform this analysis we select all the traffic from users that we know that are watching the event (we select these users by using our regular expression) from the ISP data. Once we have all the traffic from those users, we analyse all the opened connection, specifically we observe the server domain name to understand if there are other flows that can potentially be related to DAZN streaming.

Noise windows identification

Once all the Dazn live flows are identified, the next step is to identify and remove noise windows from the dataset. To do this we extract all the 10-second windows from all the DAZN live flows identified in the previous step and we analyse the distribution of the amount of data downloaded during those windows. The idea is to identify windows with a very low amount of data downloaded that can be

considered as noise windows (e.g., windows where the user has paused the video player or windows where the user has the dazn app opened in background). The identification of those windows is performed by setting a threshold on the amount of data downloaded during the window. All the windows with an amount of data downloaded below that threshold are considered as noise windows and removed from the dataset.

3.2 Stall events classifier

In order to estimate the user QoE, a classifier that is able to identify rebuffering events from network logs has been developed. The model was devoled using a supervised machine learning approach. It was developed by simulating DAZN streaming event in a controlled testbed, in order to be able to obtain the ground truth of rebuffering events.

3.2.1 Data collection

The data used to train, validate and test the models were collected using streambot. We collected several streaming sessions with differents bandwidth condition. Experiments with differents bendwidth values were performed to ensure that the classifier generalizes properly (i.e., it does not confuse low video quality caused by limited bandwidth with actual rebuffering events).

During the streaming session, we simulate rebuffering events and streambot trace the state of the video player every second in order to obtain the groundtruth for the model. Thanks to its ability to interact with the DOM, streambot is able to extract the HTML Media Element corresponding to the video player and read its properties. Specifically it is able to read the `readyState` properties of the HTML Media Element, when the video player is working correctly the `readyState` property is set to `4` (HAVE_ENOUGH_DATA), when a rebuffering event occurs the `readyState` property takes differents values depending on the amount of data available in the buffer.

Forcing Rebuffering event

Rebuffering events are forced by using `throttle` a JavaScript library that allows to limit the bandwidth. To collect data two differents modes of throttling were used:

- `Without randomicity`: during playback, at regular intervals of time, the bandwidth is throttled to a specific value for a specific duration.

- `With randomicity`: during playpack three differents values are randomly chosen:
  - `wait`: time before a rebuffering event
  - `reb`: duration of a rebuffering event
  - `throttle`: value of the bandwidth

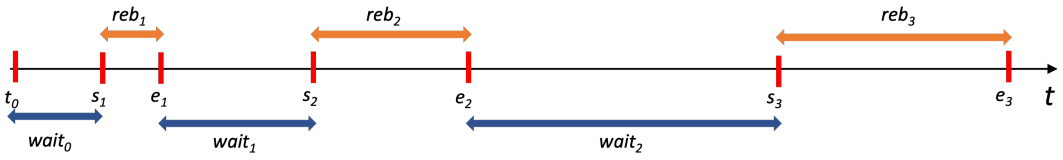


Figure 3.1: Example of rebuffering simulation with randomicity

When the connection is throttled and the buffer is emptied, a rebuffering event starts. When the connection is restored, the buffer starts to fill up and therefore streaming resumes.

3.2.2 Data Pre-processing

As said in Section 2.2.1 when streambot ends the streaming session, it generates several log files, one for each session:

- `log_net_complete`: a `.pcap` packet trace.
- `log_har_complete`: a HTTP Archive (`.har`) file that contains all the HTTP messages exchanged during the streaming session in a json format.
- `log_bot_complete`: a `.txt` file that contains the start time and the end time of the streaming session.
- `log_reb_complete`: a `.txt` file that trace the state of the video player during the streaming session at each second (Table 2.1).

All these files are processed in order to extract the features required for training the model. The following steps are carried out for each streaming session:

1. `log_net_complete processing`: using `tstat` the `.pcap` file is processed to obtain the logs complete and periodic of all the TCP and UDP connection opened during the streaming session.
2. `Flows filtering`: the regular expression `(?=.*live)(?=.*dazn)` is used to filter from the log_complete file all the flows related to the DAZN streaming.

3. `Bins filtering`: all the bins reletad to the DAZN streaming in the log_periodic are identified by using the socket pair that uniquely identify an opened connection, so for each opened connection in the log_complete all the corresponding bins in the log_periodic are selected.

4. `windows extraction`: considering t_{start} and t_{end} (from the `log_bot_complete` file) as the timestamp when the event start and end, all bins are divided using a sliding-window technique with dimension `winsize` and step `step_size`. Considering the interval $[t_i, t_j]$ as the time range covered by a window, all the bins are selected in the following way:

$$t_j \geq t_s \text{ and } t_i \leq t_e$$

For each window all the feature in Table 3.1 are extracted. For temporal features the inactivity time within the window and the statistics (max, min, avg, std) of the space occupied by each bins within the window are computed. For volumetric features the percentage of space occupied by the bins within the window is first computed as:

$$f = \frac{\max(t_s, t_i) - \min(t_j, t_e)}{t_e - t_s}$$

Then this factor is multiplied by the value of the feature. Finally for each window the sum of all contributions is computed:

$$\sum_{i=1}^{n} x_i * f_i$$

where n is the number of bins within the window, x_i is the value of the feature for the bin i and f_i is the factor computed for the bin i.
For each window, from the `log_reb_complete` file, the number of seconds spent in each state (playing, rebuffering) is computed.

feature	description
win_idle	Estimation of the period of inactivity within the window
avg_span	Avg duration of all bins in a windows
std_span	Std of the duretion of all bins in a windows
max_span	Max duration of all bins in a windows
min_span	Min duration of all bins in a windows
c_bytes_all	Bytes sent from the client
c_ack_cnt	Packets from client with the ACK flag set
c_ack_cnt_p	Packets from server with the ACK flag set with no payload
c_pkts_all	Packets sent from the client
c_pkts_data	Packets from client with payload
c_bytes_retx	Bytes retrasmitted from the client
s_bytes_all	Bytes sent from the server
s_ack_cnt	Packets from server with the ACK flag set
s_ack_cnt_p	Packets from server with the ACK flag set with no payload
s_pkts_all	Packets sent from the client
s_pkts_data	Packets from server with payload
s_bytes_retx	Bytes retrasmitted from the server

Table 3.1: List of features extracted from each window

3.2.3 Models training, validation and test

To develop the model we use the `scikit-learn` library in Python. It is a widely used library that provides several tools for the development of classification models. The model chosen for the classification task is a `Random Forest` classifier. This algorithm combines the output of multiple decision trees to reach a single result. Each decision tree is built using a different subset of the training data with a random subset of features, then each tree makes a prediction based on the input data. The results of all these trees are then combined to make a final prediction (usually by majority voting for classification tasks).

Data Processing

Once all the windows are extracted from all the streaming sessions, data are divided and combined into three datasets: training, validation and test set. The division in performed by maintaining all the windows extracted from a single streaming session in the same dataset. This approach is adopted because dividing all the samples randomly there is the possibility to give some «bias» to the model. For example we can divide a buffering event sending in the training set the end and

the start of it, instead the central part of the event can go on the test set. In this case the model can maybe understand that this specific sample is a buffering event because it «knows» that a buffering event start and end. When the data are divided two other step are performed:

- `Labelling`: each window is labelled as `rebuffering` if

  $$n_{stall} \geq \texttt{threshold}$$

  where n_{stall} is the number of seconds spent in rebuffering state during the window and `threshold` is a parameter. Otherwise the window is labelled as `playing`.

- `Standardization`: all the features are standardized using the `StandardScaler` from `scikit-learn`. This scaler standardize the features by removing the mean and scaling to unit variance. The standard score of a sample `x` is calculated as

  $$z = \frac{(x - u)}{s}$$

  where `u` is the mean of the training samples and `s` is the standard deviation of the training samples.

Models optimization and validation

The model is trained using the training set and then validated using the validation set. The validation set is used to evaluate the model during the training phase and to tune hyperparameters. To fine-tune the hyperparameters `GridSearch` is applied. The grid search systematically explores a predefined set of hyperparameter combinations to identify the best configuration for the model. The hyperparameters considered for the optimization are listed in Table 3.2. The chosen macro to evaluate the model performance during the validation phase is the `F1-score` for the `rebuffering` class. The best model obtained with the grid search is then evaluated using the test set.

Hyperparameter	Values
n_estimators	[5, 10, 15, ... 45]
max_depth	[6, 8, 10, 12, 14]
min_samples_leaf	[5, 10, 20, 40, 80, 120]
citerion	[`gini`, `entropy`]

Table 3.2: Hyperparameters used for the GridSearch.

3.3 ISP Data processing

In this section is described how the data provided by the ISP are processed in order to extract the features required for the classifications task and all the usefull information about the user behavior during the streaming. The data processing is divided into four main steps:

1. `Reading`: The ISP data are read and filtered.
2. `Make Win`: Data are processed to extract the features required for the classification task and other usefull information about the user behavior.
3. `Classify`: Windows extracted in the previous step are classified using two different classifiers: one for the video quality and one for the rebuffering events.
4. `Merge`: The classification results of the day are merged in a single file and some general statistics are computed.

At the end of this process, we will have a set of file that describe the users behavior during the streaming session.

3.3.1 Reading

The first step of data processing is to read and filter the data provided by the ISP. Given the large volume of data to be processed, the reading and filtering process is performed by using `PySpark` [1]. Spark speeds up computation by caching data in memory, allowing it to be reused across multiple parallel operations. The reading module allows to select a specific day and a time interval within that day and perform the following steps:

- `Select files`: from the ISP data are selected all the `log_tcp` and `log_udp` files where
$$start - 1 \leq HH \leq end + 1$$
where `start` and `end` are the start and end hour of the selected time interval.
- `Read complete and periodic`: all the selected `log_tcp_complete`, `log_tcp_periodic`, `log_udp_complete` and `log_udp_periodic` files are read and stored in a Spark DataFrame. For each entry of the DataFrame, all IPv6 addresses are converted to IPv4 addresses.

[1] PySpark: PythonAPI for Apache Spark, an open-source analytics engine designed for Big Data workloads

- `Select live DAZN flows and bins`: using the regular expression `(?=.*live)(?=.*dazn)` all the flows related to DAZN streaming are filtered and using the socket pair that uniquely identify an opened connection all the corresponding bins are selected.

- `Filter periodic TCP and UDP bins`: considering all the users with a live DAZN flow during the selected time interval, all the TCP and UDP bins not related to those users are removed. All the bins are also filtered in the following way:
  For each user `u` let $ts^{(u)}_{\text{min},dazn}$ and $ts^{(u)}_{\text{max},dazn}$ the minimum and maximum timestamp of the live DAZN flows related to that user.

$$\text{A bin } i \text{ is considered if } ts^{(i)}_{\text{u}} \in [\, ts^{(u)}_{\text{min},dazn},\, ts^{(u)}_{\text{max},dazn} \,]$$

$$\text{which is:} \quad ts^{(i)}_{\text{u}} \geq ts^{(u)}_{\text{min},dazn} \;\&\; ts^{(i)}_{\text{u}} \leq ts^{(u)}_{\text{max},dazn}.$$

- `Save filtered data`: All the filtered data are saved in a folder named `YYYY_MM_DD_xx_yy`

At the end of this process we will have the following output structure:

```
/output Folder
└── YYYY_MM_DD_xx_yy/
    ├── tcp_periodic/
    │   ├── ip1/
    │   │   ├── file1.csv
    │   │   └── .../
    │   ├── ip2/
    │   └── .../
    ├── udp_periodic/
    │   ├── ip1/
    │   │   ├── file1.csv
    │   │   └── .../
    │   ├── ip2/
    │   └── .../
    ├── live_complete.csv
    └── live_periodic.csv
```

Where `YYYY_MM_DD` is the selected day, `xx` is the start hour and `yy` is the end hour of the selected time interval [2]. In the `tcp_periodic` and `udp_periodic`

[2] the time interval goes from xx:00 to yy:59

directory there is a folder for each IP address that has at least one DAZN live flow during the selected time interval with all the TCP and UDP bins related to that IP address. In the `live_complete.csv` and `live_periodic.csv` files there are all the live DAZN flows and bins observed during the selected time interval.

3.3.2 Make win

The second step of data processing is to extract the features required for the classification task (Table 3.3) and other usefull information about the user behavior during the streaming. This process is performed by using `Pandas`. During this step two types of windows are extracted:

- `Quality windows`: 10-second non overlapping windows that contains all the features required as input for the video quality classifier and other usefull information about the user behavior during those 10 seconds.

- `Rebuffering windows`: 10-second overlapping window with a step of 5 second that contains all the features required as input for the rebuffering classifier.

For each user finded in the previous step, all the bins related to that user are read and processed in the same way described in Section 3.2.2 step 4. The difference is that in this case, for quality windows are extracted not only the features required for the classification task but also:

- The number of opened TCP and UDP connections during the window identified by the number of unique socket pairs.

- The total amount of data downloaded/uploaded during the window for both TCP and UDP

- C2S and S2C average Round Trip Time for live DAZN connections and TCP connections during the window.

Once all the windows are extracted for each user, they are saved in the same output folder of the previous step.

```
/output Folder
└── YYYY_MM_DD_xx_yy/
    ├── ...
    ├── quality_windows.csv
    └── rebuffering_windows.csv
```

feature	description
win_idle	Estimation of the period of inactivity within the window
avg_span	Avg duration of all bins in a windows
std_span	Std of the duretion of all bins in a windows
max_span	Max duration of all bins in a windows
min_span	Min duration of all bins in a windows
c_bytes_all	Bytes sent from the client
s_bytes_all	Bytes sent from the server

Table 3.3: Quality features

3.3.3 Classify

This module is the last step of data processing. It takes as input the quality and rebuffering windows extracted in the previous step and classifies them using two different classifiers: one for the video quality and one for the rebuffering events.

- `Quality classification`: each quality window is classified in one of the following classes: `0` for LQ video, `1` for MQ video, `2` for HQ video.
- `Rebuffering classification`: each rebuffering window is classified as: `1` if there is a rebuffering event within the window, `0` otherwise.

The output of the classifiers is concatenated to the input windows and saved in the same output folder of the previous steps. Once the classification phase is completed the results are merged in `time slot` of `time_interval` in order to compute the total number of resolution changes and the number of rebuffering events for each user during that interval. In this case we consider a `time slot` valid for a specific user if he has at least `N_WINDOWS` during that interval. The number of rebuffering events and resolution changes for each slots are computed in the following way:

- `Rebuffering events`: considering all the rebuffering windows classified during the time slot, the number of rebuffering events is computed as the sum of all the windows classified as `1`.
- `Resolution changes`: considering all the quality windows classified during the time slot, we count the number of changes in the predicted quality per time-slot (two consecutive prediction with different predicted quality), in case of a missing prediction, we consider the last available one.

At the end of this process both the windows labelled with the predicted quality and rebuffering events and the the number of resolution changes and rebuffering events per time slot are saved in the same output folder of the previous steps.

Window	0	1	2	3	4	5	...	28	29
Predicted Quality	HQ	HQ	MQ	-	HQ	MQ	...	HQ	HQ
Change	-	NO	Yes	-	Yes	Yes	...	NO	NO

Table 3.4: Example of computing resolution changes within a 5-minute time slot. In this example, three resolution changes occur.

```
/output Folder
└── YYYY_MM_DD_xx_yy/
    ├── ...
    ├── quality_windows_labeled.csv
    ├── rebuffering_windows_labeled.csv
    ├── quality_changes.csv
    └── rebuffering_events.csv
```

3.3.4 Merge

This module takes as input all the classification results obtained in the previous step and merges them in a single file. This module is used to merge the results of a single day and to compute some general statistics. By selecting a specific day this module performs the following steps:

- read the output folder of the previous step for the selected day (YYYY_MM_DD_xx_yy) and find the minimum `xx` and the maximum `yy` available.
- read all the files `quality_windows_labeled.csv`, `rebuffering_windows_labeled.csv` and concatenate them in a single file.
- read all the files `quality_changes.csv` and `rebuffering_events.csv` and merge them in a single file and, if necessary, fill the missing time slots.
- compute some general statistics for the selected day:
  - total number of users that streamed DAZN during the day.

    $$|\text{unique}(X)|$$

    where X is the list of ips in the `quality_windows_labeled.csv` file.
  - average number of active users who streamed DAZN per time slot.

    $$\frac{1}{\#slots}\sum_{j=1}^{\#slots}\left(\sum_{i=1}^{N}U_{i,j}\right)$$

    where N is the total number of users and $U_{i,j}$ is 1 if the user i is active during the slot j, 0 otherwise.

  - average number of quality changes and rebuffering event per slot.

$$\frac{1}{\#slots}\sum_{j=1}^{\#slots}\left(\sum_{i=1}^{N} Event_{i,j}\right)$$

    where N is the total number of users and $Event_{i,j}$ is the number of quality changes/rebuffering events for the user i during the slot j.

  - average number of users that experienced at least one rebuffering event/quality change per slot.

$$\frac{1}{\#slots}\sum_{j=1}^{\#slots}\left(\sum_{i=1}^{N} U_{i,j}\right)$$

    where N is the total number of users and $U_{i,j}$ is 1 if the user i experienced at least one rebuffering event/quality change during the slot j, 0 otherwise.

- save all the merged files in a new directory, those files will be used as input for the dashboard.

As example, if we have the following output folders from the previous step:

```
/output Folder
├── 2025_03_29_12_15/
├── 2025_03_29_16_19/
├── 2025_03_29_20_23/
└── ...
```

and we select the day `2025-03-29`, the module will read all the files in the three folders, merge them and save them in a new directory. The output will be divided in two parts:

```
/merged Folder
├── days/
│   └── 2025_03_29_12_23/
│       ├── isp_data.csv
│       ├── quality_windows_labeled.csv
│       ├── rebuffering_windows_labeled.csv
│       ├── quality_changes.csv
│       └── rebuffering_events.csv
└── general_info/
    └── 2025_03_29.json
```

If available, this module save also some user information provided by the ISP (e.g., the number of connected device, presence of saturation,...).

3.4 Dashboard development

The final step of the proposed pipeline consists in the development of an interactive dashboard designed to visualize the results obtained in the previous stages. The dashboard has been developed using `Streamlit`[3] All the data processed are stored in a remote server, in order to handle a large amount of data. To access these data from the dashboard a `SSH Tunnel` is used. The idea is to create a secure connection between the machine hosting the dashboard and the server and download only data required for the visualization to reduce the amount of data transferred and speed up the loading time of the dashboard.
The dashboard is organized in three different pages, each one with a specific purpose:

- `Home`: it is the main page of the dashboard, it summarize the overall network performance and users' Quality of Experience (QoE) during days. Specifically, it displays the results of the general statistics computed in the `Merge` step (Section 3.3.4) for each day available.
- `Daily analysis`: this page allows to visualize the network condition and users' Quality of Experience (QoE) during a specific day by showing the information aggregated per time slot.
- `User analysis`: this page allows to visualize the QoE during a specific day for a specific user with also some information about the user behavior during the streaming session and, if available, some information provided by the ISP.

When the dashboard is opened, an SSH connection is established with the server and all the `*.json` files are downloaded using `rsync`:

```
rsync -avz --progress user@server:.../general_info/*.json local_path
```

`rsync` [4] allows to synchronize files and directories between two locations over a secure shell (SSH) connection, it is used because it transfers only the differences between the source and the destination, reducing the amount of data transferred and speeding up the synchronization process. Once all the `*.json` files are downloaded the `Home` page is displayed and the available days are saved in the streamlit session state [5].

[3]Streamlit is a Python framework for building interactive web applications.

[4]rsync: is a command for transferring and synchronizing files between a computer and a remote server

[5]Session State is a way to share variables between reruns, for each user session. In addition session state also persists across apps inside a multipage app

This solution allow the user to open the dashboard and visualize the `Home` page in a few seconds, and then, if he wants to visualize the data of a specific day or user he can select it from a dropdown menu, in this case only the required data are downloaded from the server using:

```
rsync -avz --progress user@server:.../days_file/{root}* local_path
```

Where `root` is the selected day.

Chapter 4

Experimental Results

In this chapter the experimental results obtained are showed. Specifically in the first part we discuss the outcome obtained by studing the ISP data, while in the second part we present the results obtained by the classifiers developed in this thesis.

4.1 Live Dazn Flow Identification

We know that there is the possibility to identify DAZN live flows by filtering them using the regular expression `(?=.*live)(?=.*dazn)`. The idea is to use this regular expression to filter DAZN live flows from the ISP data, in order to understand the transport protocol used by DAZN to stream video content. The analysis is performed by selecting the traffic related to the last two serie A matchdays of the 2024/2025 season, specifically 5 differents days are analyzed:

- 18-05-2025
- 17-05-2025
- 23-05-2025
- 24-05-2025
- 25-05-2025

For each day we select all the traffic from 12:00 to 23:59, in order to cover all the time range when the matches are played. For each day all the live DAZN flows are selected and the number of users that used TCP or UDP to stream video content is computed. The results are reported in Table 4.1.

DAY	TCP Users	UDP users	TCP and UDP Users
18-05-2025	138	19	19
17-05-2025	377	32	32
23-05-2025	339	39	39
24-05-2025	197	66	66
24-05-2025	259	17	17

Table 4.1: TCP and UDP users in different days

Observing the results we can see that the majority of users use TCP to stream video content, while only a small percentage of users use UDP. We also observe that all the users that use UDP also open TCP connections to stream video content. We report in Table 4.2 the percentage of traffic downloaded using TCP or UDP, we can see that the amount of data downloaded using UDP is negligible compared to the amount of data downloaded using TCP.

DAY	% TCP Traffic	% UDP Traffic
18-05-2025	92	8
17-05-2025	90	10
23-05-2025	94	6
24-05-2025	91	9
24-05-2025	92	8

Table 4.2: TCP and UDP users in different days

Looking at the amount of data downloaded by users that use TCP and UDP in Figure 4.1 we can see that, for users who use both protocols, TCP is the predominantly used protocol in almost all cases.

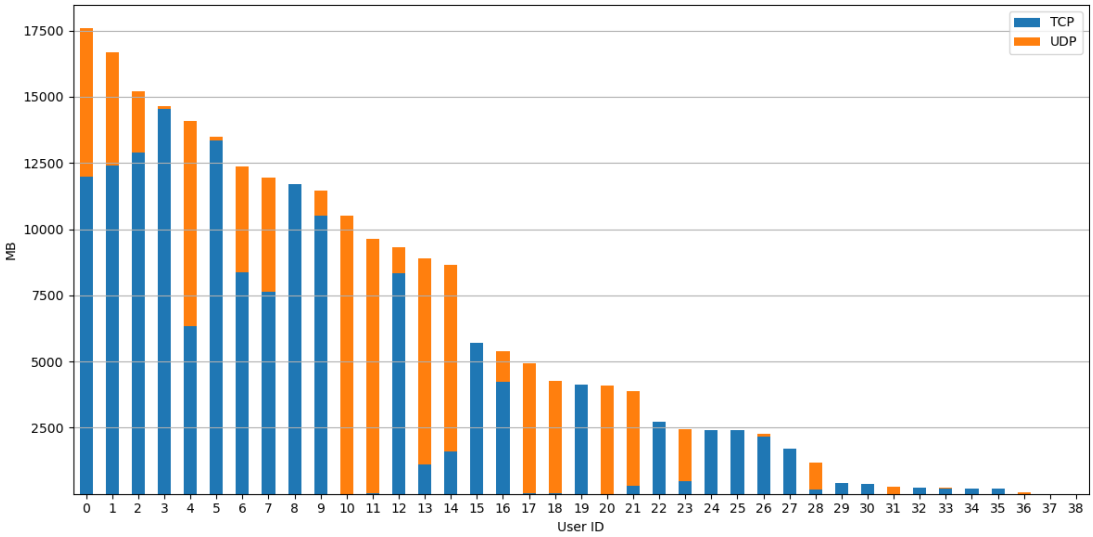


Figure 4.1: Amount of data downloaded by users that use TCP or UDP during the day 23-05-2025

Given that most users rely on TCP for video streaming, and that UDP traffic is negligible compared to TCP, we have decided to focus on TCP for the subsequent analyses.

Regular expression consolidation

To further validate the effectiveness of the regular expression `(?=.*live)(?=.*dazn)` in identifying DAZN live flows, we conduct an additional analysis. From the ISP data, we select from `log_tcp_complete` all opened TCP connection by users that watched DAZN live content (those users are identified using the regular expression). For each selected TCP connection, we select all the traffic that matches our regular expression, and we compute the amount of data downloaded during those connections. For the remaining traffic, we extract the Top Domain Level (TDL) and we compute the amount of data downloaded for each TDL. The figure 4.2 shows the amount of data downloaded for each TDL on May 23rd, 2025, between 20:00 and 23:00 — a period during which the matches Napoli–Cagliari and Como–Inter are played.

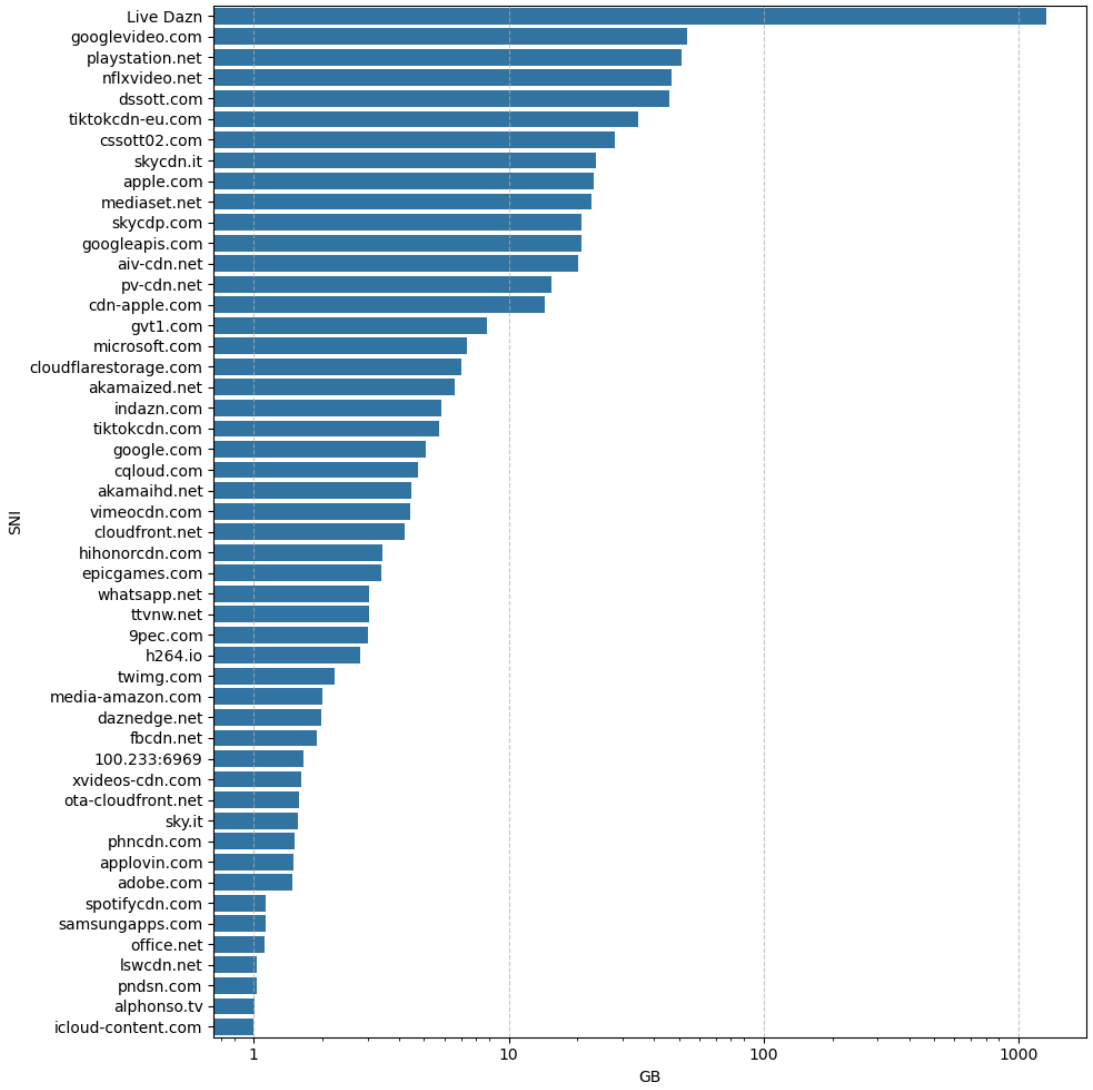


Figure 4.2: Top 50 server per downloaded data on May 23rd, 2025, between 20:00 and 23:00

Observing the results, we can see that no servers are downloading a significant amount of data other than those identified by our regular expression that can be associated with DAZN video streams, except for `indazn.com`, `daznedge.net`, and `akamaized.net`. Regarding `indazn.com` and `daznedge.net`, these domains are part of DAZN's content delivery infrastructure (CDN). By analyzing the full domain names rather than just the top-level domains, we can see that flows from these domains are captured by our regular expression when they include the word "live" (e.g., `dcb-deit-livedazn.daznedge.net`). Streams from these domains

that do not contain the word "live" download a negligible amount of data compared to those that do, indicating that they correspond to content other than live video streaming, which is necessary for the proper functioning of the DAZN service. Looking at the results for the `akamaized.net` domain in Figure 4.3, two main categories of flows can be identified: those related to DAZN, which contain the words "dazn" and "live" in the domain name, and those referring to other types of video streams, specifically Rai video streams.

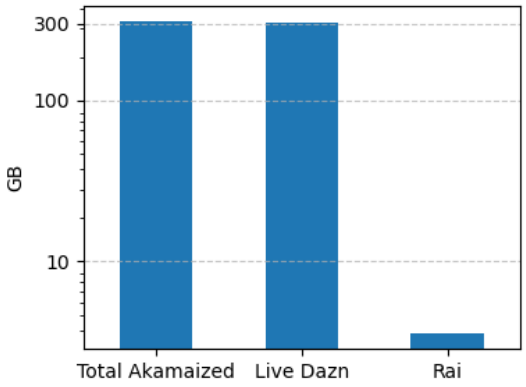


Figure 4.3: akamaized.net downloaded data for Dazn and Rai on May 23rd, 2025, between 20:00 and 23:00

In conclusion, we can confirm that the regular expression `(?=.*live)(?=.*dazn)` is effective in identifying DAZN live flows, as it captures the majority of data downloaded for live video streaming on DAZN.

Identify noise windows

Once all the DAZN live flows are identified, the next step is to identify and remove noise windows from the dataset. Once all the 10-second windows are extracted from all the DAZN live flows identified in the previous step, we observe the prediction of the quality classifier. Observing the results in Figure 4.4 we can see that 47% of the windows are predicted as `low quality`.

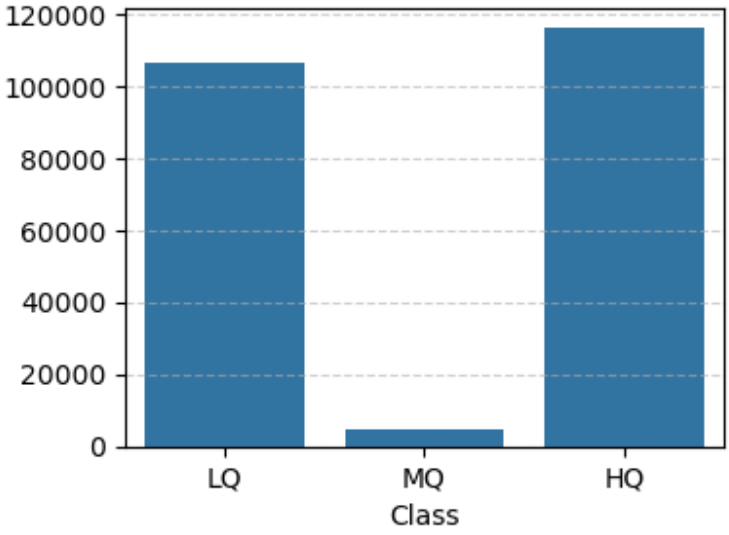


Figure 4.4: Quality model classification during a Serie A match

This is an unexpectedly high value, given that during a live streaming session the video quality should be generally good. This behavior can be explained by the presence of noise windows in the dataset. To identify those windows, we analyse the distribution of the amount of data downloaded during those windows, the results are reported in Figure 4.5.

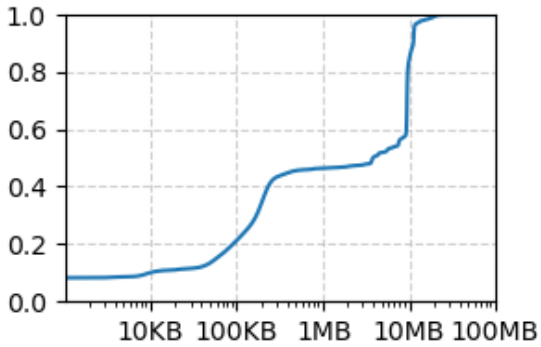


Figure 4.5: ECDF of the amount of downloaded data in a 10-seconds window

Observing the results, we can see that 50% of the 10-second windows have a downloaded data volume lower than 1 MB. Such a low amount of data is not compatible with video streaming, so it probably corresponds to non-video streaming activities, such as background traffic or idle periods. Since these windows cannot be associated with actual video streaming, we decided to introduce a threshold at

1MB to filter them. After removing those windows from the dataset, we observe that the percentage of windows classified as `low quality` drops to 1%, as shown in Figure 4.6.

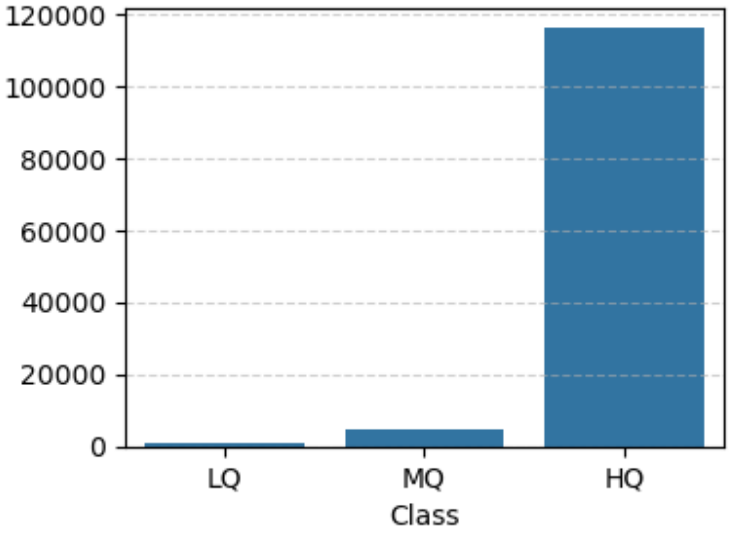


Figure 4.6: ECDF of the amount of downloaded data in a 10-seconds window with more than 1 MB of downloaded data

The obtained results appear to be more realistic, confirming the necessity of removing noisy windows in order to achieve more accurate and reliable outcomes.

4.2 Identify Rebuffering Events

4.2.1 Data Collection

To collect data we used `Streambot` to record 20-minutes playback sessions in order to build a dataset of realistic events. Since DAZN relies mainly on TCP as seen in Section 4.1, we decide to force the use of TCP during the streaming session by blocking UDP traffic on the machine running Streambot. Data were collected with two different network conditions:

- `Full bandwidth`: no limitation on the network bandwidth.
- `Limited bandwidth`: the network bandwidth was limited to 1500 Kbps for both download and upload.

Specifically, we collected 40 events – 20 minutes each:

- 20 with a network with full bandwidth

- 20 with a network with download/upload bandwidth limited to 1500 Kbps

Rebuffering Simulation

We decided to train the model in a supervised way, therefore we need to simulate rebuffering events during the streaming session in order to have the ground truth. As said in section 3.2.1 two differents modes were used to simulate rebuffering events:

- `Without randomicity`: during playback, every 120 seconds, the connection is throttled to 128 Kbps for a duration of 25 seconds.

- `With randomicity`: during playpack three differents values are randomly chosen:

  - `wait` [60-120] s: time before a rebuffering event
  - `reb` [15-25] s: duration of a rebuffering event
  - `throttle` [128,256,512] Kbps: value of the bandwidth

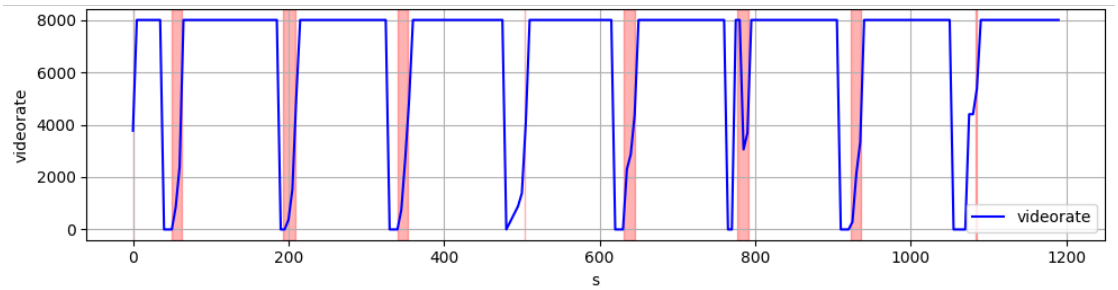


Figure 4.7: Evolution of the video rate during a streaming session with deterministic rebuffering events and full bandwidth

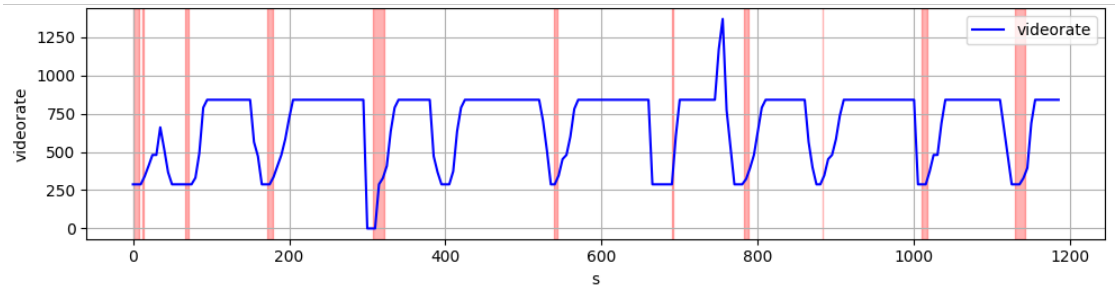


Figure 4.8: Evolution of the video rate during a streaming session with random rebuffering events and limited bandwidth (1500 Kbps)

The images 4.7 and 4.8 show the evolution of the video rate during a streaming session with deterministic and random rebuffering events respectively. In red are highlighted the rebuffering events.

4.2.2 Data Preprocessing, Partitioning and labelling

We apply the operational steps discussed in Section 3.2.2, setting the parameter `winsize` to 10 seconds and `step_size` to 5 seconds for both dataset. Once all the windows are extracted for each session we split them in training, validation and test set in the following way:

- `Training set`: 60% of the dataset
  - 12 streaming session at normal condition
  - 12 streaming session with the bandwidth limited to 1500 kbps
- `Validation set`: 20% of the dataset
  - 4 streaming session at normal condition
  - 4 streaming session with the bandwidth limited to 1500 kbps
- `Test set`: 20% of the dataset
  - 4 streaming session at normal condition
  - 4 streaming session with the bandwidth limited to 1500 kbps

In order to label the windows extracted we decide to put the `threashold` value to 2, it means that if in a window there are at least than 2 seconds of rebufferig, the window is labelled as `1` (rebuffering event), otherwise it is labelled as `0` (no rebuffering event).

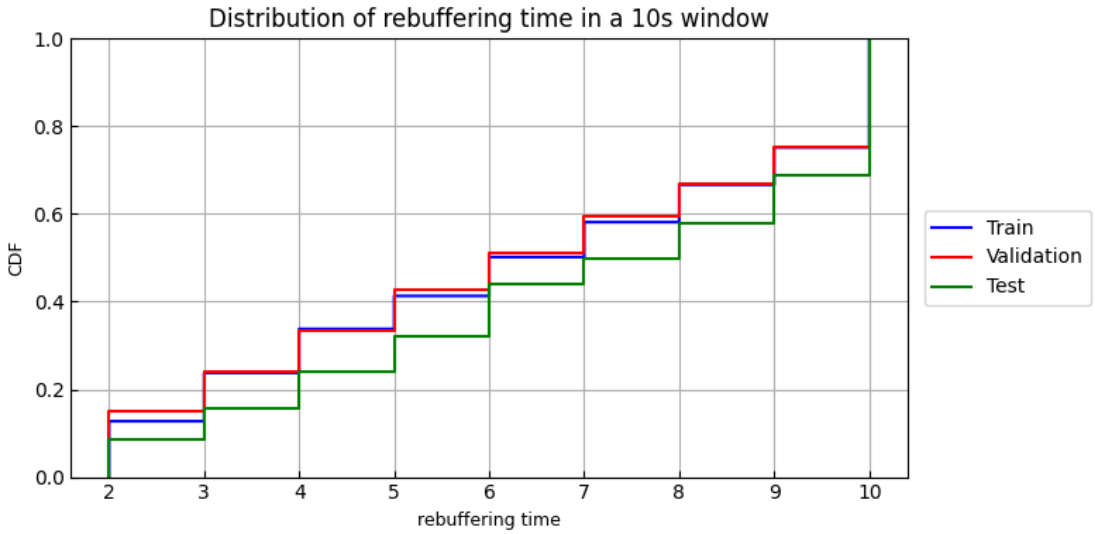


Figure 4.9: ECDF of the duration of rebuffering events in the dataset with deterministic rebuffering events

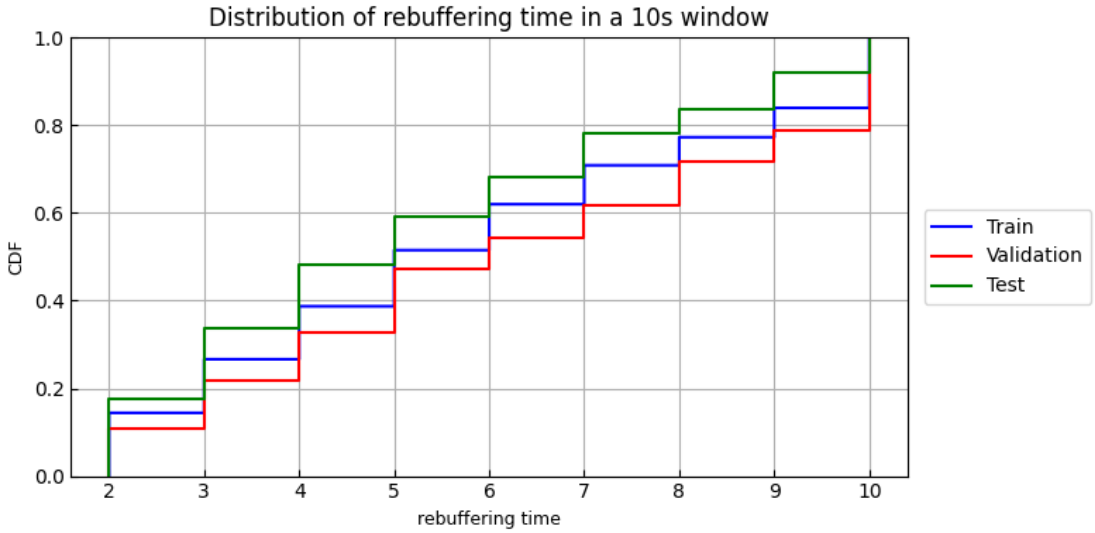


Figure 4.10: ECDF of the duration of rebuffering events in the dataset with random rebuffering events

At the end of the labelling process the data are standardized using the `StandardScaler` from `sklearn` library.

4.2.3 Models Training and Validation

The model was trained using both dataset, the one with deterministic rebuffering events and the one with random rebuffering events, in order to compare their performance and choose the most suitable one for the purpose. As discussed in Section 3.2.3 the model's hyperparameters were optimized using a `GridSearch` approach, the best model was chosen based on the `F1-score` metric for the rebufferig class evaluated on the validation set. The best hyperparameters for both dataset are reported in Table 4.3.

Hyperparameter	Deterministic model	Random model
n_estimators	35	30
max_depth	12	14
min_samples_leaf	5	10
criterion	entropy	gini

Table 4.3: Best hyperparameters found with GridSearch

The performance of both models on the validation set are reported in Table 4.4 and Table 4.5.

Class	Precision	Recall	f1-score
Normal	0.97	0.97	0.97
Rebuffering	0.80	0.81	0.80
accuracy			0.94
macro avg	0.88	0.89	0.89
weighted avg	0.94	0.94	0.94

Table 4.4: Deterministic model performance on the **validation set**

Class	Precision	Recall	F1-score
Normal	0.96	0.97	0.96
Rebuffering	0.83	0.76	0.79
accuracy			0.94
macro avg	0.89	0.87	0.88
weighted avg	0.94	0.94	0.94

Table 4.5: Random model performance on the **validation set**

The Figures 4.11 and 4.12 show the confusion matrix for both models evaluated on the training and on the validation set.

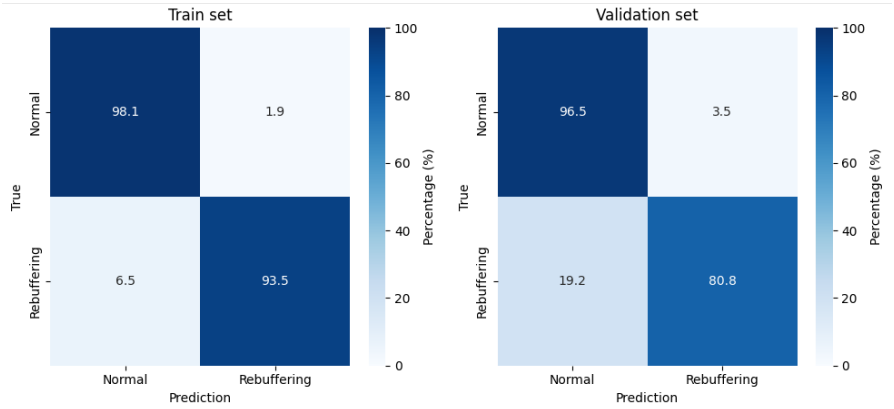


Figure 4.11: Confusion matrix of the dataset with deterministic rebuffering events for both training and validation set

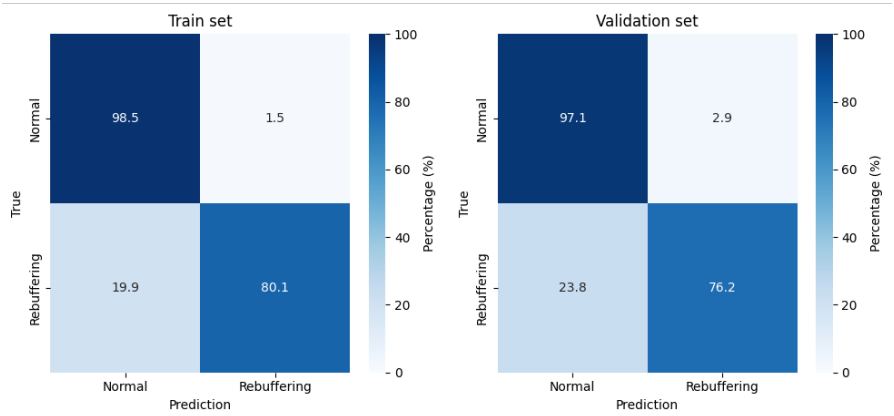


Figure 4.12: Confusion matrix of the dataset with random rebuffering events for both training and validation set

4.2.4 Model Testing

The last step of the process is to test the two models on unseen data, in order to evaluate their performance. On the table 4.6 and Table 4.7 are reported the performance of both models on the test set.

Class	Precision	Recall	F1-score
Normal	0.98	0.97	0.97
Rebuffering	0.84	0.89	0.86
accuracy			0.95
macro avg	0.91	0.93	0.92
weighted avg	0.95	0.95	0.95

Table 4.6: Deterministic model performance on the **test set**

Class	Precision	Recall	F1-score
Normal	0.96	0.97	0.96
Rebuffering	0.76	0.70	0.73
accuracy			0.94
macro avg	0.86	0.84	0.85
weighted avg	0.94	0.94	0.94

Table 4.7: Random model performance on the **test set**

In figure 4.13 and figure 4.14 are reported the confusion matrix for both models evaluated on the test set.

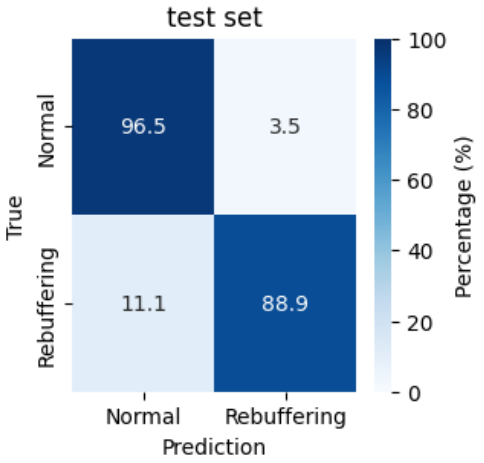


Figure 4.13: Confusion matrix of the model trained with deterministic rebuffering events for the **test set**

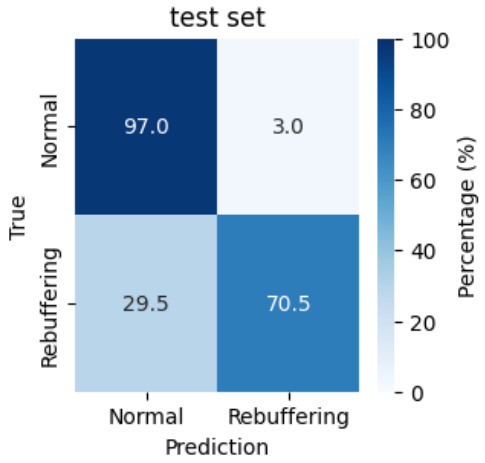


Figure 4.14: Confusion matrix of the model trained with random rebuffering events for the **test set**

Models Evaluation

Observing the results obtained on the test set on Section 4.2.4, we can see that the first model, trained with deterministic rebuffering events, achieves higher performance on the test set compared to the validation set. Specifically, the model achieves an F1-score for the rebufferig class of 0.86 on the test set, while it is 0.80 on the validation set. This behavior can be explained by the limited variability within the dataset, which might have led the training and test sets to share similar patterns, producing results that are more optimistic than in reality. When randomness is introduced in the data, the performance slightly drops. In this case, the model achieves an F1-score of 0.79 on the validation set and 0.73 on the test set for the rebuffering class. Even if the test performance are slightly worse than in the first model, in this case the model appear to be more robust. The introduction of randomicity ensures greater variability in data, providing a more realistic representation of real-world scenarios.
Both models are tested on real traffic data provided by the ISP, in order to evaluate their performance in a real-world scenario. The Figure 4.15 shows an example of the output obtained by the models during a three hours streaming session.

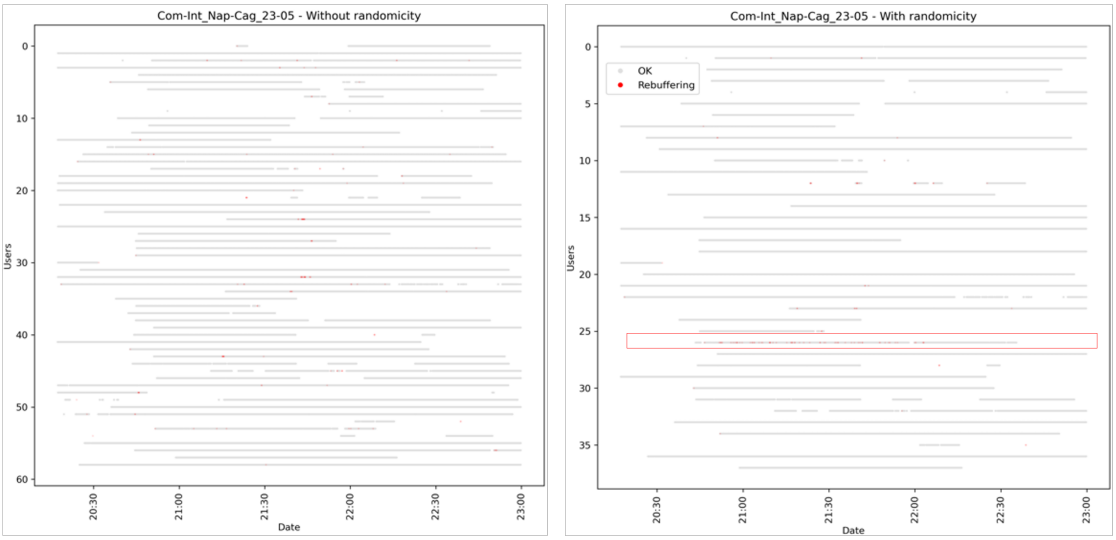


Figure 4.15: Output of both models during the last matchday of Serie A 2024/2025. In red is highlighted a user with an high number of rebuffering events.

Observing the output of both models out of 200 active users:

- The deterministic model identifies 59 users experiencing rebuffering events.
- The random model identifies 38 users experiencing rebuffering events.

Observing this results we can say that the model trained with random rebuffering events is more conservative in identifying rebuffering events, however it was able to identify a particlarly problematic user, experiencing a high number of rebuffering events during the streaming session. Once the problematic user is identified, the results are verified with the ISP, confirming the presence of issues during the streaming session.

4.3 Models Results on ISP Data

4.3.1 Insights from the weekend of March 29–31, 2025

To test the validity of the developed models, we first applied the video quality classification model to the ISP data collected during the 30th matchday of the 2024/2025 Serie A season, held over the weekend of March 29-31, 2025. First, 10-second windows were extracted from all live DAZN streams identified in the ISP data, retaining only those with a downloaded data volume greater than 1 MB, as discussed in Section 4.1, since these are compatible with an actual video stream. Additionally, all users who watched DAZN for less than 10 minutes were excluded from the analysis, as they are not sufficient to carry out a meaningful analysis.

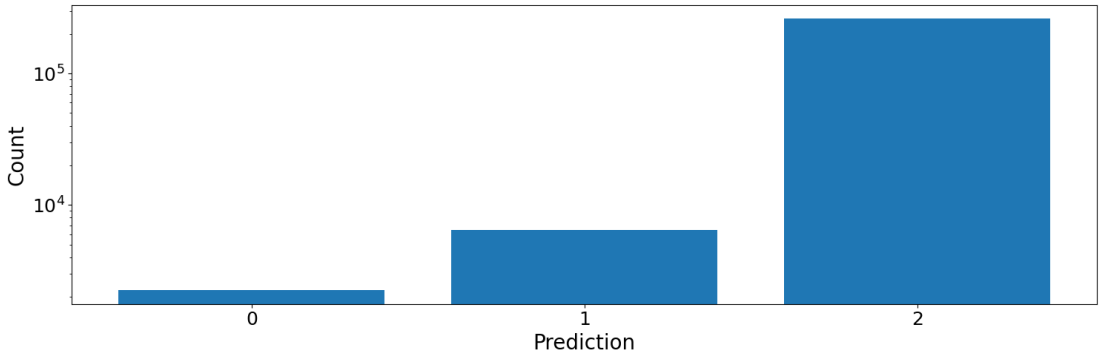


Figure 4.16: Prediction of the quality classifier during the Serie A matchday 29-31 March 2025: 0: 0.8%, 1: 2.4%, 2: 96.8%

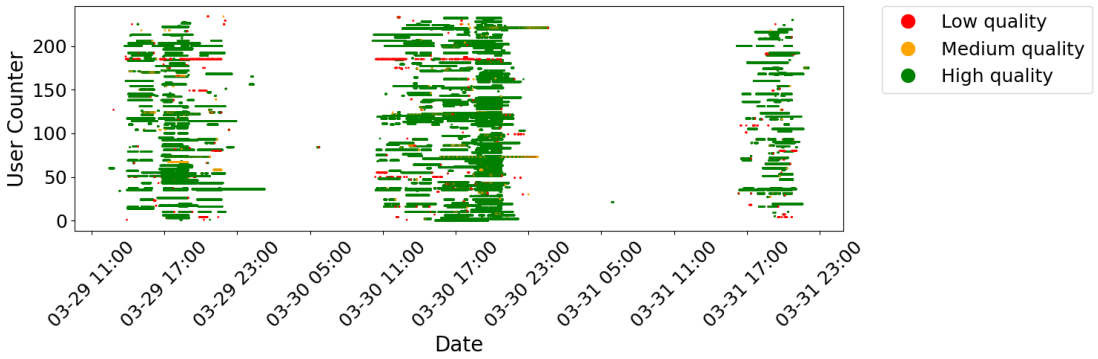


Figure 4.17: Distribution of the quality classifier predictions during the Serie A matchday 29-31 March 2025

Looking at the classifier's predictions over the entire weekend, we can see that, as expected, the prediction for most windows is 2 («High quality»). Almost all users show consistently high-quality prediction. To better understand the model's behavior, special attention was given to the most relevant match of the weekend, Napoli – Milan, played on 30 March 2025. The goal was to analyze the conditions leading to low-quality predictions and to assess the overall performance of the model. To this end, all 10-second time windows analyzed during the entire duration of the match were considered. The focus was in particular on the total volume of data downloaded, identified as the most relevant feature among those used by the model. To simplify analysis, the download volume has been rounded to the nearest integer value.

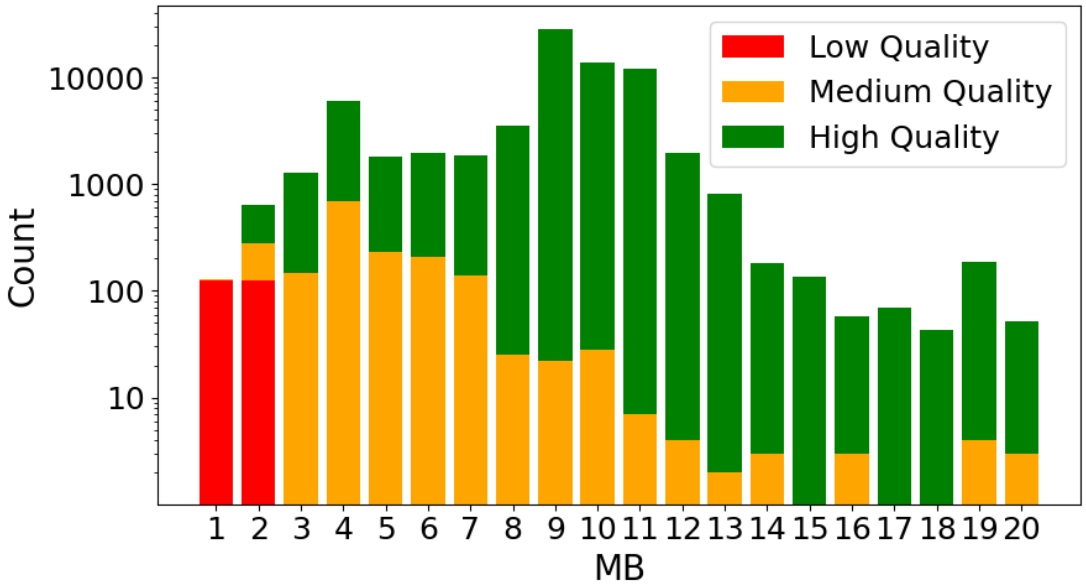


Figure 4.18: Windows prediction during Napoli - Milan match on March 30, 2025 based on downloaded data volume

Observing the results, it turned out that all `low quality` predictions correspond to downloaded data volumes of less than 3 MB. On the other hand, as the download volume increases, the probability that the video has a high resolution increases, a sign that the amount of data downloaded is linked to the quality of the video stream. However, these outcomes suggest that the behaviour of the model cannot be explained exclusively using the volume of the downloaded data. Although this variable represents a key indicator, the model uses a more complex combination of factors to make the predictions. In summary, low video quality is generally associated with a small volume of downloaded data, but the model's decision-making mechanism involves additional elements that enrich its predictive capacity. Let's now look at the behavior of the model applied to a specific user during the match.

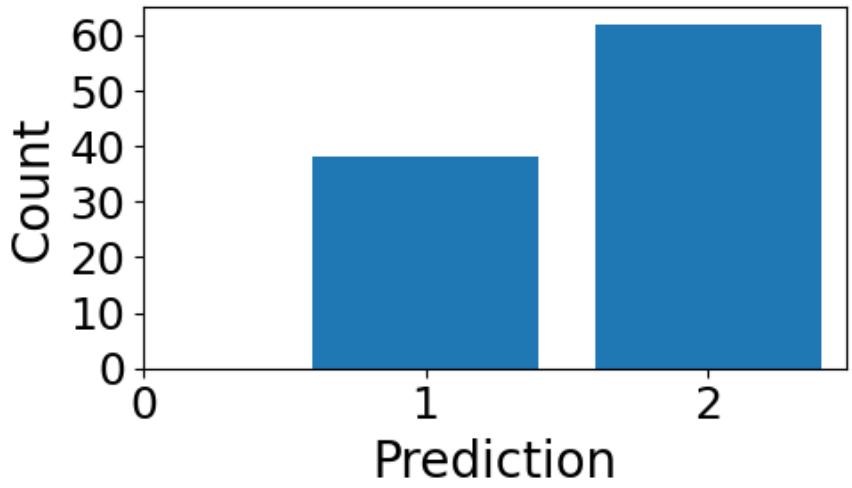


Figure 4.19: Prediction of the quality classifier during Napoli-Milan, user IPv4:ec60231010904e17badc954b7207bfdf: 0: 0%, 1: 38%, 2: 62%

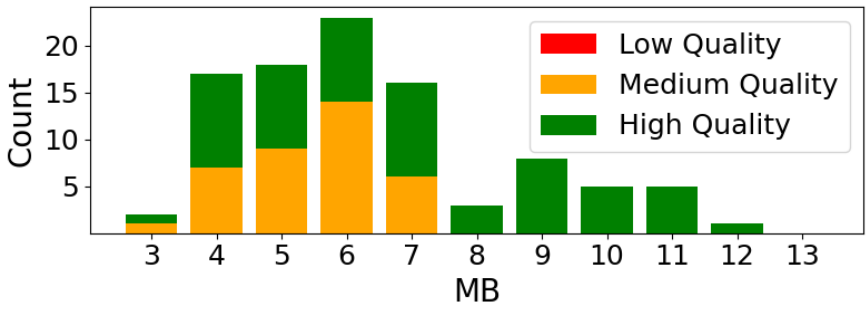


Figure 4.20: Windows prediction during Napoli - Milan match on March 30, 2025 for user IPv4:ec60231010904e17badc954b7207bfdf based on downloaded data volume

For the selected user, 38% of the windows were predicted as `medium quality`, and by observing the amount of downloaded data, it can be seen that these windows fall within a range of 3 to 7 MB. When analyzing the temporal evolution of the predictions and comparing it with the user's behavior over time, as shown in Figure 4.21, it can be observed that windows classified as `high quality` correspond to moments where the traffic is less intense. When the traffic becomes more intense, the network may become congested, which can lead to a decrease in video quality. By observing the user's traffic, we can notice a correlation between traffic spikes and a drop in the predicted quality.

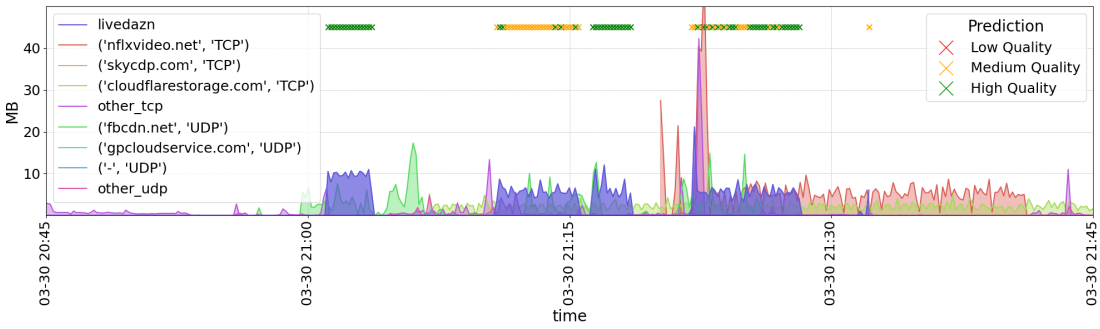


Figure 4.21: Time series of the downloaded data and quality prediction during Napoli-Milan match on March 30, 2025 for user IPv4:ec60231010904e17badc954b7207bfdf

This highlights the model's ability to capture the dynamic nature of video streaming quality, which can fluctuate based on network conditions and user behavior.

4.3.2 Matches analyses

We decided to extend the analysis to four three hour time interval corresponding to important matches of the 2024/2025 Serie A season.

Day Hour	Match	Active users
30/03/2025 20:00-23:00	Napoli-Milan	130
27/04/2025 14:15-17:15	Inter-Roma	113
18/05/2025 20:00-23:00	All matches (No Genoa-Atalanta)	232
23/05/2005 20:00-23:00	Napoli-Calgiari and Como-Inter	200

Table 4.8: Matches analyzed and number of active users during the streaming session

As described in Section 3.3 we extracted all the 10-seconds windows from all the live DAZN streams identified in the ISP data, retaining only those with a downloaded data volume greater than 1 MB. From all identified useres, we decided to focus the analysis on those who watched DAZN for at least 10 minutes during the analyzed time interval. In table 4.8 are reported the matches analyzed and the related active users during the streaming session. Considering all the active users during all the streaming sessions, we identify in total 334 differents active users. Observing the results obtained by the quality classifier during all the matches analyzed in Table 4.9, we can see that the vast majority of windows are classified as `high quality` as expected.

Quality	Windows	%
Low (LQ)	1975	0.5
Medium (MQ)	10269	2.6
High (HQ)	382702	96.9

Table 4.9: Quality classifier results during all the matches analyzed

Prediction	Windows	%
Ok	661594	99.8
Rebuffering	1025	0.2

Table 4.10: Rebuffering classifier results during all the matches analyzed

In table 4.10 are reported the results obtained by the rebuffering classifier during all the matches analyzed, we can see that only 0.2% of the windows are classified as rebuffering. As described in Section 3.3.3 we extract also the amount of quality changes experienced by each user during the streaming session by dividing the original 3-hour time interval into smaller intervals of 5 minutes. For each 5-minutes interval we consider an user active if he has at leat 15 10-seconds windows during that interval. To study the user QoE during the different matches we compute the average number of quality changes and the probability of having a quality change during an active slot as follows:

$$\text{Average user quality changes} = \frac{\sum Quality\_changes}{User\_active\_slots}$$

$$\text{Quality changes probability} = \frac{\sum Slot\_user\_quality\_changes}{Slot\_predicted\_windows}$$

Figures 4.22 and 4.23 show the ECDF of the average number of quality changes per user and the probability of having a quality change during an active slot respectively.

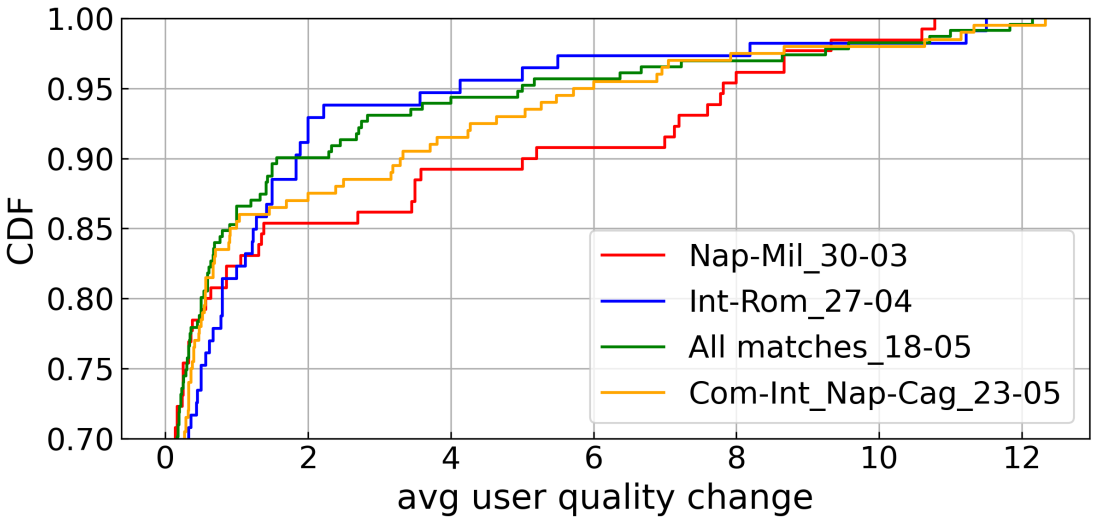


Figure 4.22: ECDF of the average number of quality changes per user

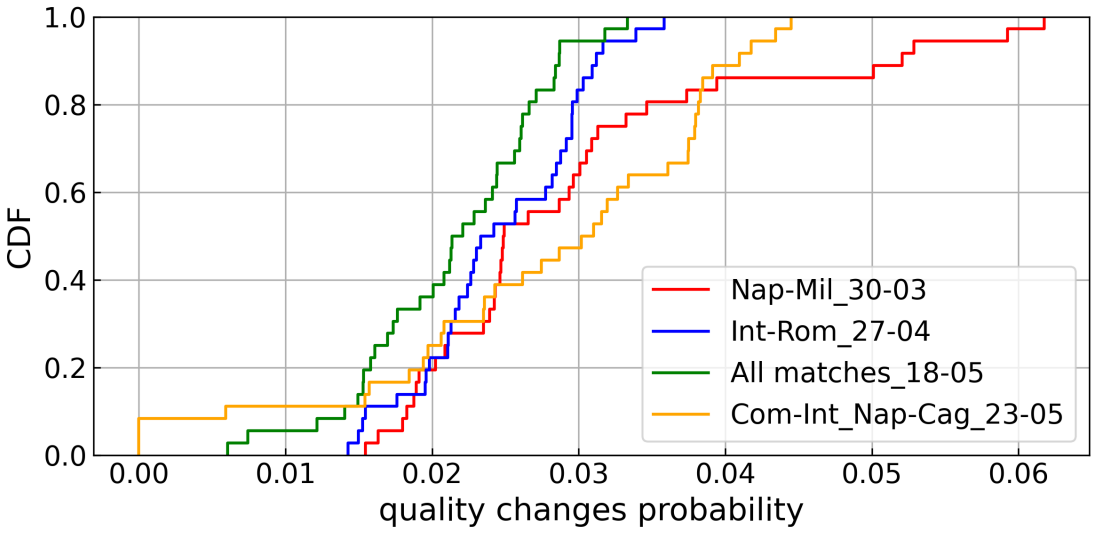


Figure 4.23: ECDF of the probability of having a quality change during an active slot

As we can see from the results All 4 3-hours time-slots show similar distribution with low probability in quality changes, however there are a small percentage of users that had more than 10 quality changes per slot. Observing Figure 4.24 the highest probability of quality changes is when we have a few active users. Specifically, we start the analysis about 30 to 45 minutes before the match starts.

During this period, the number of active users is relatively low and we have the higher probability of quality changes.

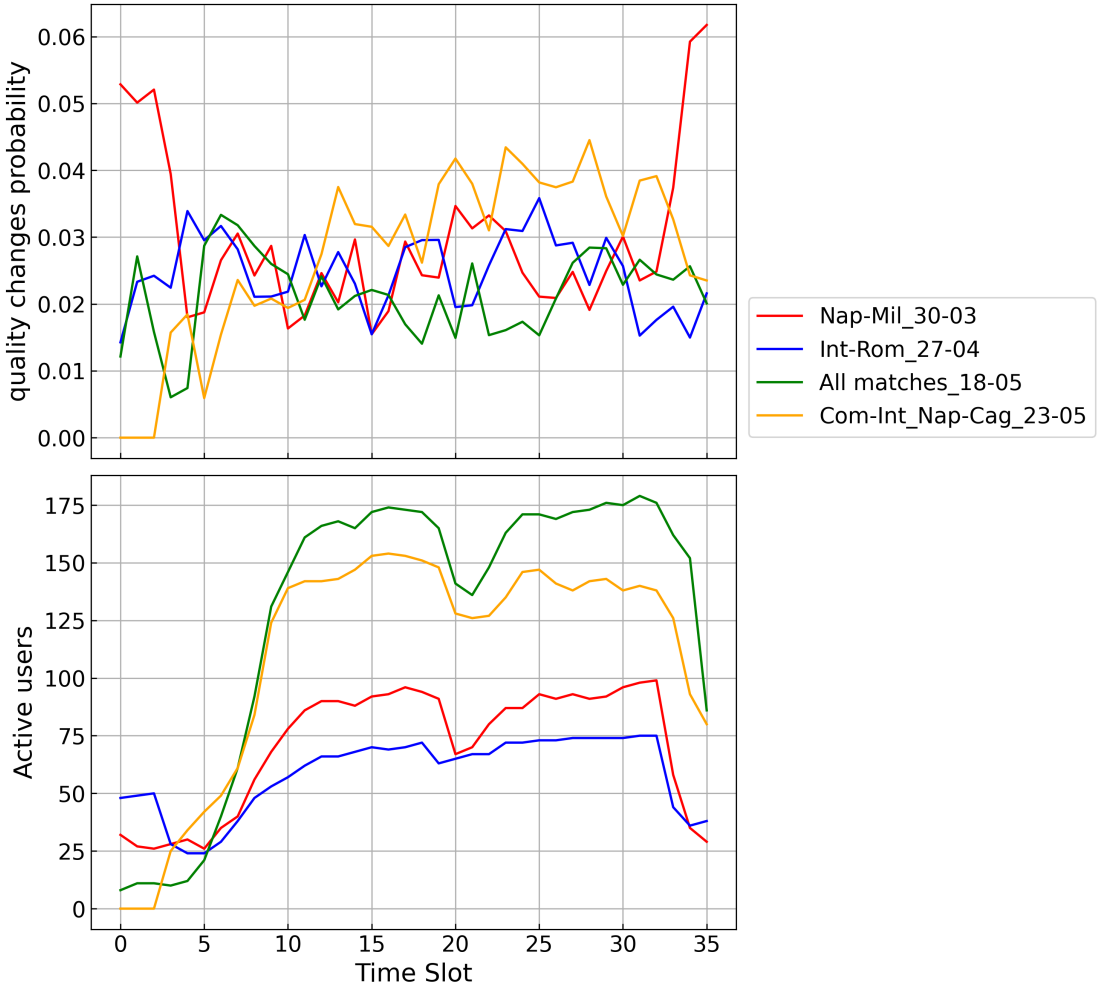


Figure 4.24: Quality changes probability during differents time slots compared to the number of active users

During the analysed matches, we observe that 81 of the 334 active users experienced at least one quality change during their streaming session. In Figures 4.25,4.26, are reported the number of quality changes experienced by each user during the matches of the last Serie A matchday. The results of the other matches analyzed are reported in the Appendix A. Reported results show only the users that experienced at least one quality change during the streaming session in at least one of the matches analyzed.

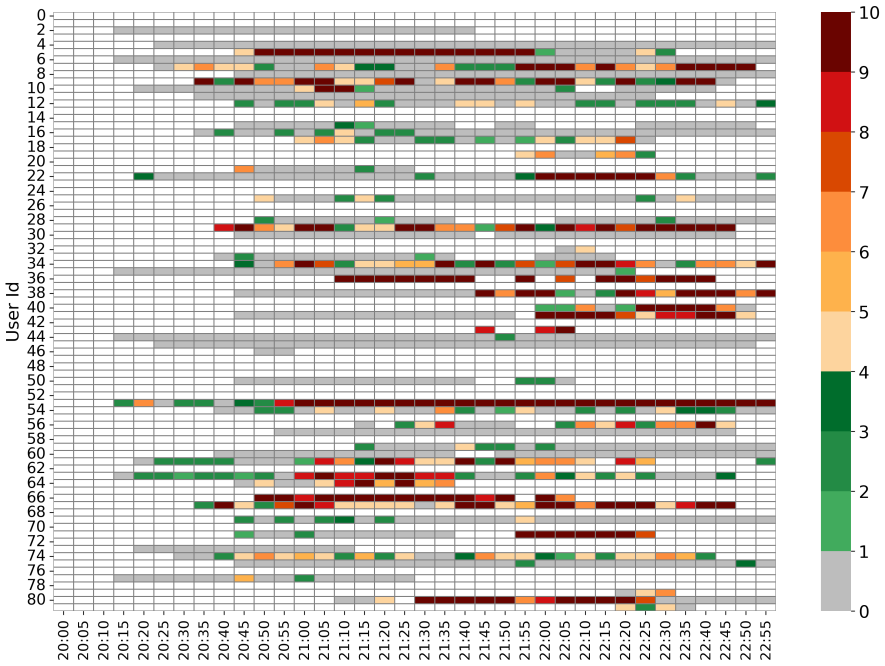


Figure 4.25: Quality changes experienced by each user that had at least one quality changees during the analysed matches on 23/05/2035 between 20:00 and 23:00

Figure 4.26: Quality prediction for each user that had at least one quality changees during the analysed matches on 23/05/2035 between 20:00 and 23:00

4.3.3 Users analyses

From the Matches analyzed in Section 4.3.2, we decided to focus the attention to 5 differents users that experienced different quality levels during the 23/05/2005 during Napoli-Calgiari and Como-Inter matches (is reported also the user id showed in Figures 4.25):

1. User with stable HQ predictions (**normal_user**)

   - IPv6:0e6b4d39a11b1f0840ef115feb044dae:622eb2f849ab5087901970ec55b109e3
   - `user id`: 45

2. User with frequent HQ $\rightleftarrows$ MQ quality switches (**high_medium**)

   - IPv4:5017f835dfd9668fcdb94a7e936fd541
   - `user id`: 29

3. User with frequent MQ $\rightleftarrows$ LQ quality switches (**medium_low**)

   - IPv6:c2389a8eedcee5f7a4663f825a70c1e2:ffe44a19a9b23e43c377776d4955d1a7
   - `user id`: 66

4. User with persistent LQ predictions (**always_low**)

   - IPv4:9fbd87b55c32c76b06bfdda3aafd5857
   - `user id`: 56

5. User with multiple rebuffering events (**rebuffering**)

   - IPv6:1076b330718df1e0b903715778cbfd1a:f1bd6d11dc3703eafeffefa2fa5b9659
   - `user id`: 5

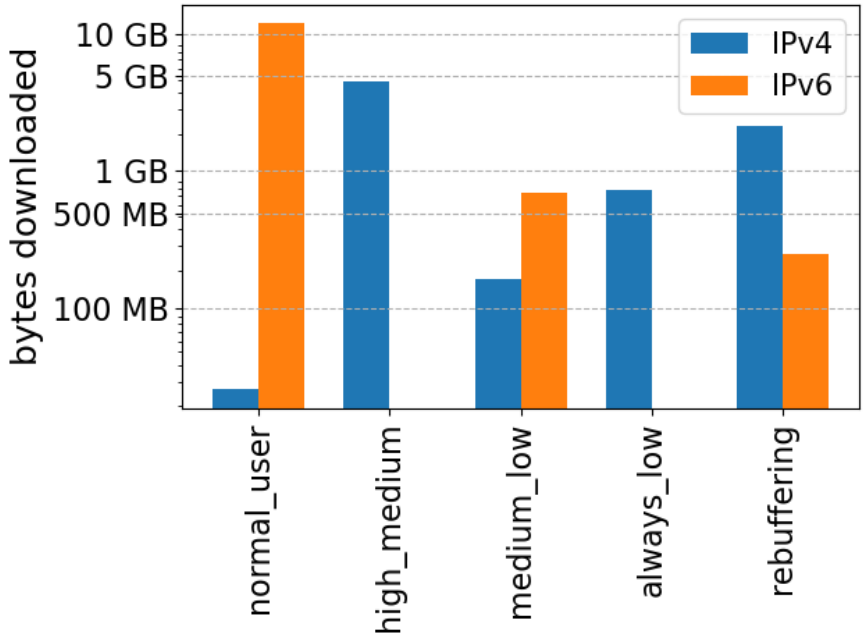


Figure 4.27: IPv4 and IPv6 download data from live DAZN servers during the last Serie A matchday of 2024/2025 season

We start the analysis by looking at the downloaded data from DAZN live servers during the matches, divided by IPv4 and IPv6 traffic as shown in Figure 4.27. We also look at the number of different connections opened by each user during the streaming session as shown in table 4.11.

User	IPv4	IPv6	Total
normal_user	5	7	12
high_medium	9	0	9
medium_low	325	649	974
always_low	55	0	55
rebuffering	6186	11	6197

Table 4.11: Number of connections per user with a live DAZN server

By examining the obtained results, users identified as `normal_user` and `high_medium` download a significant amount of data while maintaining a relatively small number of active flows. This behavior is typical of an efficient and stable use of the service, where data transfer occurs through a few persistent, high-volume connections. On the other hand, users `medium_low` and `rebuffering` display an anomalous behavior. Even if their traffic volume is not negligible, they generate a very high

number of active flows, indicating a fragmentation of the connections. To further investigate this behavior, we analyzed the characteristics of the flows opened by these users by examining them directly from the `log_tcp_complete`.

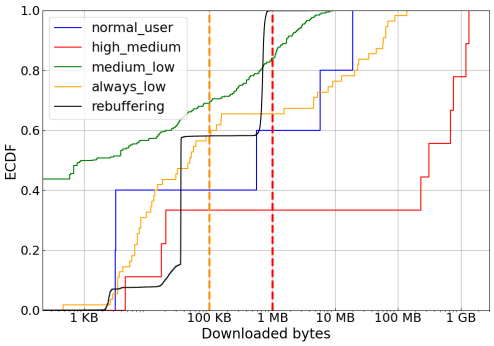


Figure 4.28: ECDF of downloaed data for **IPv4** flows

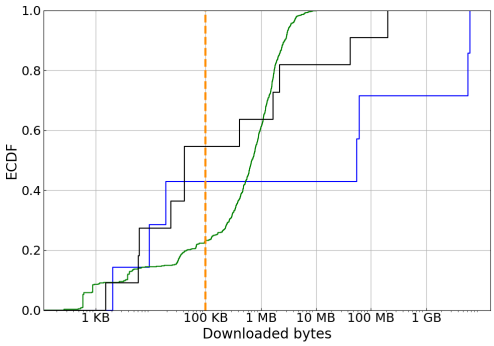


Figure 4.29: ECDF of downloaed data for **IPv6** flows

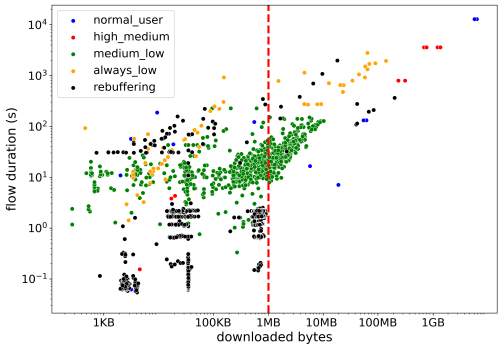


Figure 4.30: Scatterplot of **downloaed data** vs **flow duration**

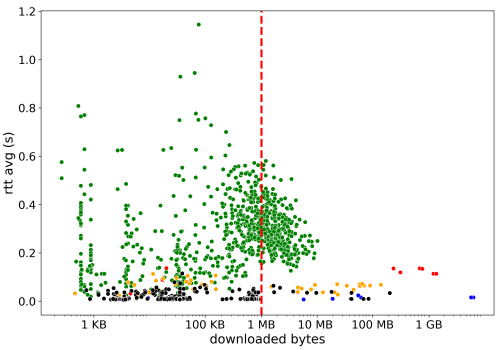


Figure 4.31: Scatterplot of **downloaed data** vs **rtt avg**

Figures 4.28 and 4.29 show that most of the flows opened by problematic users (`medium_low` and `rebuffering`) download only a small amount of data. More than 50% of the flows generated by these users transfer less than 100 KB per flow. Moreover, by examining the behavior of the `rebuffering` user, it can be observed that almost all flows download less than 1 MB of data. Figures 4.30 and 4.31 that shows respectively the relationship between downloaded data with flow duration and average RTT, highlight further anomalies. Specifically, most flows associated with the `rebuffering` user have a very short duration (less than 10 seconds), while those of the `medium_low` user exhibit a high average RTT (above 200 ms). These results indicate that the developed models are able to correctly identify problematic

users who display anomalous behaviors during streaming sessions. After consulting the ISP, it was also confirmed that the user experiencing numerous rebuffering events had encountered issues while watching the match and contacted them to have technical support.

After this analysis we decide to apply a filter to the `log_tcp_complete`, selecting only flows with more than 1 MB of downloaded data, in order to remove low-volume flows that may have been generated by activities other than video streaming. After applying this filter, we repeated the analysis.

User	IPv4	IPv6	# flows after filter	# flows before filter
normal_user	2	4	6	12
high_medium	6	0	6	9
medium_low	52	252	304	974
always_low	19	0	19	55
rebuffering	9	4	13	6197

Table 4.12: Number of connections per user with a live DAZN server after filtering flows with downloaded data $\leq$ 1MB

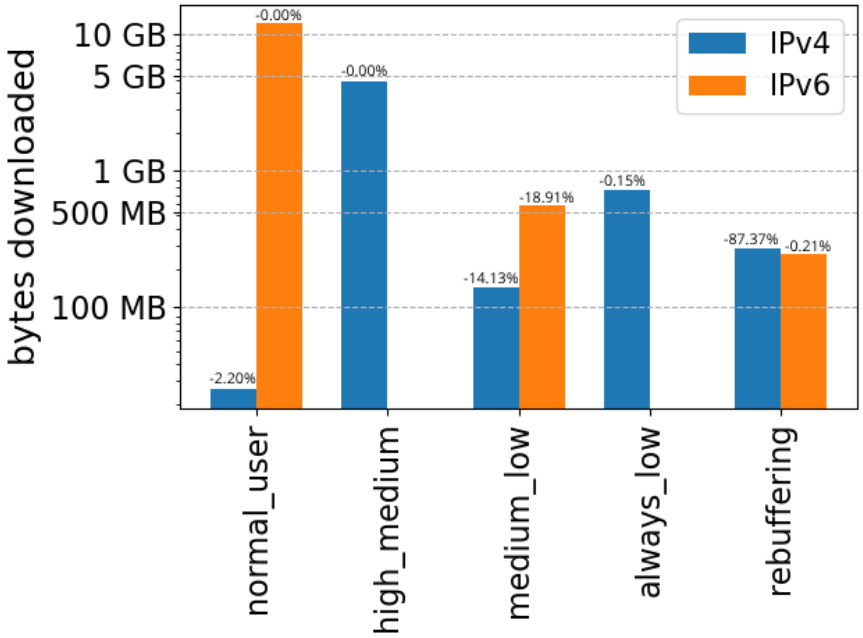


Figure 4.32: IPv4 and IPv6 download data from live DAZN servers after filtering flows with downloaded data $\leq$ 1MB

As shown in Table 4.12, as expected, the filtering process has a significant impact on the number of considered flows, particularly for the problematic users. Observing the amount of download data for IPv4 and IPv6 flows in Figure 4.32 we can notice that the filtering process has an important effect also on the total volume of data downloaded from DAZN live servers by the problematic users, instead it has no effect for the normal users. Figure 4.32 shows that the filtering process significantly affects the total volume of data downloaded from DAZN live servers by problematic users (For the `rebuffering` user, IPv4 flows show an 87.37% decrease in the amount of downloaded data), instead it has a negligible impact on normal users. After applying the filter, the corresponding bins were selected from the `log_tcp_periodic` file, and 10-second windows were extracted again. Comparing the results before (Figure 4.33) and after filtering (Figure 4.34) shows that, for the `rebuffering` user, the number of valid windows is drastically reduced, causing the user to be identified as non-problematic by the model.

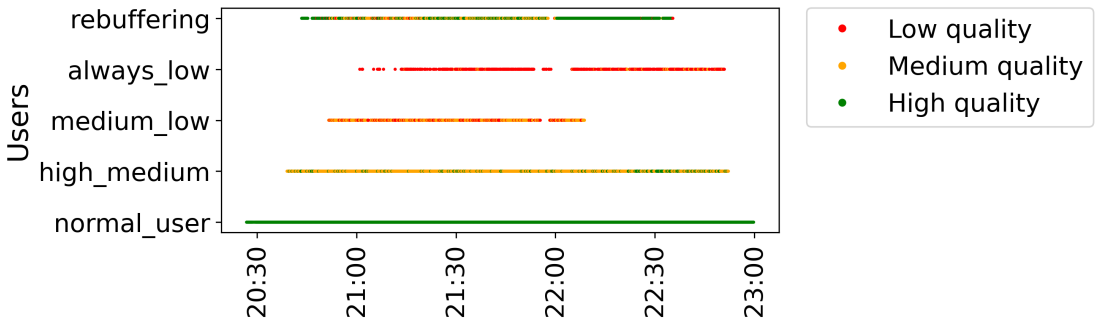


Figure 4.33

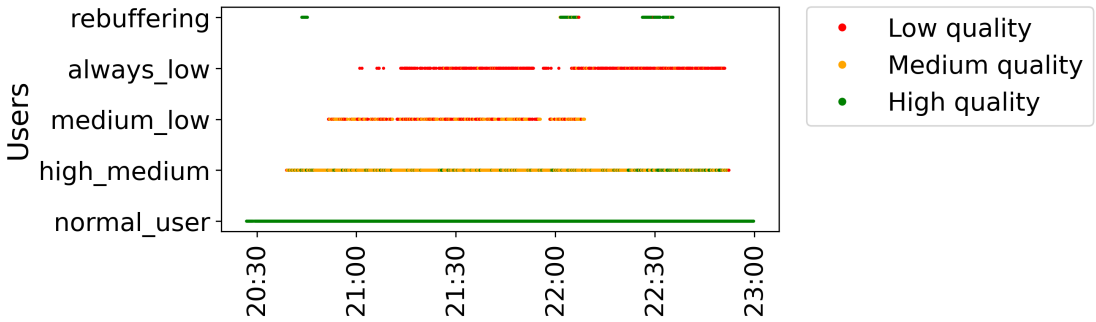


Figure 4.34

By observing the results of the rebuffering classifier in Figure 4.35, it is clear that applying the filter results in the loss of all windows where the model had predicted

a rebuffering event. Since the ISP confirmed that the user had indeed experienced issues during the match, this indicates that the filter is overly restrictive and leads to the loss of important information. Therefore, it was decided not to apply it when extracting windows from the ISP dataset.

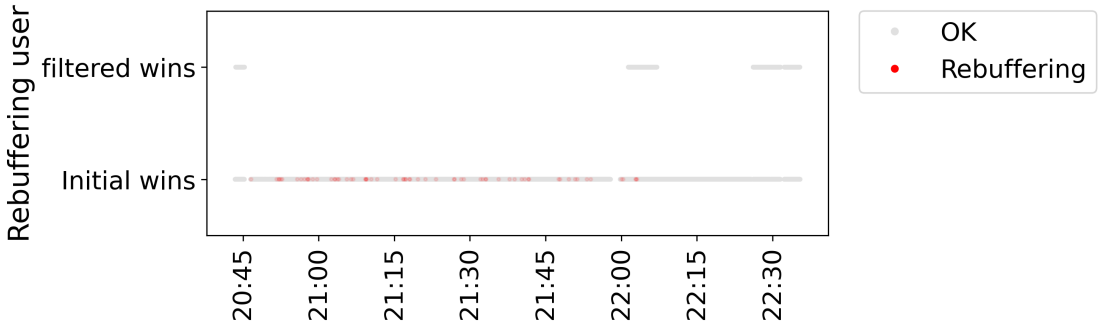


Figure 4.35

4.4 Dashboard Deployment

This section presents the final deployment of the dashboard developed in Section 3.4 showing its structure and functionalities. As said in Section 3.4, the dashboard is organized in three different pages: `Home` (Section 4.4.1), `Daily analysis` (Section 4.4.2) and `User analysis` (Section 4.4.3). The goal is to monitor quality changes and rebuffering events to identify potential network congestion or anomalous users.

4.4.1 Home

This page is the main page of the dashboard. It collects and displays aggregated analyses across multiple days. The goal is to provide an overview of the data, highlighting trends, variations, and key insights. Specifically this page displays the results of the general statistics computed in the `Merge` step in Section 3.3.4 for each day available. The Figures 4.36, 4.37 and 4.38 shows respectively:

- The total number of users that streamed DAZN during the day and the average number of active users who streamed DAZN in a 5-minutes time slot.
- The average numebr of quality changes and the average number of users experiencing at least 1 quality change in a 5-minutes time slot.
- The average numebr of rebuffering event and the average number of users experiencing at least 1 rebuffering event in a 5-minutes time slot.

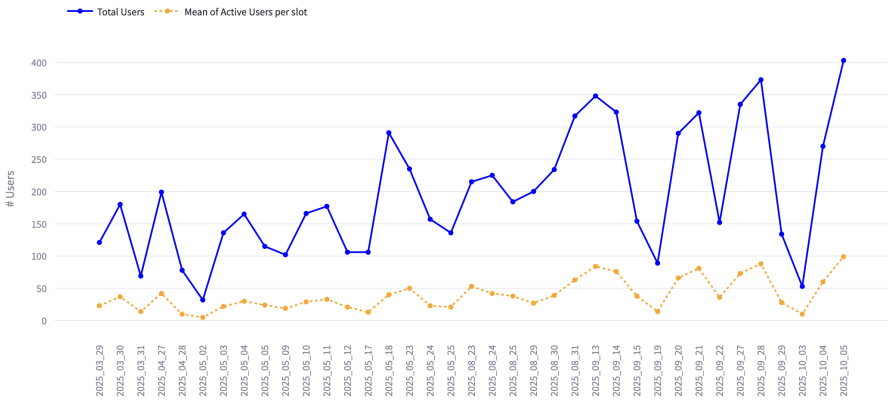


Figure 4.36: General statistics about the users streaming DAZN

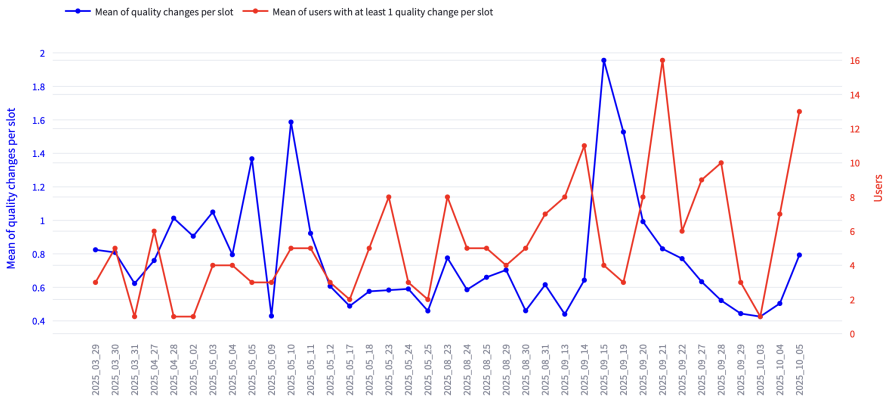


Figure 4.37: General statistics about the number of quality changes during the streaming sessions

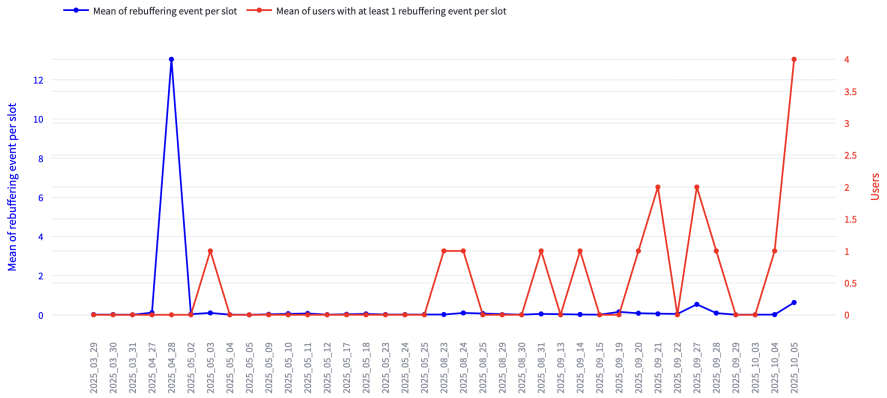


Figure 4.38: General statistics about the number of rebuffering during the streaming sessions

4.4.2 Daily analysis

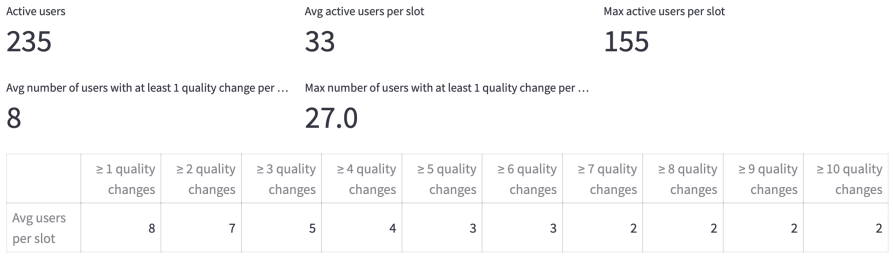
This page provides a detailed analysis of a specific day, allowing users to explore data and insights for that particular day. It is divided into three main section. After selecting a specific day and a time interval, the first section shows some aggregated statistics about users, changes in resolution and stall events during that time interval. The second and the third section show respectively information about resolution changes and rebuffering events with a 5-minutes granularity. Specifically we have:

1. `Aggregated statistics`:

   - `User Information`: Total number of users that streamed during the selected time interval, average and maximum number of users that simultaneously stremed during a 5-minutes time slot[1].

   - `Quality changes information`: Maximum and average number of users that experienced at least one resolution change and the average number of users that experienced up to 10 resolution changes during a 5-minutes time slot simultaneously.

[1]the original interval is divided into n different interval of 5-minutes



	≥ 1 quality changes	≥ 2 quality changes	≥ 3 quality changes	≥ 4 quality changes	≥ 5 quality changes	≥ 6 quality changes	≥ 7 quality changes	≥ 8 quality changes	≥ 9 quality changes	≥ 10 quality changes
Avg users per slot	8	7	5	4	3	3	2	2	2	2

Figure 4.39: Statistics on users and quality changes during May 23, 2025

- `Stall events information`: Total number of users that had up to ten rebuffering event, the maximum and average number of users that had at least one rebuffering event in a 5-minutes time slot simultaneously.

	≥ 1 rebuffering event	≥ 2 rebuffering event	≥ 3 rebuffering event	≥ 4 rebuffering event	≥ 5 rebuffering event	≥ 6 rebuffering event	≥ 7 rebuffering event	≥ 8 rebuffering event	≥ 9 rebuffering event	≥ 10 rebuffering event
Avg users per slot	17	9	7	5	4	2	2	1	1	1

Figure 4.40: Statistics on rebuffering events during May 23, 2025

- `Classifiers results`: for both classifiers (quality and rebuffering) the percentage of windows classified in each class considering all the users.

2. `Resolution changes`:

- `Heatmap`: It shows for each user and for each 5-minutes time slot the number of resolution changes experienced by that user in that time slot with the possibility to hilight bad users (users with at least `n` quality changes in more than 50% of the time slots where the users are active).

- `Time evolution`: for each 5-minutes time slot it shows the number of active users, the number of users that experienced at least one resolution change and the average number of resolution changes per user (ratio between the total number of quality changes and the number of active users).

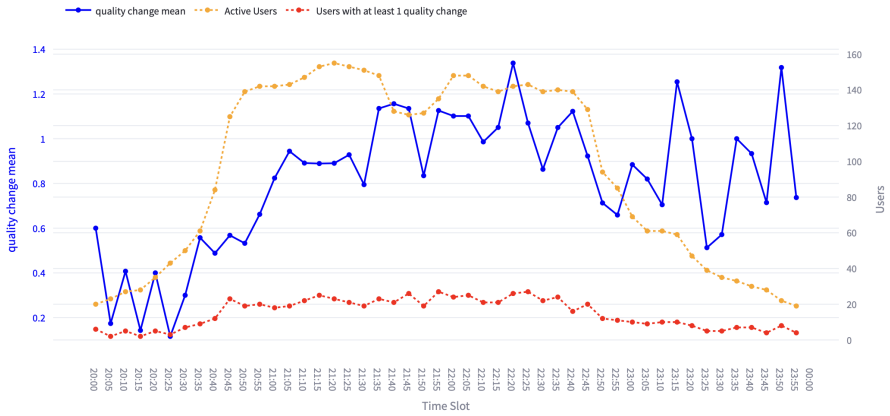


Figure 4.41: Quality changes and users information on May 23, 2025 for each 5-minutes time slot

- `Quality change probability`: The probability of having a resolution change in a 5-minutes time slot comuted as the ratio between the number of quality changes and the number of predicted windows in that time slot.

- `Users prediction`: there is the possibility to select one or more users and visualize for each of them all the prediction made by the **video quality classifier** during the selected time interval.

3. `Rebuffering events`:

- `Heatmap`: It shows for each user and for each 5-minutes time slot the number of stall events experienced by that user in that time slot with the possibility to hilight bad users (users with at least `n` stall events during the specified interval).

- `Time evolution`: for each 5-minutes time slot it shows the number of active users, the number of users that experienced at least one resolution change and the average number of rebuffering event per user (ratio between the total number of quality changes and the number of active users).

- `Users prediction`: there is the possibility to select one or more users and visualize for each of them all the prediction made by the **rebuffering classifier** during the selected time interval.

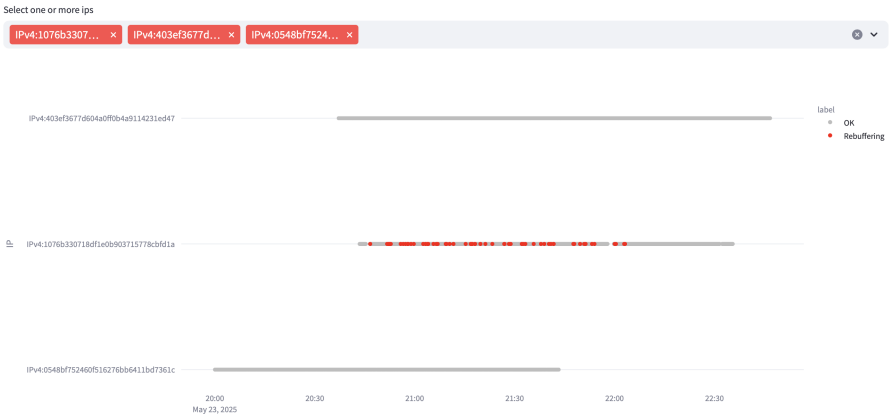


Figure 4.42: Prediction of the stall event classifier for three different users on May 23, 2025

4.4.3 User analysis

This page allows to select a specific user for a specific day and visualize some usefull information to understend the behavior of that user during that day. It is divided in two main sections:

1. `ISP data`: This section shows some information provided by the ISP for the selected user during that day.

Figure 4.43: Example of ISP data for a specific user

2. `Overview`: this section shows classifiers results by displaying the total amount of stall events and quality changes during the day. It also shows a timeline where all the prediction made by both classifiers are reported.

3. `Volumetric metrics`: this section shows some volumetric metrics about both TCP and UDP traffic generated by that user during that day. Specifically it

shows the total amount of downloaded/uploaded bytes, the number of opened connection and the average RTT in both directions (c2s/s2c) for TCP and UDP traffic.

Figure 4.44: downloaded/uploaded data for TCP and UDP traffic for a specific user

Chapter 5

Conclusion

This thesis has developed a way to estimate users' Quality of Experience (QoE) in live video streaming services, relying exclusively on passive network traffic measurements. The proposed method addresses the challenge posed by end-to-end encryption, which prevents Internet Service Providers (ISPs) from directly accessing application-level QoE parameters. The main objective was to design a system that was able to identify DAZN streaming flows and predicting users' QoE, with a particular focus on rebuffering events and the video quality experienced during content playback. Starting from the classifiers' results, the idea was to correlate their predictions with network metrics in order to infer possible causes of low QoE. The experiments conducted validated the effectiveness of the developed models. The rebuffering model, trained on controlled data, achieved an F1-score of 73% for class 1 (rebuffering events). The models were then successfully applied to real-world data provided by an ISP. In conclusion, this work demonstrates that ISPs can be equipped with effective tools for monitoring QoE in video streaming services. The analysis of passive network data, when properly processed through machine learning techniques, proved capable of accurately identifying users experiencing degraded service quality and correlating such degradation with measurable network parameters.

5.1 Future Work

This work presents some limitations that open several directions for future research. First, the developed models were trained on DAZN live streaming flows; therefore, their generalization to different streaming services and networks with varying characteristics remains to be evaluated. An ongoing effort aims to extend the analysis to other services, such as Amazon Prime Video, which employs different network protocols (e.g., SYE). In particular, current research focuses on identifying

SYE flows and correlating them with QoE metrics to perform an analysis similar to the one presented in this work. Further work is also being carried out to enhance the dashboard functionalities by automating the anomaly detection process. This would enable the system to automatically identify users experiencing low QoE and provide insights into possible root causes, thus improving the overall monitoring of service quality. Additional developments could include the design of new models capable of predicting rebuffering events or video quality, moving from traditional machine learning methods to deep learning approaches such as Recurrent Neural Networks (RNNs), which are particularly suitable for time-series analysis.

Appendix A

Matches analysed

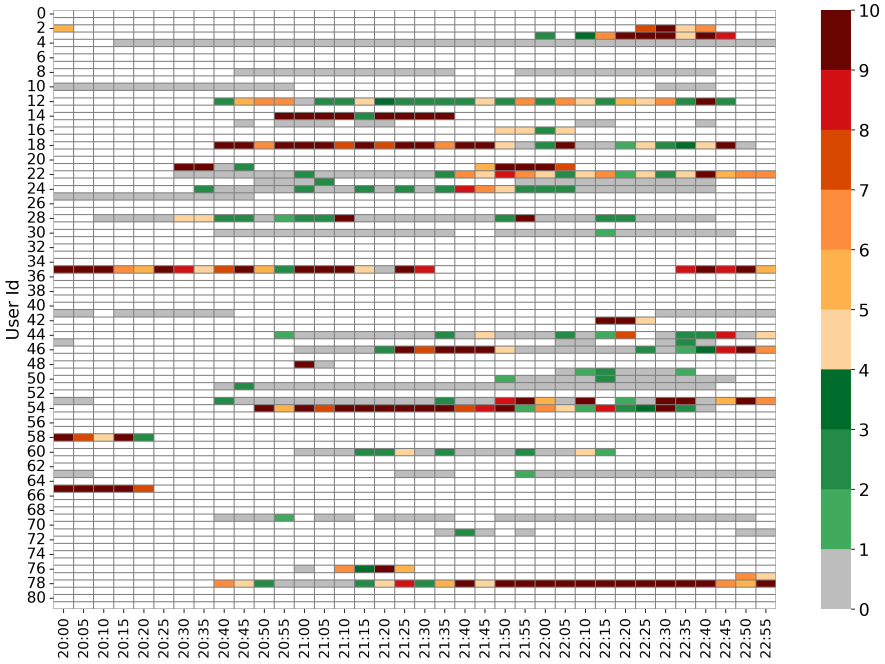


Figure A.1: Quality changes experienced by each user that had at least one quality changees during the analysed matches on 30/03/2025 between 20:00 and 23:00

Figure A.2: Quality prediction for each user that had at least one quality changees during the analysed matches on 30/03/2025 between 20:00 and 23:00

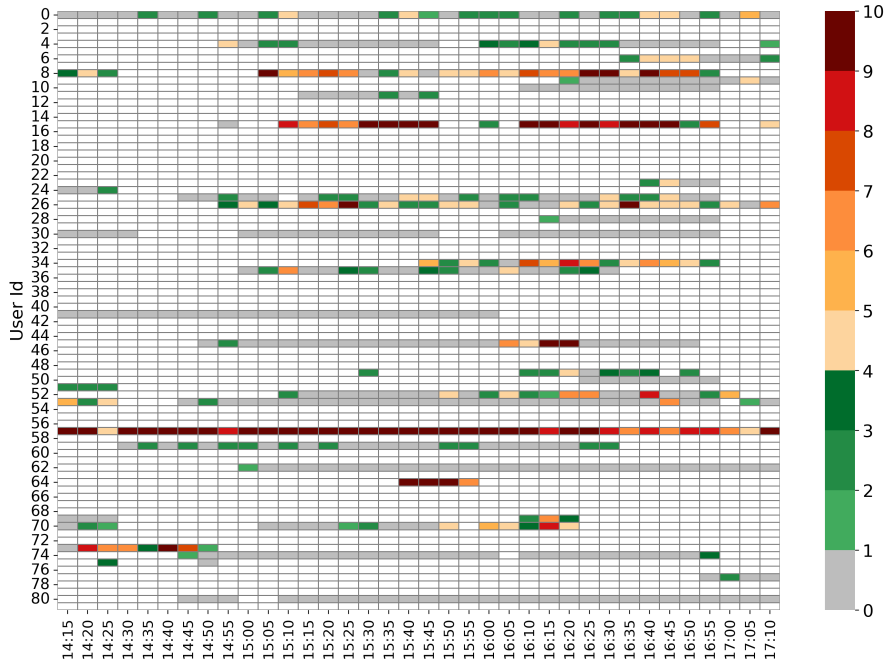


Figure A.3: Quality changes experienced by each user that had at least one quality changees during the analysed matches on 27/04/2025 between 20:00 and 23:00

Figure A.4: Quality prediction for each user that had at least one quality changees during the analysed matches on 27/04/2025 between 20:00 and 23:00

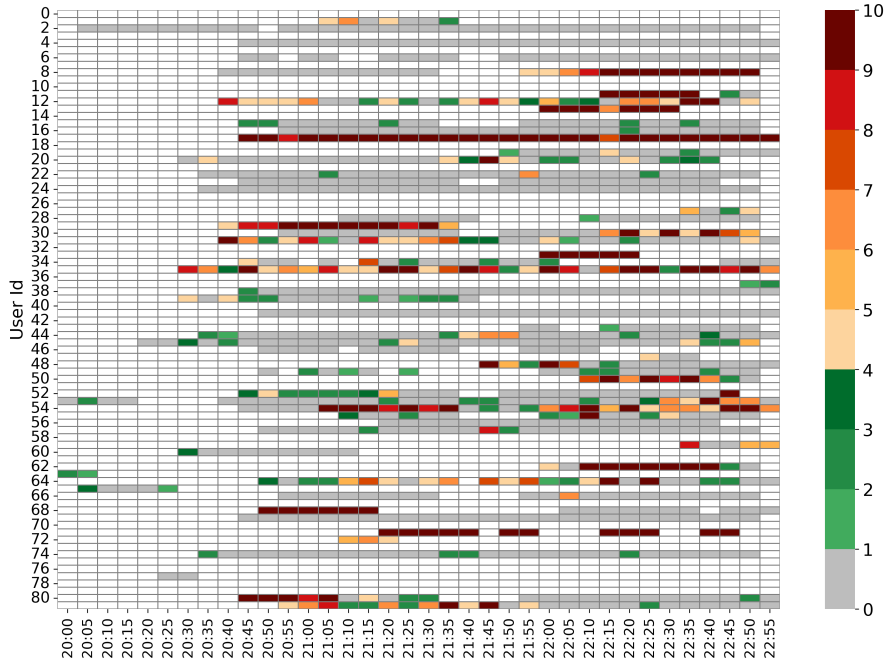


Figure A.5: Quality changes experienced by each user that had at least one quality changees during the analysed matches on 18/05/2025 between 14:15 and 17:15

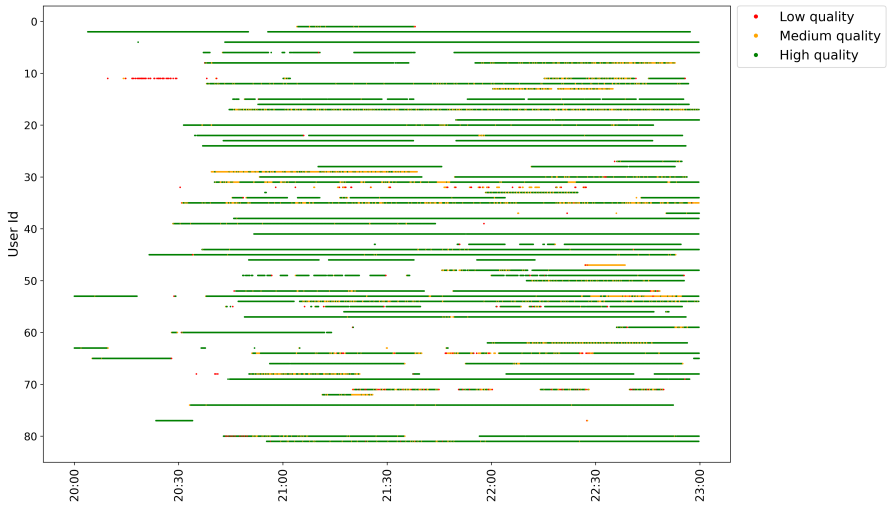


Figure A.6: Quality prediction for each user that had at least one quality changees during the analysed matches on 18/05/2025 between 14:15 and 17:15

Bibliography

[1] Mario Freire and Manuela Pereira. *Encyclopedia of Internet Technologies and Applications*. Information Science Reference, Oct. 2007 (cit. on p. 4).

[2] Telecommunication Networks Group - Politecnico di Torino. *Tstat: Network monitoring tool.* 2008. URL: `http://tstat.polito.it` (cit. on p. 4).

[3] Giorgio Daniele Luppina. *Streambot*. 2025 (cit. on p. 6).

[4] sitespeed.io. *throttle*. URL: `https://github.com/sitespeedio/throttle` (cit. on p. 6).

[5] R. Schatz P. Casas and T. Hossfeld. «Monitoring YouTube QoE: Is your mobile network delivering the right experience to your customers?» In: *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*. Apr. 2013, pp. 1609–1614 (cit. on p. 7).

[6] T. Hossfeld R. Schatz and P. Casas. «Passive YouTube QoE monitoring for ISPs». In: *Proc. 6th Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput.* July 2012, pp. 358–364 (cit. on p. 7).

[7] M. Ammar T. Mangla E. Halepovic and E. Zegura. «Using session modeling to estimate HTTP-based video QoE metrics from encrypted network traffic». In: *IEEE Transactions on Network and Service Management* 16.3 (Sept. 2019), pp. 1086–1099 (cit. on p. 7).

[8] E. Halepovic V. Krishnamoorthi N. Carlsson and E. Petajan. «BUFFEST: Predicting buffer conditions and real-time requirements of HTTP(S) adaptive streaming clients». In: *Proc. 8th ACM Multimedia Syst. Conf.* June 2017, pp. 76–87 (cit. on p. 7).

[9] Irena Orsolic, Dario Pevec, Mirko Suznjevic, and Lea Skorin-Kapov. «A machine learning approach to classifying YouTube QoE based on encrypted network traffic». In: *Multimedia Tools and Applications* 76.21 (2017), pp. 22267–22301 (cit. on p. 7).

[10] Francesco Bronzino, Paul Schmitt, Sara Ayoubi, Guilherme Martins, Renata Teixeira, and Nick Feamster. «Inferring Streaming Video Quality from Encrypted Traffic: Practical Models and Deployment Experience». In: *Proc. ACM Meas. Anal. Comput. Syst.* 3.3 (Dec. 2019) (cit. on p. 7).

[11] Michael Seufert, Pedro Casas, Nikolas Wehner, Li Gang, and Kuang Li. «Stream-based machine learning for real-time QoE analysis of encrypted video streaming traffic». In: *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. Feb. 2019 (cit. on p. 7).

[12] M. Hammad Mazhar and Zubair Shafiq. «Real-time Video Quality of Experience Monitoring for HTTPS and QUIC». In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 2018, pp. 1331–1339 (cit. on p. 7).

[13] Vaneet Aggarwal, Emir Halepovic, Jeffrey Pang, Shobha Venkataraman, and He Yan. «Prometheus: toward quality-of-experience estimation for mobile apps from passive network measurements». In: *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*. Feb. 2014 (cit. on p. 7).

[14] Giorgio Daniele Luppina. *Inferring Video Quality in Live Streaming Flows Using Network Passive Metrics*. Apr. 2025. URL: `http://webthesis.biblio.polito.it/id/eprint/35249` (cit. on p. 11).