POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Machine Learning Methodologies for QoE prediction in Satellite Networks

Supervisors
Prof. Marco Mellia
Prof. Danilo Giordano

Candidate
Andrea Di Domenico

April 2023

† A Giuseppe, mio fratello, sei stato e sempre sarai il mio punto di riferimento.

Abstract

Satellite Communication (SatCom) enables Internet access in remote locations where conventional infrastructure is unavailable or too expensive to be deployed. In a SatCom connection, a customer uses a parabola to connect to a satellite, which sends all of the customer's traffic to a ground station, which relays the traffic to the Internet. Unlike traditional infrastructures, where the latency to retrieve content is on the order of tens of ms, SatCom's latency is much higher: $\sim 550ms$ for the satellite link to reach the ground station, plus the time it takes for the ground link to reach the content. In this scenario, the Quality of Experience (QoE) of customers is significantly impacted by the SatCom connection, as slowdowns in the satellite or ground link can severely impair the QoE of customers. In order to identify and investigate such impairments, it is of utmost importance for the SatCom operator to build models to estimate the customers' QoE leveraging only network flows generated by the customers. These models are really useful for the Internet Service Provider (ISP) because in case of poor QoE it can take action with the purpose of increasing customer satisfaction.

Thus, the goal of this thesis is to develop a method for assessing customer QoE while browsing the web. I develop a machine learning model to estimate the Speed Index or the onLoad, two of the metrics used to measure Web QoE, from data flowing through the operator's network. This data is collected by Tstat, a custom flow monitoring software installed in the operator's ground station. Tstat captures all IP packets and groups them into flows to track the evolution of TCP and UDP flows.

The proposed work focuses on different parts: (i) to investigate metrics for measuring QoE in web browsing and tools for automatic benchmark (ii) to create a testbed for automatic collection of network and QoE data in web visits, (iii) to investigate the state of the art in machine learning techniques for regression and classification (iv) the creation of models to investigate QoE prediction capabilities in a real scenario.

The final dataset is created by merging two datasets: one *active*, collected on the client side, which contains the label (the metric used to measure Web QoE) and other information such as website and URL, and a second one *passive*, collected by Tstat on the ISP ground station, which contains the intercepted network flows. Relevant features are extracted from these data and used as predictors for the machine learning models.

The work considers both regression models, such as linear regression or random forest regression and classification models, such as support vector machines or

random forest classification. Although the problem can intuitively be viewed as a regression, I can also formulate it as a classification problem, e.g., by defining categories for QoE: good, medium, and poor.

The proposed model achieves promising performance in predicting Web QoE metrics on independent test sets. By further investigating aspects of real-world visits, such as the presence of local cache, and simulating typical browser visits, the performance decreases.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter acts as an introduction for the thesis work. The first part describes the satellite communication. The second part focuses on QoE description and the third part contain the description of the task proposed in this work. Finally, an overview of the SatCom network and related work in this field is presented.

1.1 Satellite Communication

Satellite communication (SatCom) is one of the main means for international and domestic communications over long or moderate distances. The main reason for using SatCom is that a single satellite spans over a third of the Earth's surface [11]. SatCom is composed by two main components (Figure 1.1):

- Ground segment
- Space segment

The sender (transmitter) uses a parabolic dish antenna on the ground to transmit the signal up to the communication satellite. Then the communication satellite receives the signal from the transmitter's antenna and amplifies it. The receiver receives the signal from the satellite using a parabolic dish antenna on the ground station. The signal is then amplified and processed to remove any noise or distortion. Once the signal has been received, the data is demodulated and decoded before being processed by the receiver's computer or communication device.

Satellite Communication plays an important role in different applications:

- DHT or Satellite Television
- Satellite radio
- GPS

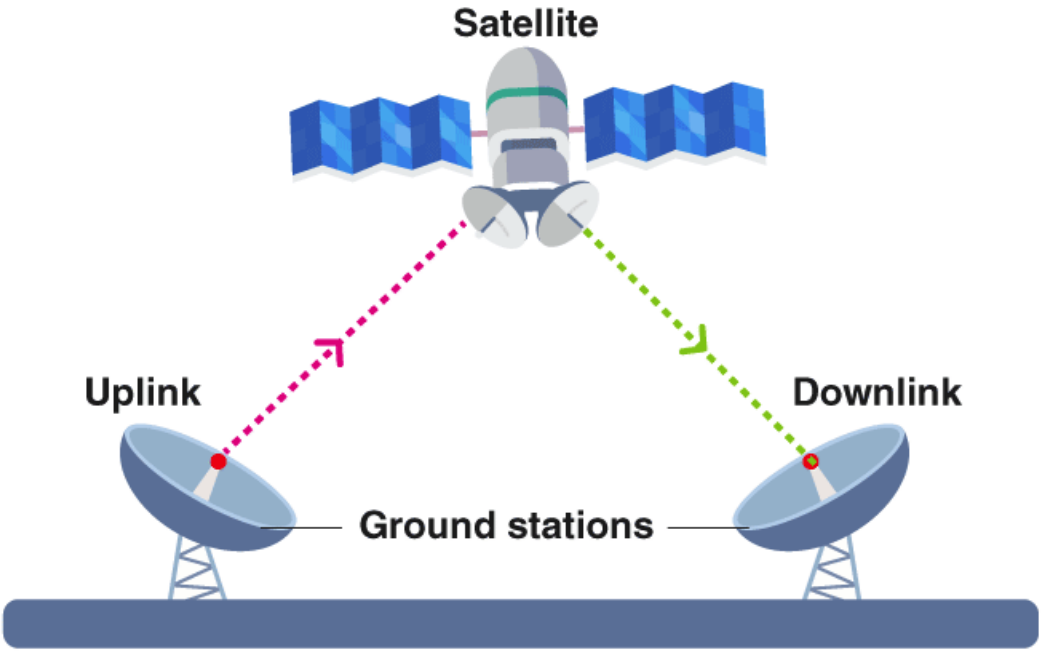

Figure 1.1. Satellite communication scheme, taken [16]

- Military application
- Weather condition monitoring
- Satellite Internet access

Our work concerns the last application. Over the last few years the growth of the internet makes Internet Service Providers one of the largest costumers for satellite services.

Satellites can be classified by their orbit distance. Geostationary (GEO) satellites are the most widely used solution. Those satellites orbits the Earth at an altitude of about 36 000 km in a circular orbit, and have the same angular velocity of the Earth's rotational speed. In this way it is possible to establish a connection between the satellite and a ground station because the satellite appears to remain stationary in the sky as viewed from a fixed point on the ground. This makes it easier to maintain a continuous connection, particularly for applications such as television broadcasting, where a constant signal is required. However, the high altitude of GEO satellites also means that there is a higher latency in the communication as signals have to travel a longer distance, resulting in a delay between the transmission and the reception of data.

There are also satellites that have a different orbit, low Earth orbit (LEO), typically between 160 km to 2000 km of altitude. Because of their lower altitude, LEO satellites can provide faster communication and lower Earth-to-sat latencies, around 25 to 35 ms, compared to GEO satellites. But unlike GEO, LEO satellites does not stay in a fixed position in the sky since they are moving at a much faster speed. So a continuous connection is not guaranteed, but there is a need for protocols to switch between satellites. Also, they have a much smaller coverage area because of the distance, there is therefore a need a satellite constellation to cover the same area. A satellite constellation consists of a high number of satellites in orbit, so LEO systems are more expensive compared to GEO systems because they require more satellites to be effective.

In addition, there are several differences between SatCom networks and terrestrial network:

- Coverage: With SatCom it is possible to cover each and every corner of the earth like rural areas or territory of underdeveloped countries; SatCom can provide also polar coverages. The cost is not dependent of the coverage area since a single satellite with geostationary orbit can cover entire continents; instead, terrestrial networks provide coverage area over a limited geographical area.

- Bandwidth: Terrestrial networks generally offer higher bandwidth compared to SatCom networks. GEO SatCom suffer from limited capacity due to the limitations of satellite technology, a single beam have a capacity on the order of 10 Gb/s.

- Latency: The distance from earth forces a Earth-to-sat round trip time above 550 ms in geostationary orbit, in addition to the terrestrial delay. This is the most important difference, because the added latency is substantial.

- Weather dependency: SatCom networks are more susceptible to weather conditions such as rain, snow, and thunderstorms, which can affect the signal quality and lead to disruptions in communication.

- Mobility: SatCom networks can provide communication services to mobile platforms such as ships, airplanes, and vehicles, while terrestrial networks are limited to stationary locations.

- Security: Terrestrial networks can be more secure due to the physical control and protection of the network infrastructure, while SatCom networks are more susceptible to interference and hacking due to the open nature of the communication channel.

1.2 Quality of Experience

According to the definition of COST Action QUALINET, Quality of Experience (QoE) is:

> [...]The degree of delight or annoyance of the user of an application or service. It results from the fulfillment of his or her expectations with respect to the utility and/or enjoyment of the application or service in the light of the user's personality and current state.[5]

Quality of Experience refers to a user's overall satisfaction with the use of a product or service. It encompasses subjective factors such as perception, emotion, and cognition, as well as objective factors such as performance and functionality. QoE is often used to evaluate the quality of multimedia services, but it can also be applied to other types of products and services.

QoE is important for ISPs because it is a measure about how users are satisfied with the quality of their service which in this case depends on many factors, such as latency, jitter and percentage of packets lost. In this way ISPs are able to improve their service in order to increase costumer satisfaction and buisness growth. ISPs want to maximize the Quality of Experience of users to avoid problems, misbehavior of the service and, as a result, churning by users. ISPs are responsible for delivering high-quality services to their customers, whether it's a home user browsing the internet or a business relying on the internet for critical operations. If the QoE is poor or inconsistent, customers may switch to a different provider, resulting in a loss of revenue for the ISP. The QoE depends on several factors, not only on the connection of the partecipants, but also on the network architecture and the in-network management. [15]

QoE measurement methods can be classified into two classes:

- **Subjective Measurements:** They are based on human ratings, they involve collecting feedback directly from users through subjective means such as surveys, interviews, or questionnaires. The goal is to gather information on the user's perception of the quality of the product or service, as well as their experience using it. This method provide rich and detailed data about the user's experience. However, they can be time-consuming and expensive to conduct, and the results may be influenced by factors such as the user's mood, expectations, and biases. A widely used measure is the Mean Opinion Score (MOS), an opinion score is a score on a predefined scale that a user assigns to his opinion of the performance of a system, MOS is the average of this values. [19]

- **Objective Measurements:** They provide an automated alternative to subjective tests. These methods involve collecting data directly from the system

or application itself, without relying on user feedback. This may include metrics such as response time, error rate, or network performance. The advantage of objective methods is that they are more objective and less susceptible to user bias. However, they may not fully capture the user's experience or satisfaction, and may not provide detailed insights into specific aspects of the user's experience.

Both subjective and objective methods have their strengths and weaknesses, and often a combination of both methods is used to obtain a more comprehensive understanding of QoE.

1.3 Motivation and Objectives

Due to the limited capacity and high latency of satellite communication, it is of great importance to have an estimate of customer satisfaction. There are several ways to do this, but because of the limitation of the SatCom connection it would be complicated to take the network data directly from the user. This causes an occupation of the user's precious bandwidth and can cause delays in communication. So data can be taken after the ground station, when the satellite connection ends and the terrestrial connection begins. The goal of this research is to create a machine learning model that is able to predict the QoE of web-browsing. This would be very useful for Internet Service Providers whose can have a score of navigation quality thus being able to take action on the state of the network. The proposed approach entails obtaining scores by using supervised machine learning models. In particular we wish to find a connection between network traffic and QoE in web-browsing seen by the user. Because this is a supervised task we need a method to capture QoE scores then tie them with the right flows in the network. Once the data collection process is finished, we need to pre-process the raw data in order to get a good result and extract features from the network flow information. Then we focus on trying various machine learning models (Linear Regression, Random Forest, Support Vector Machine) on several types of dataset and see the results.

Lastly we focus on the real case scenario, it is a tough scenario because we have multiple overlapping visits and it is difficult to distinguish the right packets that belong to a visit.

1.4 Overview of the SatCom network

The satellite navigation system is the same as that described in the work of Perdices et al. [14]. This work focuses its analysis in passive traffic characterization of a global SatCom network. It takes in consideration thousands of costumers using a

GEO satellite network. This technologies, as mentioned before, have a limited and shared with several costumers bandwidth and an RTT of at least 550 ms. GEO satellite have directional beams, each one enables efficient space and frequency multiplexing providing a total channel capacity on the order of 10 Gb/s. The operator has deployed a satellite infrastructure consisting of GEO satellites and a ground infrastructure. As we can see in Figure 1.2, costumers use a dedicated equipment, i.e., CPE (Costumer-premises Equipment), in order to connect their devices to the SatCom network. The satellite acts as a relay for subscriber traffic which reaches the ground station that forwards the traffic to the internet.

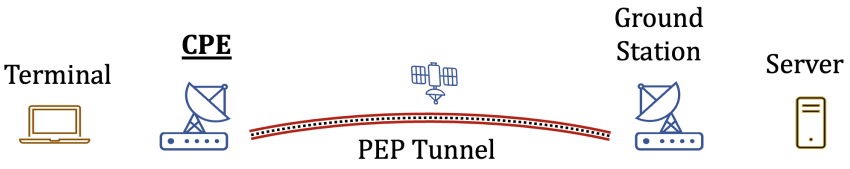

Figure 1.2. Overview of our setup using SatCom Network.

On the shared uplink channel, the first time that the CPE needs to transmit a packet it makes use of a complicated MAC protocol. To avoid collision, a Time Division Multiple Access (TDMA) scheduling protocol is used to establish connection between the CPE and the satellite. The latter forwards all the packets to the ground station via a dedicated high-capacity beams.

On the download channel, packets take the reverse path from the uplink channel, so the ground station send the packets directly to the satellite, which transmits them to the CPE by selecting the correct frequency and beam. Here the packets are broadcasted to all receivers, so CPE filters those intended for it. To have a reliable connection, the SatCom operator implements a Forward Error Correction (FEC) and an Automatic Repeat Request (ARQ) mechanisms.

Moreover, to limit the disadvantages of high latency, a Performance Enhancing Proxy (PEP) is implemented improving the TCP performance. PEPs work by intercepting and analyzing network traffic between a client and server, and making various optimizations to improve the speed and efficiency of data transfer. For example, a PEP may compress data, prioritize certain types of traffic, or optimize the use of network resources to reduce latency.

As we can see in Figure 1.3 the subscriber CPE acts as a transparent TCP proxy. It intercepts the connection between the end-user and the internet, then it forwards TCP payload to the ground station via a bidirectional reliable tunnel over UDP. Also the ground station works as a L4 proxy since when it receives a connect request it establishes a new TCP connection to the actual destination server. This

works only with TCP traffic and not with UDP traffic which is forwarded as is.

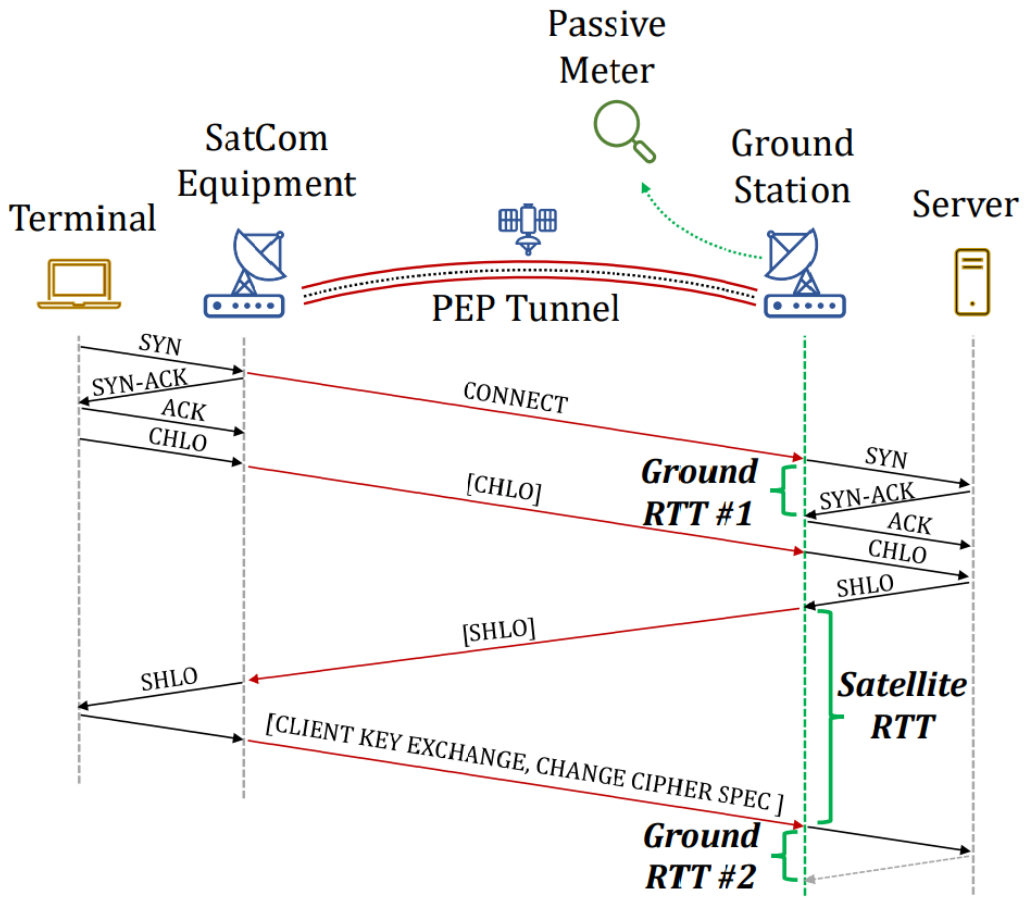

Figure 1.3. Division of the total RTT: Home RTT (Ground RTT #1), Satellite RTT and Ground RTT (Ground RTT #2).

Additionally, the ground station acts as a Network Address Translation (NAT) box and DNS resolver. The SatCom operator also uses Quality of Service schedulers to prioritize interactive traffic.

The authors installed a passive probe at the operator's ground station in Italy in order to collect passive measurements. The traffic is collected after the operation of the PEP. All user traffic from that segment of the satellite passes through here. Each user is identified by his SatCom CPE IPv4 IP address. The server runs Tstat (TCP STatistic and Analysis Tool [22]), a traffic measurements software that generates various information in real time from the processed data packets.

Here the estimation of the TCP Round Trip Time (RTT) is not accurate because of the presence of PEP, indeed as we can see in Figure 1.3 the total RTT is divided into three parts:

- *Home RTT:* between the user's device and the user's SatCom CPE.
- *Satellite RTT:* between the user's SatCom CPE and the ground station (PEP tunnel). For the estimation of this time the authors use the information provided by the TLS handshake protocol taking the measure of the time between the Server Hello message and the next Client Key Exchange message /Change Cipher Spec message. This time also contains the *Home RTT*, but it can be considered insignificant compared to the satellite.
- *Ground RTT:* between the ground station and the destination server. Here Tstat uses the information of the TCP handshake protocol, indeed the TCP connection is initiated by the ground station PEP

1.5 Related work

In recent years, satellite communication has become increasingly important because of its ability to provide reliable, and global connectivity that is critical for a range of applications and industries. Therefore many studies have been done in this area.

In [10] the performance of modern web protocols over satellite links was evaluated. In particular, technologies such as VPNs and new protocols such as QUIC. PEPs pep cannot go into action when TCP connection are encripted within VPNs. Moreover, QUIC has encrypted transport layers by design and also in this case PEPs cannot be used. The work of J, K-S, and R shows that HTTP/1.1 does not perform well, in all the case scenario (with or without PEPs); instead, HTTP/2 has much better results if it uses PEPs; HTTP/3 and QUIC achieved worse performance than HTTP/2 with PEPs.

Several research papers have already studied Quality of Experience under different perspectives [6]. In [13] the authors investigate the connection between the QoE in various wireless services and the network and channel condition. The paper proposes the *MLQoE, a modular algorithm for user centric QoE prediction.* Empirical measurements based on network metrics (average Delay, packet loss, average jitter, average burst Interarrival-packets...) are used as predictors. As predicted outcome, subjective opinion scores reported by actual users are used. The MLQoE achieves good results compared to the ML alghotitms, in fact it predicts fairly accurately the QoE score.

The work done by Alreshoodi and Woods study the existing correlation models which attempt to map Quality of Service (QoS) to Quality of Experience (QoE)

for multimedia services. [1]. The term QoS is determined by deterministic network parameters, and this paper analyses a number of previous works that seek techniques that can reliably calculate weighting coefficients for QoS/QoE mapping.

Chapter 2

Testbed

In order to satisfy the purpose of this research a dataset need to be generated. The data collection work is divided into two parts: **passive measurements** (network data) and **active measurements** (QoE data in web visits).

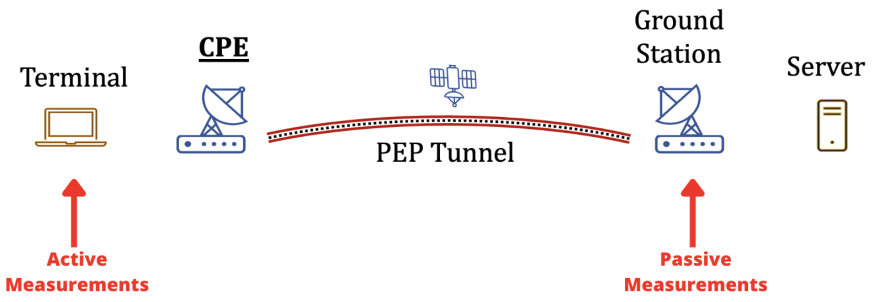

Figure 2.1. Measurements setup overview. Active measurements generated by the terminal and Passive measurements generated by capturing flows after the ground station.

2.1 Passive Measurements

This dataset is built from the work done by Perdices et al. [14]. As mentioned above a custom flow monitoring software (Tstat) is used. A passive probe is installed at the operator's ground station capturing all the data after the PEP operation. Since PEP operates like a proxy server, in the case of TCP connection, the client sends a request to establish a connection to the proxy server instead of the intended destination server. The proxy server then forwards this request to the destination server on behalf of the client. When the server responds to the request, it sends the response back to the proxy server, which then forwards it back to the client.

So in our case, Tstat intercepts and creates statistics about the communication between the PEP and the web server.

2.1.1 Tstat

Tstat (TCP STatistic and Analysis Tool [24]) is a passive sniffer developed by the networking research group at Politecnico di Torino. It is developed in ANSI C to have good efficiency and it has an highly flexible design with several plugin modules.

Tstat captures all IP packets and groups them in flows, according to some rules. Both TCP and UDP flows are identified. This grouped packets are analyzed by Tstat, which gives several summary information such as packet and byte counters. This information is calculated for both sent and received packets in order to give statistics for client-to-server and server-to-client flow.

Tstat creates different LOG files:

- **log_tcp_complete:** reports every TCP connection that has been tracked by Tstat.
- **log_udp_complete:** reports every tracked UDP flow pair.
- **log_video_complete:** reports every TCP Video connection that has been tracked.
- **log_mm_complete:** reports statistics for the RTP and RTCP flows.
- **log_chat_complete:** reports statistics for MSN Messenger, Yahoo! Messenger and Chat based on XMPP Protocol flows

All these statistics are stored in different TXT files where each row corresponds to a different flow and each column is associated to a specific measure.

In Tables 2.1 and 2.2, we have respectively relevant information for TCP flows and for UDP flows. This data is used to construct our features.

2.2 Active Measurements

This dataset is created from the data generated by the terminal connected to the internet through SatCom connection. Specifically, the client visits several web pages collecting metrics that describe the Web QoE for each visits. So, this part of the data gives us information about the target variable (dependent variable) of our machine learning model. The WebQoE metrics used are *speedindex* and *onLoad*. The experiments were done in an automated way through *browsertime* and *docker*, as will be explained later.

Column	Description
s/c_ip	IP addresses of the server/client
s/c_port	TCP port addresses for the server/client
First	Absolute time in *ms* of first packet of flow
s/c_first	Time of first server/client packet with payload since the first flow segment
s/c_rtt_avg	Average RTT computed measuring the time elapsed between the data segment and the corresponding ACK
s/c_rtt_min	Minimum RTT observed during connection lifetime
s/c_rtt_max	Maximum RTT observed during connection lifetime
s/c_rtt_std	Standard deviation of the RTT
s/c_ttl_min	Minimum Time To Live
s/c_ttl_max	Maximum Time To Live
c_tls_SNI	For TLS flows, it is the server name indicated by the client in the Hello message extensions
s/c_sit_n	With *n* equal from 1 to 9. It is the interarrival time between the *n* packet and the *n-1* packet sent by the server/client

Table 2.1. List and description of the relevant column in the *log_tcp_complete* file. [24]

Column	Description
s/c_ip	IP addresses of the server/client
s/c_port	UDP port addresses for the server/client
s/c_first_abs	First server/client packet in absolute time
quic_SNI	For QUIC flows(QUIC integrates TLS), it is the server name indicated by the client in the Hello message extensions
s/c_sit_n	With *n* equal from 1 to 9. It is the interarrival time between the *n+1* packet and the *n* packet sent by the server/client

Table 2.2. List and description of the relevant column in the *log_udp_complete* file. [24]

2.2.1 Speed Index

Introduction

Speed Index (SI) is one of the metrics that is used as outcome of our Regression model. It is expressed in milliseconds and it measures how quickly content is visually displayed during page load. Speed Index can give a solid idea of a site's overall speed, efficiency , and performance. It is also used by Google's PageSpeed Insight that defines it as "*how quickly the contents of a page are visibly populated*". SI does not take into account when the browser renders all elements, but only the visible parts of a webpage to be displayed.

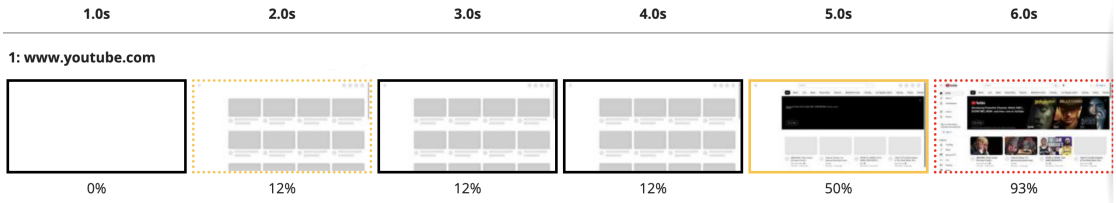

Figure 2.2. WebPageTest captures video of the page loading, showing the percentage of visually complete.

WebPageTest [26] is one of the most popular tools for measuring and analyzing the performance of web pages and it added the Speed Index metric in 2012. WebPageTest records a video of the page is loading and then it analyzes each video frame (10 frames per second). It assigns a certain percentage of completion to each frame, so the score is 0% for a blank screen and 100% for a visually complete page. In Figure 2.2 we can see an example.

Measuring Visual Progress

There are two main methods to calculate the completeness of each video frame: Visual Progress from Video Capture and Visual Progress from Paint Events.

Visual Progress from Video Capture is a technique that take histograms of the colors in the image and just look at the overall distribution of the colors in the page. The difference between the histogram of the first frame and the histogram of the last frame is calculated and used as the baseline. Then for each frame is calculated the difference between its histogram and that of the first video frame. This technique works well in most of the cases but it is very sensitive to the end state and it has problems with video playing on pages or slideshows animating.

Visual Progress from Paint Events is a more recent technique and it uses the Paint Events that are exposed by Webkit through the developer tools timeline. It does not require capturing video. The process requires a fair bit of filtering and

weighting. In the end the visual progress is calculated by adding each paint's contribution to a running total, approaching the overall total.

Measuring Speed Index

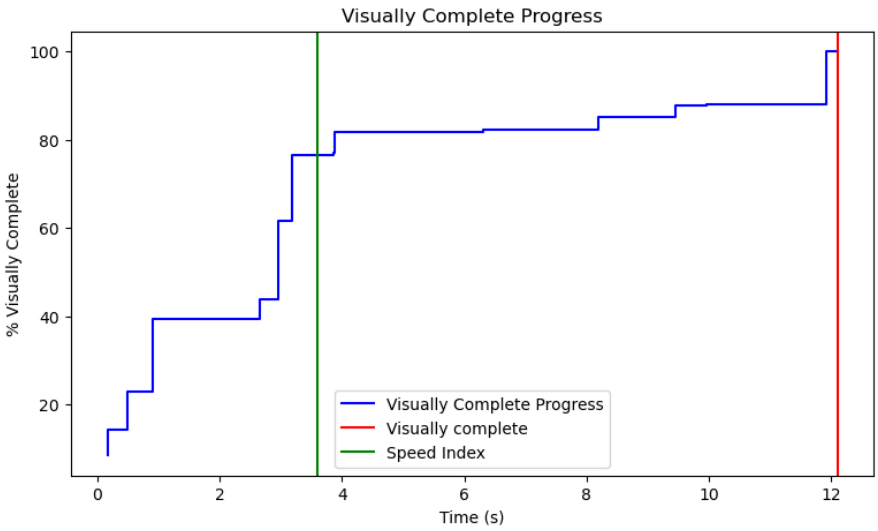

Figure 2.3. Example of Visually Complete Progress Graph

After calculating the percentages of completeness they are plotted over time obtaining a graph like the one in Figure 2.3. The formula for quantifying Speed Index is:

$$SpeedIndex = \int_0^{end} 1 - \frac{x}{100} dx$$

where *end* is the end time in millisecond, and x is the percentage of visually complete.

As we can see in Figure 2.4, the Speed Index is the area above the complete progress curve. Left pictures in Figure 2.4 shows page renders very late, instead right picture shows page starts rendering earlier; in this way the user can immediately see most of the elements on the page, showing good user experience.

2.2.2 Page Load Time (onLoad) and other metrics

The Page Load Time is measure used to estimate the QoE of Web users. Also this metric is expressed in milliseconds and it is always higher than Speed Index. It is easily measurable since it measures the time it takes for the page to fully load. It is still used as the main indicator of performance in most recent scientific

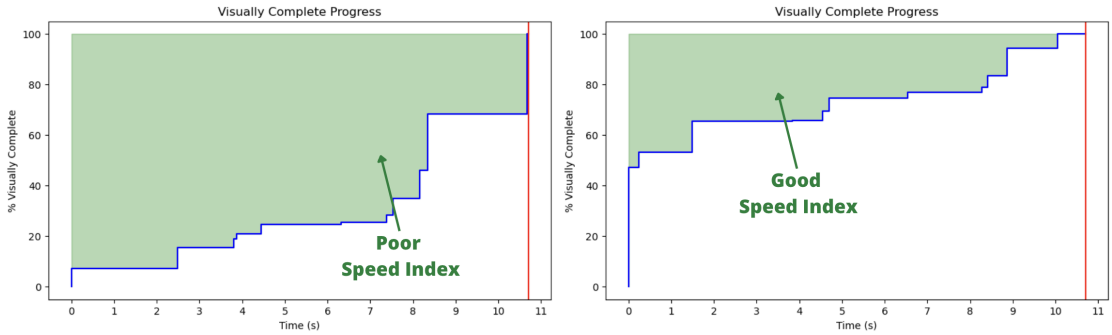

Figure 2.4. Measuring Speed Index through Visual Progress

work [2]. OnLoad is less realistic than SpeedIndex since the last object may not be important for the page rendering, but SI suffers with dynamic pages and it is computational intensive to measure.

	Metric name	Layer 3	Layer 4	Layer 7	Unit/ Range	Description
Time Instant	TTFB	≈	✓	✓	sec	Time at which the first byte of payload is received
	DOM	-	-	✓	sec	Time at which the Document Object Model (DOM) is loaded
	TTFP	-	-	✓	sec	Time at which the first object is painted
	onLoad	-	-	✓	sec	Time at which all bytes of payload have been received
	ATF [5]	-	-	✓	sec	Time at which the content "Above-the-fold" has been rendered
Time Integral	ByteIndex	≈	≈	✓	sec	Integral of complementary byte-level completion
	ObjectIndex	-	-	✓	sec	Integral of complementary object-level completion
	SpeedIndex [8]	-	-	✓	sec	Integral of complementary visual progress
Comp. Score	YSlow [21]	-	-	✓	[0,100]	Yahoo's compound score (23 weighted heuristics)
	PageSpeed [12]	-	-	✓	[0,100]	Google's PageSpeed Insight heuristics
	dynaTrace [3]	-	-	✓	[0,100]	dynaTrace's compound score
	MOS	-	-	-	[1,5]	User rating

Figure 2.5. Metrics to express user perceived quality of web browsing, taken [2]

In Figure 2.5 we can see the most important metrics used to measure quality of Web users experience. Quality of Experience is intrinsically subjective, so it is hard to find a metric able to measure it. Some of them are even not correlated each other as we can see in figure 2.6.

The user experience can be measured by an opinion score and summarized with the MOS, so onLoad and SI cannot directly capture the QoE. However, the work done by Hora et al. shows that the onLoad and Speed Index are correlated with users' QoE [8]: onLoad show a strong (0.81) Pearson correlation with MOS.

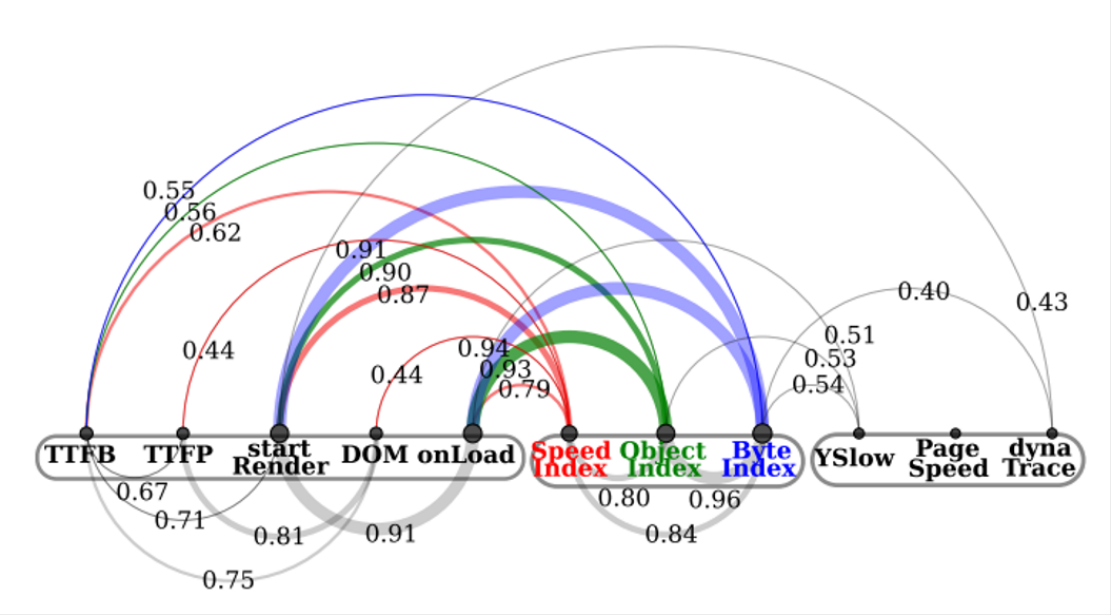

Figure 2.6. Pearson correlation between pairs of WebQoE metrics, taken [2]

2.3 Tool Description

2.3.1 Docker

Docker[7] is a software platform, it was released as open-source in 2013 and is developed by Docker, Inc., a Platform-as-a-Service (PaaS) provider company. In May 2016 an analysis showed that also organizations like Cisco , Google, Huawei, IBM, Microsoft and Red Hat are contributors to Docker in addition to The Docker Team itself.

Docker brings several advantages: Docker lets standardize application operations, easily move code, and save money by improving resource utilization.

The concept of container is the hearth of Docker, it is a small unit that contain the software and all the needed dependencies and context to run it making it possible to run the software regardless of the underlying environment. The key to its success is that a container virtualize applications in a lightweight way also because of each container shares the OS of the host machine.

Docker uses a client-server architecture. As we can see in Figure 2.7, the Docker client interacts to the Docker daemon using a REST API. The Docker daemon manages Docker object such as containers, images, network and volumes. Users interact with Docker through the Docker client which can comunicate with more than one daemon.

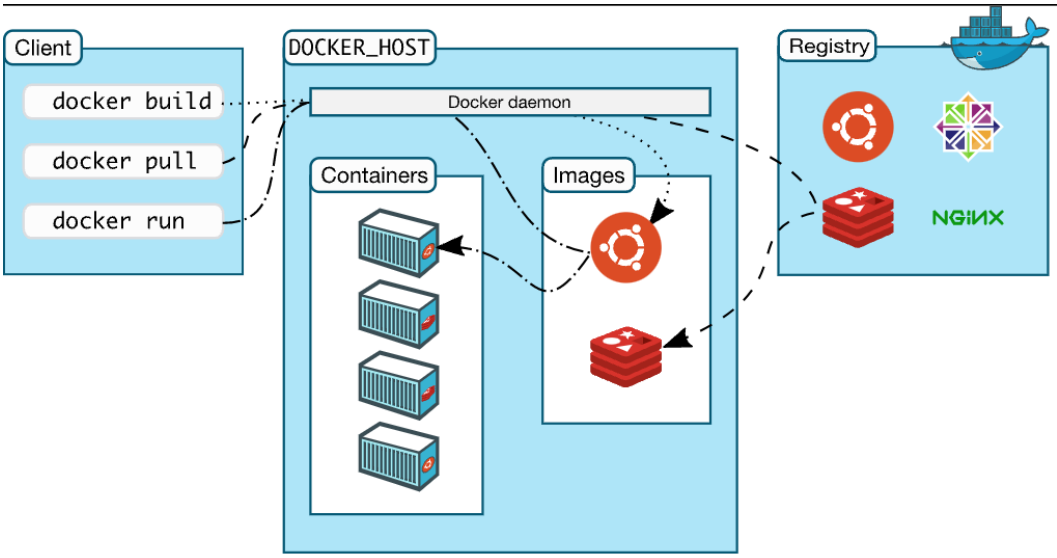

Figure 2.7. Docker architecture, taken [7]

A Docker image is a read-only template, it is the starting point to create containers since a container is a running instance of an image. Since images do not contain the kernel, which is shared with the underlying OS, they are not so heavy and slow. Image registries store all the Docker images.

2.3.2 Browsertime

Browsertime is an open-source product that is part of the family of tools sitespeed.io [17]. It is built upon other Open Source tools such as Selenium or VisualMetrics. Sitespeed.io allows us to monitor the performance of a website, simulating real users connectivity.

Browsertime is the main tool of sitespeed.io, it handles everything with the browser, in this work only Chrome browser is used but it supports several browsers such as Chrome, Firefox, edge and Safari. During the visit of the page it records a video of the Browser screen, this video is used to calculate Visual Metrics.

In order to understand how Browsertime works, sitespeed.io gives us an example of what happens when a test of a URL in browsertime is started:

1. Browsertime is run with the configuration you want.
2. Through Selenium it start Chrome or Firefox depending on your configuration.

3. Browsertime starts recording a video of the browser screen using FFMPEG.

4. The browser connects to the site via the desired URL.

5. After visiting the page it collects the timing metrics through the default JavaScript timing metrics.

6. Browsertime saves all request/responses on the page in a HAR file.

7. It stops recording a video, from which Visual Metrics like Speed Index are taken.

Moreover another advantage of browsertime is that you can run browsertime using a Docker containers, this makes it easy to deploy. Indeed the sitespeed.io team have Docker images with Browsertime, Firefox and Xvfb.

Browsertime gives as output an HAR (HTTP Archive format) that is a JSON formatted file and it is used by several HTTP session tools to export the captured data. An HAR file contains several information such as:

- All the end to end HTTP requests/responses
- Headers
- Timings
- URL
- Headers
- Cookies
- Speed Index
- onLoad
- onContentLoad

Browsertime is highly configurable, of particular importance is the option *chrome.args*. Through this option it is possible to give the user data directory to chrome. This directory contains profile data such as cookies, bookmarks, history and other information.

Figure 2.8. Example of the end to end HTTP requests/responses in an HAR file.

2.4 Browser cache and cookies

Chrome saves the cached pages and images in the a folder named *Cache*, which is itself inside the user data directory. Cookies are stored in a SQLite database file, *Cookies.sqlite*, still within the user data directory.

The **cache** is a component used to temporarily store data for faster future access. The browser cache stores different types of contents like html, images, java script etc. Chrome uses at least five files: one index file and four data files. The index file contains an hash table used to locate entries on the cache [20]. When the browser requests data from the web server, if it is available in the cache then the browser loads the data directly from its cache, without retrieving it from the web server. Through specific HTTP headers it is possible to instruct the web browser when to cache a resource, when not to, and for how long [9]. The *Last-Modified* header of a cached asset is the time a document last changed; through the *If-Modified-Since* request header field, web browser can query the server if that asset has changed over time.

Cookies are small blocks of data that a web server sends to the browser, which places this data on the memory or disk. When the browser requests an object from

the same web server in the future , the browser will send the data blocks back on the web server. In this way, a website can identify a particular user. Cookies are useful for:

- **Session management:** Web servers recognize users and recall their individual login information and preferences.
- **Personalization:** Websites can customize the page according to user preferences, thus showing personalized advertisements.
- **Tracking:** Shopping sites use cookies to track users' web browsing habits, such as items previously viewed.

Browsertime every time it loads a web page, it loads it as if it were being displayed for the first time, without using either cookies or the browser cache. In order to make the automatized tool for testing closer to the real case, there is the need to use cookies and cache also in our test. In fact, on the first visit to many websites, a small pop-up notifications (cookie banners) appear, covering the rest of the page. This cookie banner asks for users' consent before using cookies. Visiting web pages in this way would skew our tests, since in the real case scenario a user would have already accepted cookies and filled the browser cache.

In this context Browsertime comes to meet us with the *chrome.args* option used to refer to the user data directory. The cache is filled simply by making an initial visit to the desired site. As for cookies, the issue is more complicated in that you have to accept them once you visit the site. But here again browsertime helps us by providing *Test by scripting* [4], which makes it possible to measure a user journey through JavaScript. A user can login, visit multiple pages and click on HTTP items such as the *Accept All* button of cookie banners. By accepting cookies, they will be saved inside the user data directory.

With these two expedients we can now make real visits to web sites, simulating being real users.

2.5 Creation of the dataset

At this point we have a series of HAR files from the active measurements and a series of csv files from the passive measurements. As a first step we concatenate all the file of passive and active measurements respectively, obtaining two different datasets. Then we need to merge these two datasets, to do this to each row of the passive dataset (a single flow traced by tstat) we assign an active measurement. In order to do this we need to know the IP address of our machines since Tstat traces all the IP packets passing through the ground station of the ISP.

Since Tstat anonymize IP addresses through Crypto-PAn (Cryptography-based Prefix-preserving Anonymization) the IP addresses of our machines are obtained through reverse engineering work by visiting not very popular sites and then search in the passive dataset the IP that have visited that sites on a given day and time.

After filtering the data by IP address, the pairing is made based on the Server Name Information (SNI), it is an extension to the TLS protocol that allows a browser to indicate which hostname it is trying to connect at the start of the handshake process. So if we are browsing *www.google.com*, the first flow with SNI equal to *google.com* will be the trigger for the start of the visit in the passive data. Then every flow that falls in the measurement interval of the active data is assigned to that same measurement. In Figure 2.9 there is a summary of the data set process flow.

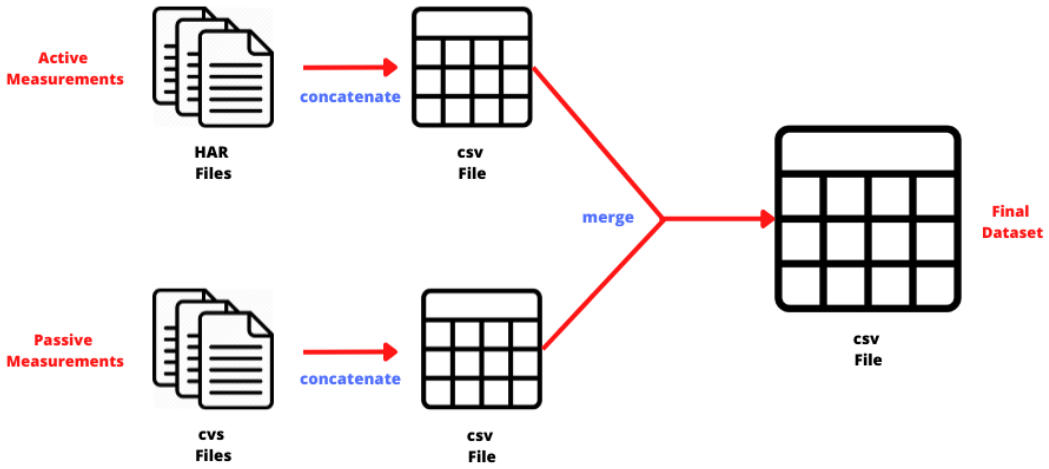

Figure 2.9. Dataset process flow

Each row of the merged data set is a single flow traced by tstat accompanied by the information from the active data set, such as Speed Index, onLoad, URL and website. We have the label needed by our regressor (Speed Index or onLoad) and the information given by tsat that need to be processed to extract relevant features.

Chapter 3

Methodologies

3.1 The Regression problem

Regression refers to predictive modeling problems that involve predicting a numeric value. It is one type of supervised learning problem which wants to find the relationship between a dependent variable and one or more independent variables. In other words, given a set of input variables X and their corresponding output values Y, the goal of the regression problem is to find a function f(X) that maps the inputs to the outputs with the least amount of error or loss.

$$Y_i = f(X_i, \beta) + e_i$$

where β is an unknown parameter, X_i are the independent variables, Y_i the dependent variables and e_i the error terms.

Regression models can be linear or nonlinear, depending on the nature of the relationship between the input and output variables. In Figure 3.1 we can see the differences between linear and nonlinear regression. The main difference between linear and nonlinear regression is the shape of the relationship between the dependent variable and the independent variables. Linear regression models assume a linear relationship, while nonlinear regression models assume a nonlinear relationship. In general, nonlinear regression models can capture more complex relationships between variables than linear regression models. However, nonlinear regression models can also be more difficult to estimate and interpret than linear regression models.

3.1.1 Linear Regression

Linear regression assumes that there is a linear relationship between the dependent variable and one or more independent variables. The linear equation takes the form

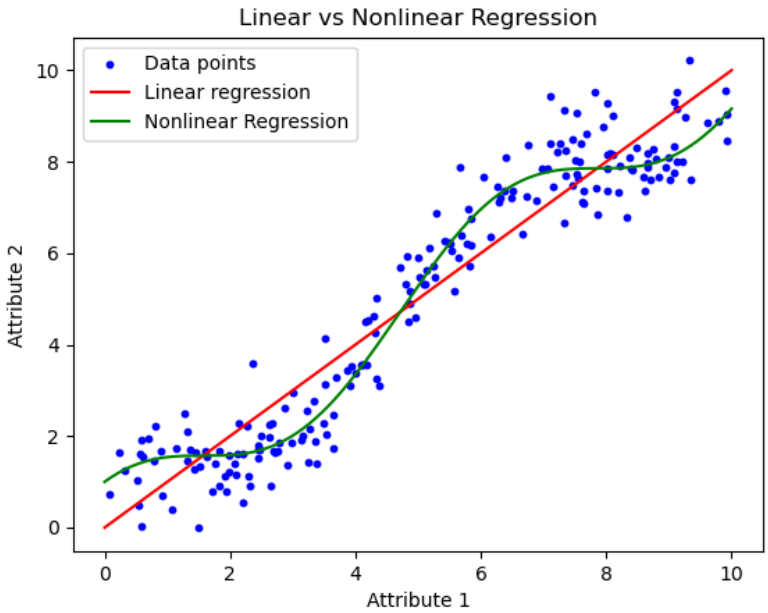

Figure 3.1. Linear vs Non-linear regression.

of $y = mx + b$, where y is the dependent variable, x is the independent variable, m is the slope or coefficient of x, and b is the y-intercept.

The goal of linear regression is to estimate the values of m and b that best fit the observed data. This line is the one which minimize an error measured using a loss function. Error is the distance between the point to the regression line. Typically, to estimate the parameters m and b a technique called least squares regression is used, it minimizes the sum of the squared differences between the predicted and observed values of the dependent variable.

Once the linear equation has been fitted to the data, it can be used to make predictions about the dependent variable based on the values of the independent variable.

3.1.2 Random Forest Regressor

Random forests is a popular ensemble learning technique used for both classification and regression tasks. It is one of the most used methods thanks to its robustness and ease of use. Random Forest Classifier (RFC) is used for classification problems, where the goal is to predict a categorical variable. On the other hand, Random Forest Regressor (RFR) is used for regression problems, where the goal is to predict a continuous variable. The method is based on the creation of a

forest of decision trees and the random selection of attributes.

RFR builds multiple decision trees and combines their predictions to obtain a more accurate and stable prediction. A Decision Tree is a type of supervised learning algorithm, it creates a tree-like model of decisions and their possible consequences. It divides the dataset into smaller subsets by creating decision nodes that split the data based on specific conditions. Each decision node corresponds to a test of some attribute, and each leaf node represents a label. The decision trees in a Random Forest are constructed using a random subset of the training data and a random subset of the features. The final prediction is then calculated by averaging the predictions of all the individual decision trees.

Building a Random Forest Regressor involves the following steps:

1. Random subset of data: Randomly select a subset of the training data, with replacement. This is known as a bootstrap sample.

2. Random subset of features: Randomly select a subset of features from the dataset. This subset of features is used to determine the best split at each node in the decision tree.

3. Build decision tree: Using the selected data and features, construct a decision tree. This tree is built recursively by splitting the data into subsets based on the selected features and finding the best split.

4. Repeat steps 1-3: Repeat steps 1-3 to build multiple decision trees.

5. Combine predictions: For a new data point, obtain the prediction from each of the decision trees and average the predictions to obtain the final prediction.

This steps are summarized in Figure 3.2. The randomness helps in building an uncorrelated forest of trees, and model fits the input data in a shorter time as each decision tree is independent, making parallel computing and modeling possible [25].

This methods has an high accuracy and efficiency, it is also scalable both in training set size and attribute number. RFR can also handle missing values and outliers, and it is less prone to overfitting than decision trees. However, it can be computationally expensive, especially for large datasets with many features.

3.1.3 Regression metrics

These metrics are used to evaluate how well the predictions made by the regression model match the actual values in the test data, and they can help you to determine whether the model is accurate and reliable enough for practical use. The metrics that we used to evaluate regression models are:

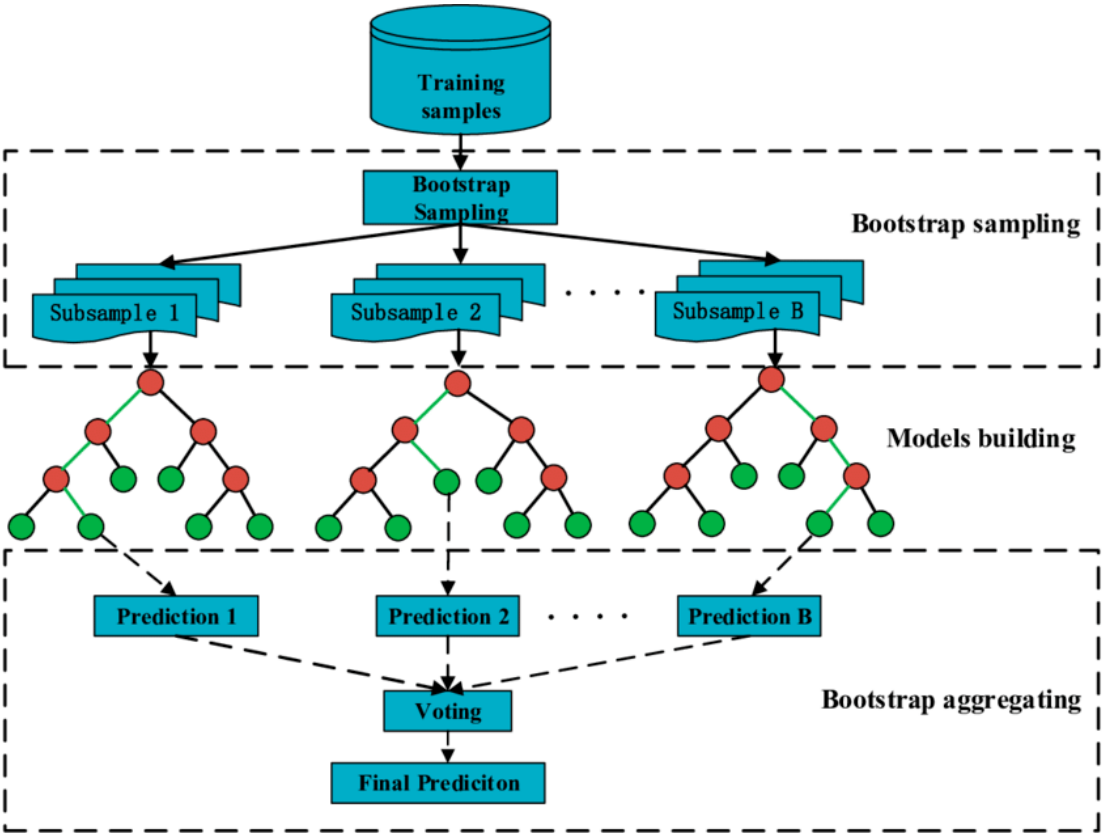

Figure 3.2. The scheme of Random Forest algorithm, taken [25]

R-squared (R2 Score)

R-squared, or coefficient of determination, is the proportion of the variance for a dependent variable that is predictable by an independent variable or variables. It normally ranges between 0 and 1, but it is possible to have negative values. An R-squared of 1 means that all of the observed variation can be explained by the model's input.

To calculate R2 score let us suppose we have a set of n predicted value $y_1, ..., y_n$ and a set of n fitted value $f_1, ..., f_n$. We define the residuals as:

$$e_i = y_i - f_i$$

And the mean of the observed data as:

$$\overline{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

The residual sum of squares is the unexplained variation:

$$SS_{res} = \sum_i e_i^2$$

The total sum of squares is the total variation:

$$SS_{tot} = \sum_i (y_i - \overline{y})^2$$

So the R2 score can be calculated by :

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

R-squared will give a good estimate of how well the regression predictions approximate the real data points. However it does not necessarily indicate that the model has a good fit, it is possible to have a good R2 score and have a poorly fitted model and vice versa.

Mean Absolute Error (MAE)

Mean absolute error is calculated as the sum of absolute errors divided by the sample size.

$$MAE = \frac{\sum_{i=1}^{n} |e_i|}{n}$$

Where e_i are the residuals, defined above.

MAE is very easy to interpret, it makes a comparison of predicted and observed value. Moreover each error contributes to MAE in proportion to the absolute value of the error.

Mean Absolute Percentage Error (MAPE)

The mean absolute percentage error is an evaluation metrics that measures the prediction accuracy of a method. It is commonly used as a loss function for regression problems and in model evaluation. MAPE is defined by the formula:

$$\frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i + f_i}{y_i} \right|$$

where y_i is the predicted value and f_i is the forecast value.

MAPE gives a very intuitive interpretation in terms of relative error, but it has problems with zero or close to zero values and it puts a heavier penalty on negative errors (when $y_i < f_i$), than on positive errors.

Root Mean Square Error (RMSE)

The root-mean-square error of root-mean-square deviation represents the square root of the second sample moment of the residuals.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(e_i)^2}{n}}$$

RMSE possesses disadvantages in interpretability over MAE. Also, each error does not influence RMSE in direct proportion, but RMSE accounts for both systematic and random events, which is not the case for MAE.

3.2 The Classification problem

Classification is a supervised learning problem which find a mapping from the space of input vectors representing the object to a discrete set of labels. It is an example of pattern recognition problem.

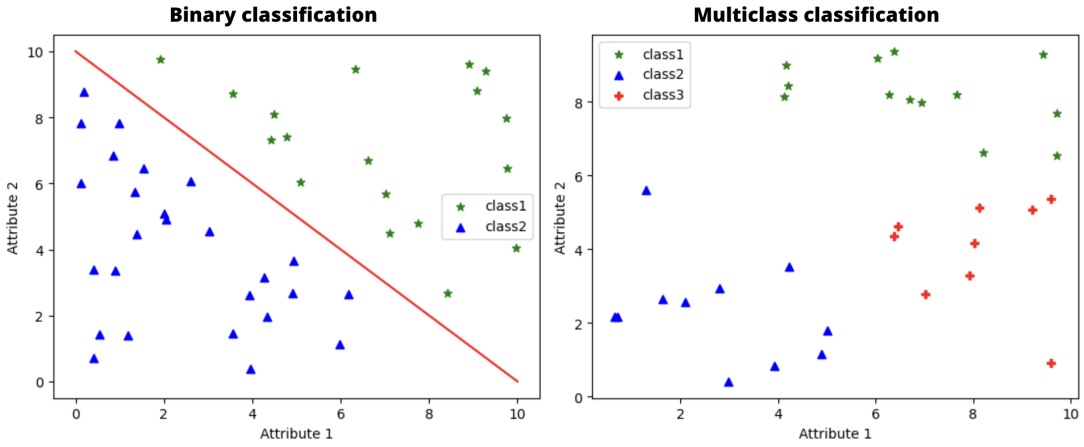

Figure 3.3. Binary vs multi-class classification.

Given $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, classification search a function *f(x)* to predict y given x, where y is categorical. x can be multi-dimensional, each dimension corresponds to an attribute. We have a binary classification if there are two classes, so y can assume only two values (0 and 1 for example). Instead we have a multiclass or multinomial classification if there are three or more classes. In figure 3.3 we can see two distinct clusters for the left picture and three distinct clusters for the other one.

The most used algorithms for binary classification are:

- Logistic Regression
- k-Nearest Neighbors
- Support Vector Machine (SVM)
- Decision Trees

Algorithms like SVM or Logistic Regression are specifically designed for binary classification, but they can be adapted for multi-class problems using two techniques:

- **One vs. All**: There are N-binary classifier models, one for each class.
- **One vs. One**: For each pair of classes we have a binary classifier model ($N * \frac{N-1}{2}$ models)

3.2.1 Random Forest Classifier

Random Forest Classifier (RFC) uses a very similar method to that already mentioned for the RFR. The main differences between Random Forest Classifier and Random Forest Regressor are:

- Output: The output of Random Forest Classifier is a class label, while the output of Random Forest Regressor is a numerical value.
- Splitting criterion: In Random Forest Classifier, the splitting criterion used to build decision trees is usually based on information gain or Gini index. In Random Forest Regressor, the splitting criterion is usually based on reducing the variance of the target variable.
- Handling outliers: Random Forest Classifier is less sensitive to outliers in the data because the final classification decision is based on a majority vote from the ensemble of decision trees. In contrast, Random Forest Regressor is sensitive to outliers because it uses the mean value of the predictions from the ensemble of decision trees to make the final prediction.

3.2.2 Support Vector Machines

Support Vector Machines (SVMs) were developed by Cortes & Vapnik in 1995, originally only for binary classification but later they were also applied to the multi-class classification problem through the One-vs-All or the One-vs-One technique.

We now consider two classes that are linearly separable, as we can see in Figure 3.4 we have an infinite number of separating hyperplanes. SVM looks for the

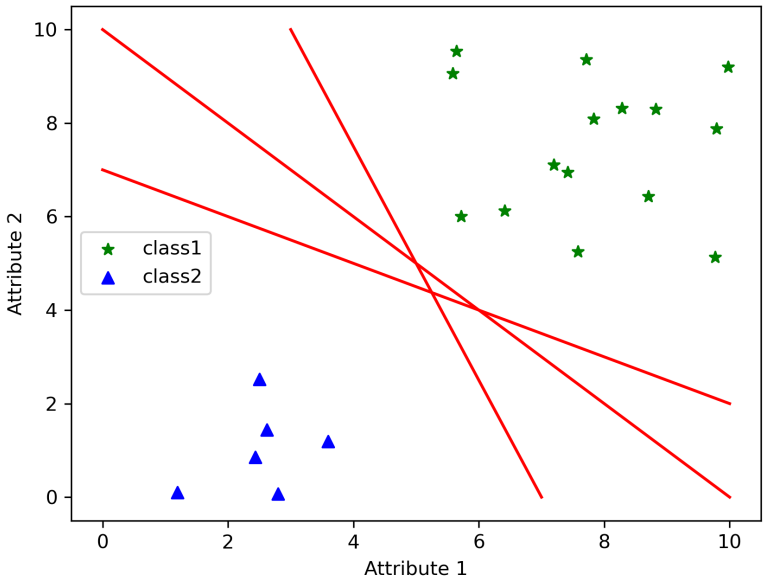

Figure 3.4. Infinite number of linear separation hyperplanes for two classes (linearly separable).

optimal separating hyperplane by maximizing the *margin* between the two classes. The margin is defined as the distance of the closest point with respect to the separation hyperplane. The points lying on the boundaries are called *support vectors*.

In the non-ideal case where classes are not linearly separable, we want to minimize the points on the wrong side of the margin (*soft margin*). In order to do this a parameter C is introduced, it determines the trade-off between margin and errors on the side of the margin.

One advantage of SVM is that is possible to train a SVM in a large (even infinite) dimensional Hilbert space having a complexity that depends only on the number of training points. We can map the data from the original space to the expanded space and then we linearly separate the data in this expanded space, which corresponds to a non-linear separation surface in the original space.

3.2.3 Classification metrics

Confusion Matrix

Confusion matrix is a table used in classification problems to visualize the performance of an algorithm. It is really useful for visualizing how much the algorithm is confusing the classes.

		Actual Class	
		Class A	Class B
Predicted Class	Class A	True Positive	False Positive
	Class B	False Negative	True Negative

Table 3.1. Binary confusion matrix.

		True Class		
		Class A	Class B	Class C
Predicted Class	Class A	TP_A	E_{BA}	E_{CA}
	Class B	E_{AB}	TP_B	E_{CB}
	Class C	E_{AC}	E_{BC}	TP_C

Table 3.2. Multi-class confusion matrix.

In Tables 3.1 and 3.2, we have an example of binary confusion matrix and multi-class confusion matrix respectively. Each row of the matrix represents the instances in a predicted class, instead each column represents the instances in an actual class.

In the case of a three-class classification problem, as in Table 3.2, TP_A is the number of true positive samples in class A, E_{AB} and E_{AC} are misclassified samples. So it is possible to calculate FN_A, the number of False Negative in the class A as $FN_A = E_{AB} + E_{AC}$.[21]

From this matrix it is possible to calculate several common metrics such as accuracy, precision, recall, and F1 score as we will see below.

Accuracy

Classification accuracy is a metric useful for evaluating the performance of a classification model. It is one of the most used metric because of it is very intuitive to understand and easy to calculate.

Starting from the confusion matrix the formula for quantifying binary accuracy

is:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where:

$$TP = True\ Positive;$$
$$TN = True\ Negative;$$
$$FP = False\ Positive;$$
$$FN = False\ Negative$$

More in general, in the case of 3 or more classes, accuracy can be calculated as:

$$Accuracy = \frac{correct\ classifications}{all\ classification}$$

Accuracy ranges from zero to one, it works as a percentage, so high value of accuracy indicate high classification performance. Accuracy is sensitive to the imbalanced data, when the samples of one class outnumber the samples of the other classes. Since this metric uses values from both columns of the confusion matrix, if the distribution of the data changes, also the value of the metric itself changes regardless of the classifier performance. [21]

F1-score

$F_1 - score$, also called F-measure, can be calculated from precision and recall.

Recall, or True positive rate (TPR), and **precision**, or positive prediction value (PPV) are defined as:

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

$F_1 - score$ represents the harmonic mean of precision and recall, defined as:

$$F_1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

Precision, recall ad F1-score are computed per-class, so in multi-class classification F1 score is computed for each class in a One-vs-Rest approach. For example, if we have three class then we have three F1 scores, one for each class.

It is more convenient to have a single number to describe overall performance, there are three different average F1 scores:

- **Macro average:** It is calculated using the arithmetic mean of all per-class F1-scores.
- **Weighted average:** It is computed by taking the mean of all per-class F1 scores while considering the number of each class samples.
- **Micro average:** It corresponds to the proportion of correct classification out of all classification, that is the same definition of *accuracy*.

The F1 score ranges from 0 to 1, with a higher score indicating better performance of the model. It is particularly useful when the classes are imbalanced or when both precision and recall are important for the problem at hand.

3.3 Feature Selection

As we will see later, the dataset will have a large number of features. To improve the performance of a model, it can be very useful to reduce the dimensionality of the data.

Feature selection is the process of selecting a subset of relevant features from a larger set of features that will be used as input for a predictive model. The goal of feature selection is to remove irrelevant or redundant features and retain only relevant features in order to improve the performance of the model, reduce the computational cost, reduce the dimensionality of the problem and, in some cases, improve the performance of the model by avoiding overfitting. It is also useful to improve the interpretability of the model by focusing on the most important features. The output of any feature selection method can be either a subset of features or a ranked list of features.

Also Feature Selection algorithms, such as learning models, can be divided in supervised, unsupervised and semi-supervised. In supervised feature selection the class/label information is contained in the data available and it is used for determining the feature quality. Unsupervised feature selection does not contain the label information, its basic idea is to cluster data such that similar objects are grouped together and dissimilar objects are separated [12].

There are several methods for Feature Selection:

- Filter methods: These methods select features based on statistical measures such as data consistency or mutual information between the feature and the target variable.
- Wrapper methods: These methods use a specific learning algorithm to evaluate the importance of each feature. It can be quite costly because it needs to build a classifier every time when a feature is considered. *Recursive feature*

elimination (RFE) is a popular wrapper method that recursively removes less important features until the optimal subset of features is identified.

- Embedded methods: These methods combine the feature selection process with the model training process. For example, decision trees and random forests have built-in feature selection mechanisms that assign importance scores to each feature based on how much they contribute to the accuracy of the model.

Recursive Feature Elimination (RFE) is a wrapper method, so it uses a specific machine learning algorithm to evaluate the importance of each feature. The method recursively removes less important features until the optimal subset of features is identified. It is one of the most popular feature selection algorithm since it is easy to use, it is a flexible method that can be applied to any machine learning algorithm and it can handle non-linear relationships between features and the target variable. On the other hand, it can be computationally expensive and it may not work with highly correlated features.

The working mechanism of RFE algorithm is:

1. A machine learning model is trained on the current set of features.
2. The importance of each feature is determined based on its contribution to the accuracy of the model.
3. The n least important features is removed, where n is the *step* parameter of the algorithm.
4. Steps 1-3 are repeated until the desired number of feature is reached.

The optimal number of features is determined by the cross-validation performance of the machine learning model with different subsets of features. The final subset of features selected by RFE is the one that yields the highest cross-validation score.

The Beiman's paper [3] describe the possibility to get the importance of variables with **Random Forests**. The feature importance in Random Forest is calculated based on the Gini impurity or mean decrease impurity of each feature. The *Gini impurity* is a measure of the probability of misclassifying an observation, it measures of how often a randomly chosen sample would be incorrectly classified based on the distribution of the target variable in the samples. The mean decrease impurity of a feature is the total reduction of the impurity that results from splitting the data based on that feature. In other words, it measures how much each feature contributes to the reduction in impurity when splitting the data.

The feature importance values are then normalized to sum up to 1, so that they can be compared and ranked. The higher the feature importance value, the more important the feature is for making accurate predictions.

3.4 Validation

To help assess of performance of a model the process of validation is introduced. Validation refers to the process of evaluating the performance of a trained model on a dataset that was not used for training. This is done to assess how well the model can generalize to new, unseen data.

The purpose of validation is to check if the model has learned the underlying patterns and relationships in the data and can accurately predict outcomes on new data. If the model performs well on the validation set, it is likely to generalize well to new data. If the performance is poor, it indicates that the model may have overfit to the training data and needs to be improved.

Cross-validation is the most commonly used method for validation. In k-fold cross-validation the data is divided into k equal sized subsamples. The model is trained on k-1 folds and validated on the remaining fold. This process is repeated k times, with each fold being used as the validation set once. The performance of the model is then averaged over the k-folds to obtain an estimate of its generalization performance.

K-fold cross-validation is a widely used technique for model selection, hyper-parameter tuning, and performance evaluation in machine learning. It provides a more reliable estimate of a model's performance than a single train-test split, as it uses all the available data for both training and validation.

Stratified k-fold cross-validation is a variant of k-fold cross-validation that is commonly used for classification problems when the dataset is imbalanced, meaning that some classes have significantly fewer examples than others. Stratified k-fold cross-validation ensures that each fold contains approximately the same proportion of samples from each class as the original dataset. This is achieved by first dividing the data into groups based on their class labels and then sampling from these groups to create each fold. It is a more appropriate technique for evaluating classifiers on imbalanced datasets than standard k-fold cross-validation, as it provides a more reliable estimate of the classifier's performance.

3.5 TF - IDF

When approaching to the real case scenario, it is possible to overlap flows belonging to different web page visits. Therefore, there is a need to find a way to identify and then aggregate flows belonging to the same visit. In this context, Term Frequency - Inverse Term Frequency can help identify to which visit the streams belong.

Term Frequency–Inverse Document Frequency (TF-IDF) [18] is one of the most popular term-weighting schemes. It is a statistical measure used to evaluate how important a word is to a document in a collection of documents or a corpus. It

is based on the idea that the importance of a word in a document increases as the frequency of occurrence of the word in the document increases, but is offset by the number of documents in the collection containing that word. The main idea behind this measure is to give more relevance to the terms that appear in the document, but which in general are infrequent.

The TF-IDF value of a term t in a document d is calculated as the product of two factors, the term-frequency and the inverse-document-frequency:

$$TF - IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

where t is a term, d is the document in which t appears, and D is the set of documents.

Term Frequency (TF): It measures how frequently a term t occurs in a document d. It is calculated as the number of times the term t appears in the document d divided by the total number of terms in the document d.

Inverse Document Frequency (IDF): It measures how rare a term t is in the corpus:

$$IDF(t, D) = \log \frac{|D|}{1 + |d \in D : t \in d|}$$

where |D| is the total number of documents in the corpus and the denominator indicates the number of documents where the term t appears, adjusted (it is increased by one) to avoid division by zero if t is in none of the documents.

The higher the TF-IDF value of a term, the more important it is to the document. Terms with high TF-IDF values are typically those that are frequent in a particular document but rare in the overall corpus, and thus they provide significant meaning and context to the document.

Chapter 4

Dataset

4.1 No cache dataset

It consists of browsing 87 main pages of different sites. It is an horizontal dataset, because we have a lot of sites but we do not go in depth, indeed we do not navigate to internal pages of sites.

In Table 4.1 there is the list of the 87 sites. In total, we have about 45k active experiments distributed over the different sites. These visits were collected between April and July 2022, and they are made without the use of browser cache and acceptance of cookies.

The dataset can be characterized from different points of view, by analyzing the statistics of the Speed Index and onLoad.

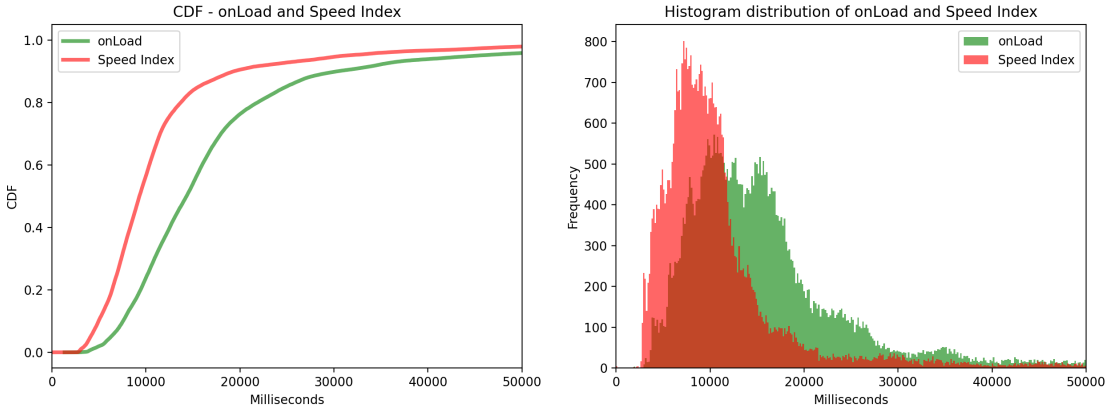

Figure 4.1. Cumulative Distribution Functions and histograms of onLoad and Speed Index of the no cache dataset.

20minutos.es	3bmeteo.it	accuweather.com	adobe.com
amazon.com	amazon.fr	amazon.it	atuuat.africa
autotrader.co.za	belgium.be	briefly.co.za	canva.com
corriere.it	csir.co.za	daft.ie	dailymaverick.co.za
elmundo.es	eltiempo.es	facebook.com	fakaza.com
focus.de	game.co.za	giallozafferano.it	github.com
google.co.za	google.com	google.de	google.es
google.fr	google.it	gumtree.co.za	honda.co.za
indeed.com	independent.ie	instagram.com	iol.co.za
leboncoin.fr	lefigaro.fr	legossip.net	lequipe.fr
linkedin.com	live.com	mediaset.it	mg.co.za
microsoft.com	news24.com	netflix.com	nedbank.co.za
office.com	otto.de	payfast.co.za	paypal.com
pinterest.com	pornhub.com	reddit.com	privateproperty.co.za
repubblica.it	spiegel.de	samsung.com	sacoronavirus.co.za
sport.es	twitter.com	standard.be	standardbank.co.za
thefuse.co.za	telenet.be	telkom.co.za	techpoint.africa
thejournal.ie	tiktok.com	timeslive.co.za	stackoverflow.com
uct.ac.za	unisa.ac.za	vodacom.co.za	vrt.be
web.de	welt.de	whatsapp.com	wikipedia.org
woww.co.za	xnxx.com	xvideos.com	yahoo.com
youtube.com	zdf.de	zoom.us	

Table 4.1. List of the sites in the no cache dataset.

In Figure 4.1 there are the *Cumulative Distribution Functions* (CDFs) of onLoad and Speed Index. The CDF of a random variable X is defined as:

$$F_X(x) = P(X \leq x), \forall x \in \mathbb{R}$$

So CDF is the probability that X will take a value less or equal than x. The CDF takes values between 0 and 1, inclusive, and is non-decreasing, meaning that as x increases, the probability of X being less than or equal to x also increases.

The histogram in Figure 4.1 gives us a good overview of the data. Histogram is a graphical representation of the distribution of a dataset. It displays the frequency of values falling into a specified interval, called a bin. The horizontal axis of the histogram represents the range of values in the dataset, divided into contiguous and non-overlapping bins, while the vertical axis represents the frequency or count of values falling into each bin. At a glance, we can see that the Speed Index data are skewed distributed, the peak is around 8 seconds. Histogram of the onLoad can identify bimodal distribution with two peaks, one at 10 seconds and the other

at 15 seconds. Both distributions extend further into the higher values than to the lower values.

4.2 Dataset with cache and accepted cookies

The dataset with cache and accepted cookies consists of browsing 12 popular websites(Table 4.2). In this dataset the navigation is more vertical, there are fewer sites but for each site also more internal page in addition to the main page. In total, we have 40k active experiments, distributed over the different sites and URLs. With this dataset we wanted to get closer to the real case by accepting cookies where we have an invasive banner and by using the browser cache. The data were collected in two different periods: between 23 December 2022 and 9 January 2023 and between 24 January 2023 and 8 February 2023.

The selection of internal pages was made by covering as much as possible the different types within a website. For example, in site like youtube, there are URLs of videos, channels and users.

Site	**Number of URLS**	**Cookies accepted**	**Invasive cookies banner**
accuweather.com	4	Yes	Yes
amazon.com	8	No	No
google.com	11	Yes	Yes
indeed.com	8	Yes	Yes
lefigaro.fr	15	No	Yes
lequipe.fr	10	Yes	Yes
pornhub.com	10	No	No
repubblica.it	10	Yes	Yes
wikipedia.org	14	No	No
xnxx.com	8	Yes	Yes
xvideos.com	8	Yes	Yes
youtube.com	15	Yes	Yes

Table 4.2. List and description of the sites in the dataset.

Table 4.2 also shows information about accepted cookies. In fact, not all sites accepted cookies because it is not necessary in all of them; in some sites the banner to accept cookies is absent or not invasive.

In this dataset the histogram 4.2 of Speed Index becomes more like a bimodal distribution. The peak moved from 8 to 4 seconds, this may be due both to the change of sites but also to the fact that caching is used in these experiments,

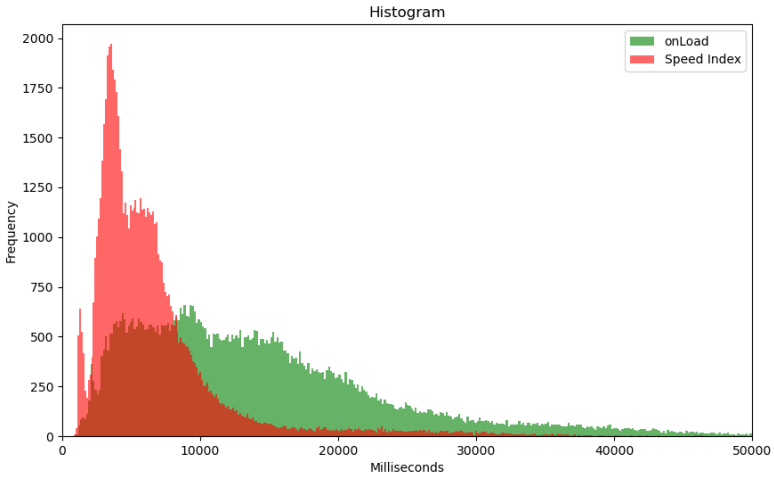

Figure 4.2. Distribution of onLoad and Speed Index of the dataset with cache and accepted cookies.

speeding up the visits.

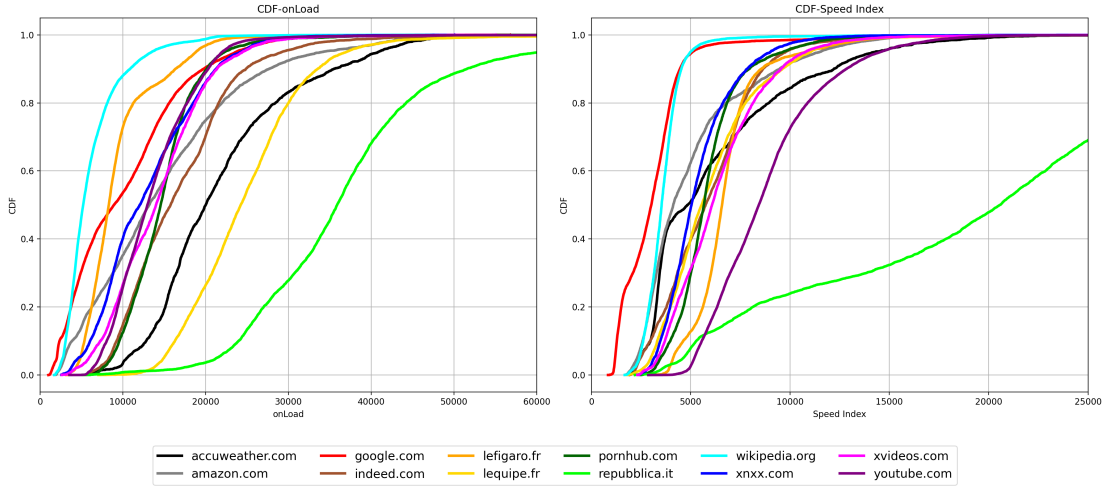

Figure 4.3. Distribution of onLoad and Speed Index of the dataset with cache and accepted cookies.

Another interesting analysis is to see the value of Speed Index and onLoad on the different sites. The CDFs in Figure 4.3 summarize the variation of onLoad and Speed index respectively in the different websites. In many sites, data are almost normally distributed despite the fact that there are several internal pages

for each site. In other website, such as *google.com* or *lefigaro.fr*, a knee is present, marking the difference between different pages. Summing up, different websites have different distributions.

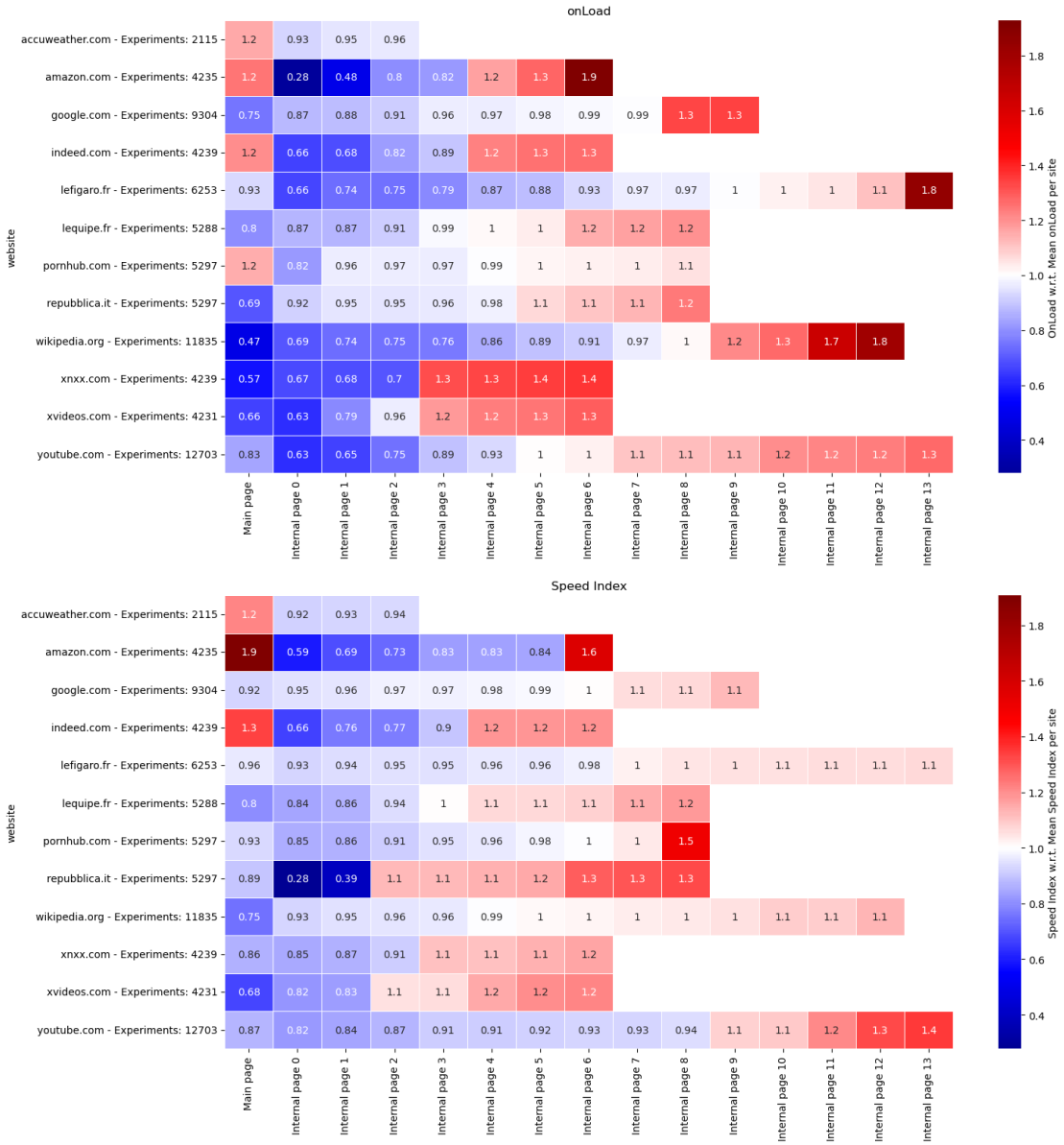

Figure 4.4. Heatmap: Summary of onLoad and Speed Index variation across different internal pages for each website. The data come from the dataset with cache and accepted cookies.

To further explore the variation in our metrics within a given website, Figure

4.4 summarize this variation across different internal pages. Each row describes a specific website and each column is an internal page of that website. The first column is always the main page (e.g for google is *www.google.com*). The value in a cell is:

$$V = \frac{\mu_{URL}}{\mu_{website}} = \frac{\frac{1}{N_{URL}} \sum_{i=1}^{N_{URL}} x_i}{\frac{1}{N_{website}} \sum_{i=1}^{N_{website}} x_i}$$

where μ_{URL} is the mean value of onLoad or Speed Index in an internal page of a website, $\mu_{website}$ is the mean value of onLoad or Speed Index in a website; $n_{website}$ is the total number of experiments on that website, n_{URL} is the total number of experiments on that site's URL. x_i is the onLoad or Speed Index value of one experiment.

So for instance the mean value of onLoad in the main page of amazon is 1.3 times slower than the average of that site. The first column always represents the main page of the site, then the internal pages are sorted by increasing value. *Pornhub.com* and *lequipe.fr* have little variation of onLoad between pages, which makes us think that a single model for the whole website might work. Instead, the pages of amazon have a high variance, particularly an internal page seems to be very light compared to the others. *Wikipedia.org*'s onLoad is the one with the most variation across pages, this makes life complicated for the model since within the same site there can be different behaviors.

As explained in Section 2.2.2, onLoad measures the time at which all bytes of payload have been received, so the more heavy objects a page has the higher the onLoad value will be. Images and videos are to be considered as heavy objects. Instead, Speed Index takes into account only the visible parts of a webpage to be displayed. Therefore, in pages without moving elements Speed Index is more stable and less subject to change.

4.3 Parallel dataset

The parallel dataset consists of browsing 2 websites: *lequipe.fr* and *pornhub.com*. In this dataset the navigation on the two websites is done in parallel. There is no form of dependence between the two visits, two scripts run in parallel, each script iteratively runs a browsertime docker image. For both *lefigaro.fr* and *pornhub.com* various URLs are available, so a single URL is selected for each run of the docker image.

In total, we have 12k active experiments, distributed over the two websites. Also in this dataset the browser cache is used and the cookies of *lequipe.fr* are accepted. *Pornhub.com* does not have an invasive banner and therefore the cookies are not accepted. This dataset is intended to get even closer to the real case, by having the flows from the two different sites mixed with each other.

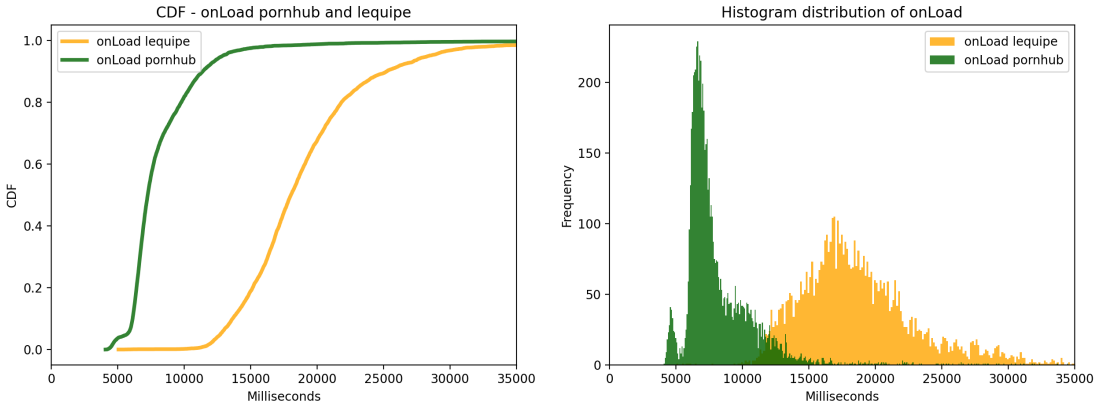

Figure 4.5. Cumulative Distribution Functions and histograms of onLoad and Speed Index of the parallel dataset.

In Figure 4.5 we have the distribution of the onLoad on this dataset. The distribution of *lequipe.fr* onLoad is almost normally distributed. Meanwhile, *pornhub.com* has a small peak at 4.5 seconds because in few occasion the main page is faster. Then it is right skewed distributed, with a very high peak.

4.4 Pipeline workflow

Starting from the previously created *final dataset*, the Machine Learning pipeline workflow (summarized in Figure 4.6) contains 4 steps:

1. **Feature Extraction**: The extracted features depend on the traffic considered: TCP features and UDP features. From the statistics generated by Tstat we extract several values that we will provide as input to our model. Several timing information are considered. This operation introduces us a parameter, so we will study the best value to assign to it.

2. **Feature Selection**: For TCP there are 52 features, not many but not all of them useful for our model. An algorithm, Recursive Feature Elimination (RFE), is then used to reduce the number of features. At the end of this step, the top 20 features are selected.

3. **ML Model**: We focus mainly on regression models such as Linear Regression (LR) or Random Forest (RFR). We also try classification models like Support Vector Machine (SVM) or again Random Forest but in classification version (RFC).

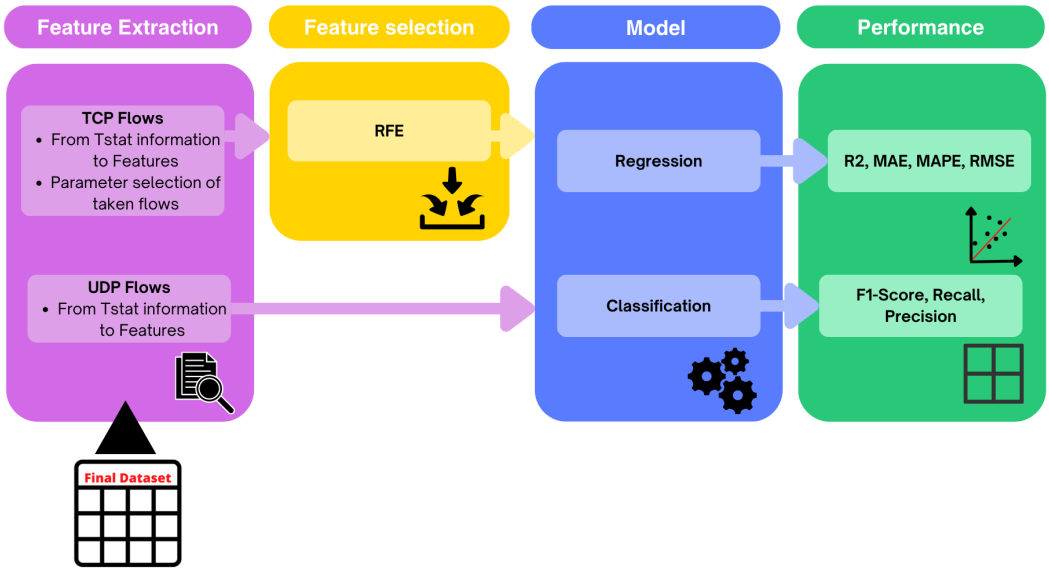

Figure 4.6. Machine learning pipeline to predict QoE.

4. **Performance**: Here we see whether our models are doing well or badly. We will try different experiments on the different datasets and then make considerations on these.

4.5 Feature Extraction

The dataset obtained by merging the active data and passive data contains the data captures by Tstat and the information about the labels, website and URL extracted by browsertime. While for the latter there is no need to reprocess the data, passive data are raw and therefore need to be processed.

Since browsertime closes the browser and the running docker container stops when the page loads we cannot use all the flow information related to the end of the flow, such as the duration or the time of the end of flows. This is because the duration of a flow could be equal to one of our metrics, the onLoad, only on our test set. In the real case a user may have a page open for an indefinite time. So in our model we use only information related to the beginning of the flow.

4.5.1 TCP flows

We now focus on TCP flows and then to udp flows. For each experiment only the first n TLS (with *s_port* equal to 443) flows are taken, where the choice of n_{flows} will be determined later by comparing several possible values. From this row we take the following column described by the Table 2.1:

- **First**: It is the absolute time of first packet of flow, it is an epoch, the time in milliseconds since January 1, 1970. Since epoch cannot be used directly by our model it is reworked: the epoch of the first flow of the experiment is saved so each epoch of all flows in the experiment is subtracted by this value. In this way it is obtained the time difference in milliseconds between the start of the first flow with the start of all other flows.

- **s_first** and **c_first**: Respectively the elapsed time from the first packet to the first server packet with payload and to the first client packet with payload. For each flow the difference between s_first and c_first is taken, obtaining the time from the first client packet with payload and the first server packet with payload.

- **c_rtt**: Tstat gives us several statistics on RTT, computed measuring the time elapsed between the data segment and the corresponding ACK. Minimum, maximum, average and standard deviation are calculated. c_rtt is approximable to the ground rtt, the time between the ground station and the web server. s_rtt can be approximated to 0, since it is the time between the PEP and the passive probe, which are ideally located in the same place.

- **s_ttl**: Minimum and maximum Time To Live are calculated, only on server side, it is the number of hops that a packet is set to exist inside a network before being discarded by a router. Each packet has a TTL count that starts from a certain value (255 in our case), every time a router receives a packet, it subtracts one from the TTL count; if this count reaches 0 the packet will be discarded by the router. The hops between the provider's PEP and the web server are measured, doing $255 - count_{TTL}$. In this case, c_ttl is not used because it is always equal to 254, between the PEP and Tstat probe there is only one hop.

- **s_sit and c_sit**: They are nine values for the client and nine for the server. They are the interarrival time between the packet and the previous packet. Two lists (one for the client sit and one for the server sit) are created with all values of all the flows in the experiment. Therefore, the lists have a length of $9 \times n$ elements, with n equal to the number of TLS flows taken. These lists are filtered, values equal to 0 are removed. But this information is

saved, **s/c_sit_0_count** is the number of sit equal to 0. Then, on this two lists several functions are computed: minimum, maximum, average, standard deviation and percentiles (25, 50, 75, 90).

In total when $n_{flows} = 5$, the number of features is 52, later we will see how to decrease it.

4.5.2 UDP flows

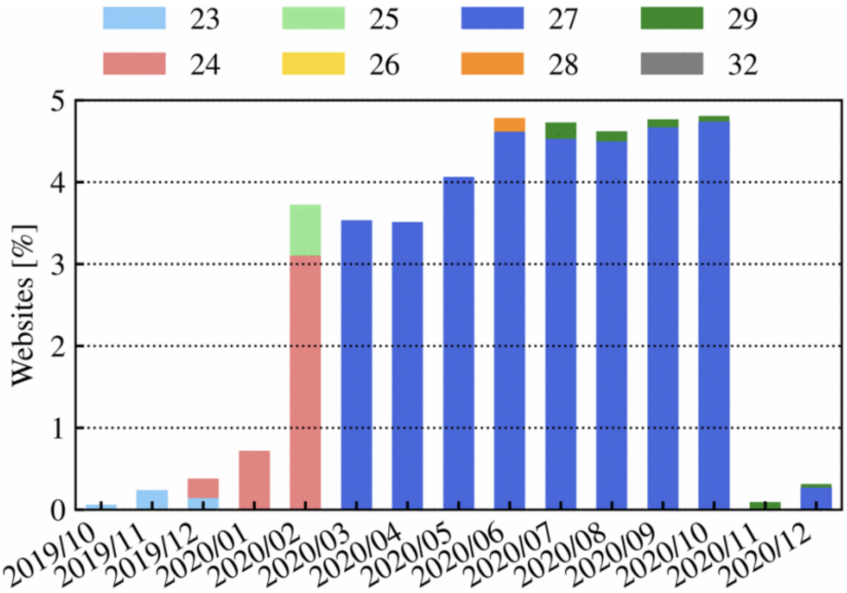

Figure 4.7. Percentage of websites in httparchive that announce support to http/3, separately by IETF draft. Taken [23].

Although TCP still remains the most widely used transport layer protocol on the Web, UDP is gradually gaining in importance thanks to the HTTP/3 protocol (Figure 4.7). HTTP/3 adopts a more efficient header compression schema and replaces TCP with QUIC (Quick UDP Internet Connections), is a multiplexed and encrypted transport protocol that is designed to provide low-latency and secure communication between clients and servers over the internet. It operates over UDP (User Datagram Protocol), which allows it to avoid some of the congestion and delay issues that can arise with TCP (Transmission Control Protocol). The use of QUIC in HTTP/3 helps to improve the speed, reliability, and security of web communications [23].

To take only QUIC flows, UDP flows are filtered by s_port, in fact QUIC uses port 443 as the default port for compatibility with existing web infrastructure (it is the default port for HTTPS traffic). Similar to what was done with TCP flows, only the first n QUIC flows are taken for each experiment. Since UDP flows have less information rather than TCP flows, a lower number of features are obtained. The same features as for TCP flows remain except for: First, c_rtt and s_ttl.

4.6 From the Regression problem to the Classification problem

There are several ways to convert a regression problem to a classification one:

- **Binning**: One approach is to bin the continuous numerical value into a set of discrete categories or intervals. For example, if the target variable is a numerical value representing the onLoad, we can divide the onLoad range into several bins such as "0-10000", "10000-20000", "20000-30000", and so on. Then, we can treat the problem as a classification task where the objective is to predict the appropriate bin for a given input.
- **Thresholding**: Another approach is to apply a threshold to the continuous numerical value, and then map the values above the threshold to one class, and the values below the threshold to another class. For example, again, if the target variable is still the onLoad metric, we can define a threshold such as 30000ms and then predict whether the onLoad of a visit is above or below this threshold.
- **Ranking**: We can convert a regression problem into a ranking problem by assigning a rank to each data point based on its numerical value. For example, we can rank the onLoad values from highest to lowest, and then predict the rank of a given input.
- **Log transformation**: Sometimes, a regression problem can be transformed into a classification problem by applying a logarithmic transformation to the target variable. This can be useful when the target variable is skewed towards higher values, and the logarithmic transformation can help to make the distribution more symmetrical.

In this work it is chosen the binning approach. In order to select the classes different bins need to be created. For this purpose, dataset is split into three class according to the onLoad value for each website. Percentiles are used: a percentile is a statistical measure used to indicate the value below which a given percentage of observations or data points in a distribution fall. In other words, it represents the point below which a certain proportion of the data lies.

- **Class 0 (good):** All experiments with an onLoad between 0 and the 50th percentile belong to this class.
- **Class 1 (medium):** All experiments with an onLoad between the 50th percentile and the 75th percentile belong to this class.
- **Class 2 (poor):** All experiments with an onLoad between the 75th percentile and the maximum value belong to this class.

4.7 Number of flows in an experiment

The first parameter examined is the number of the first flows taken in the feature extraction. The lower this parameter is then the less information will be captured, so it is more difficult for the model to have a correct prediction. On the other hand, not all experiments have a high number of flows, so in the training and testing dataset, experiments where n is greater than the number of flows within it are eliminated, thus resulting in smaller datasets. In the real case scenario the problem becomes even more complicated, in fact by taking more flows there is more risk of also collecting flows not belonging to that visit. This then worsens the performance of the model in the real case scenario.

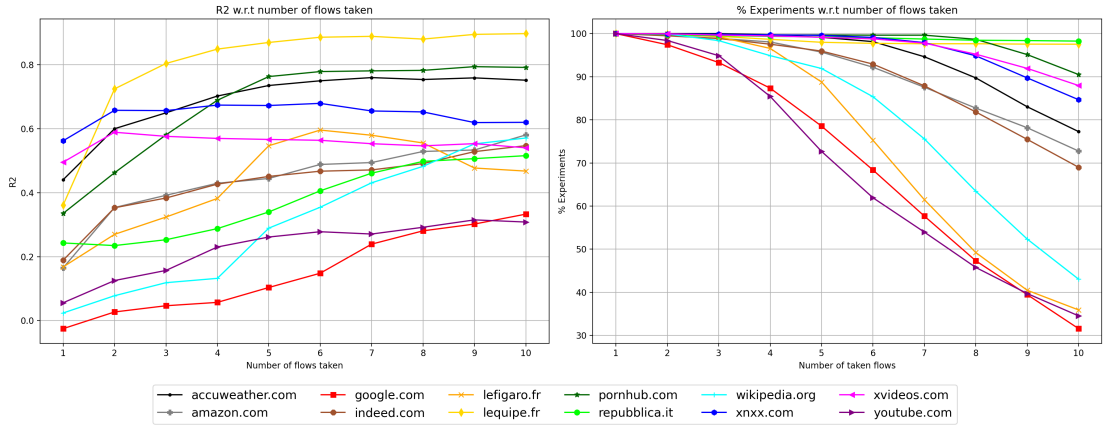

Figure 4.8. On the left the performance of the model using r2 as metric and on the right the number of experiments taken w.r.t. the number of the flows taken in features extraction. % Experiments is the percentage of the available experiments.

In Figure 4.8 there is the trend of the performance of the model (R-squared metric) as the number of captured flows increases on the different sites. For each site there is a model, trained and tested on the data of that site of the dataset

with cache and accepted cookies. The target value used is the *onLoad* metric and the model is validated on 3 fold. The Figure on the right depicts the number of the experiments eliminated on the dataset with cache and accepted cookies by taking the first n flows; it is the percentage of the experiments used, calculated as $\%Experiments = \frac{Experiments_n}{Experiments_1}$, where $Experiments_n$ is the number of experiments when n flows are taken.

In general, this results are consistent with what was expected: as the number of flows increases, performance increases but the number of experiments decreases. In *repubblica.it* and *lequipe.fr* the number of available experiments does not decrease, so for each experiment there are at least more than 10 flows available. *Google.com* and *youtube.com* have few TCP flows since they make massive use of QUIC. Therefore, with 10 flows taken more than 60% of the experiments are eliminated. Instead, *wikipedia.org* has few flows because it is a lightweight site as we also saw from the CDF in Figure 4.3 where it has very low onLoad values. For sites with a larger number of flows in each experiment, the model performance remains fairly constant from a number of flows of 5 and up. The trend of the curves of models performance is monotonically increasing for almost all sites, except for *lefigaro.* which has a reduction in performance due to the reduction in the number of samples.

So it is needed to find the right trade-off between performance and the number of experiments. Thus, a number n equal to 5 is selected.

4.8 Feature Selection

In the case of TCP flows, and where the number of flows in an experiment is equal to 5, the number of features is 52, many of which may be redundant or even irrelevant. It would be very useful to remove these features and understand how much each feature affects the outcome.

In this context, **Recursive Feature Elimination** (RFE) is used to remove unimportant features, and also Random Forests are used to assign importance scores to each feature.

Figure 4.9 shows the different performances of a Random Forest Regression model, validated with a 3-fold cross-validation method, with different number of features on the dataset with cache and accepted cookies. The metric used to evaluate the model is the R-squared. Since a 3-fold approach is used three results are obtained, the boxplot shows the minimum, the median, the maximum and the first and third quartiles. From what we can see, initially when features are removed, the performance of the model does not change until 20 features. So 20 features are more than enough for a good result, but even with a number of 10 there is not a big loss in the result. Thus, most of the features turn out to be

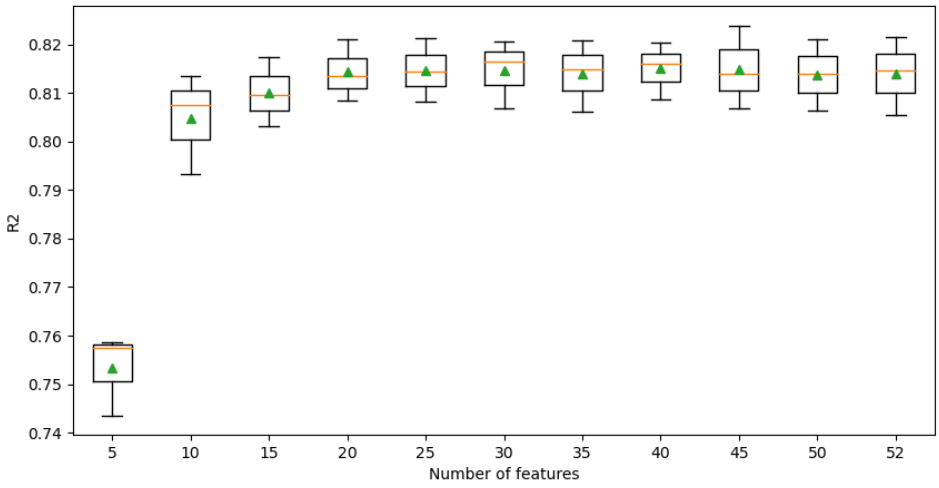

Figure 4.9. Recursive Feature Elimination, model performance for the different number of features selected.

irrelevant.

As reported in Figure 4.10, a rank of feature importance is shown by using a Random Forest Regressor model with a 3-fold validation and the dataset used is still the second one. *s_ttl_min_list_0* is the most important feature, it is the minimum Time To Live observed during the first flow of the experiment. Since each web server is in a specific position, it has a different TTL. Indeed it is particularly important for the first flow which is the flow with the *Server Name Information* equal to the contacted web server.

Also *s_sit_max* has a high importance, it is the maximum interarrival time of packets sent by the server and received by the client. Since the onLoad is a time metrics, it makes sense that it is correlated with the maximum time between packets, The longer this time, the greater the delay can be.

In Figure 4.11 we can see a heatmap for Pearson correlation between features. A heatmap for Pearson correlation is a visualization technique used to represent the correlation between two variables in a graphical form. It provides a color-coded matrix to represent the degree and direction of the correlation between each pair of variables. Pearson correlation is a statistical measure that assesses the linear relationship between two continuous variables. It takes a value between -1 and 1, where -1 represents a perfect negative linear relationship, 0 represents no linear relationship, and 1 represents a perfect positive linear relationship. In this Figure we can see that features that are generated from the same data type show a significant correlation, such as *first_rel* features or *c_sit_max* and *c_sit_std*.

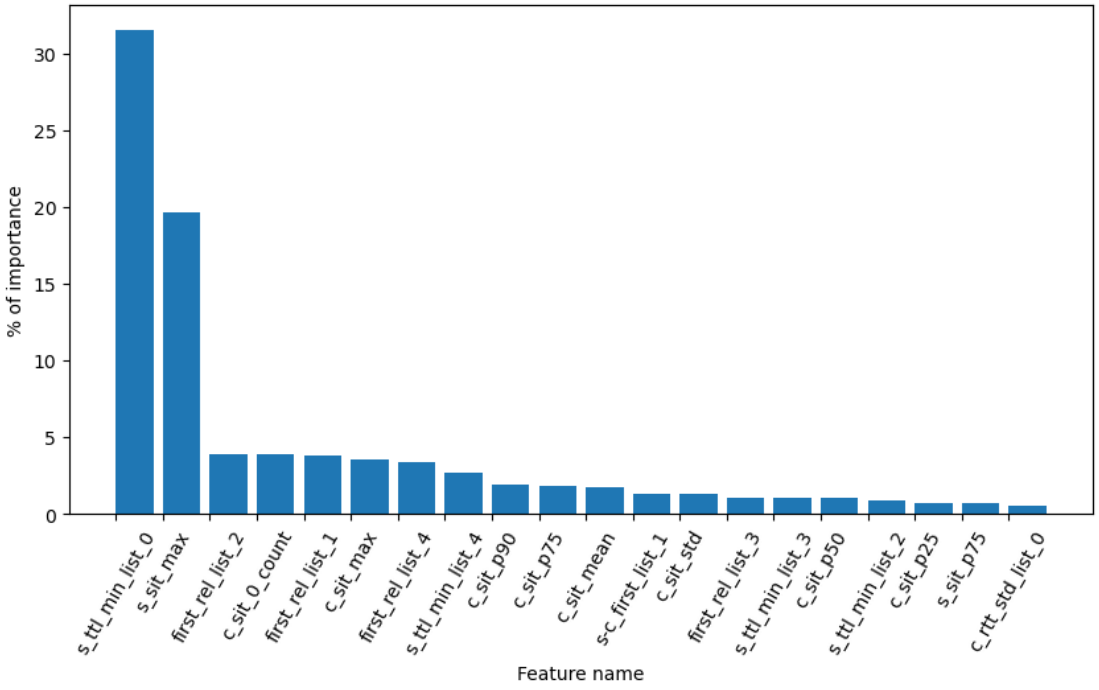

Figure 4.10. Rank of feature importance on an Random Forest Regressor model. The RFR model is trained on the dataset with cache and accepted cookies.

The correlation is a linear one, so this heatmap can not tell us everything, even if a feature is linearly uncorrelated with the target metric it may contain useful information for the machine learning model.

Moreover, the two target metrics shows a positive correlation with the majority of the features, but they show a negative correlation with *c_sit_0_count*; in fact this feature counts the number of interarrival time equal to 0, and the higher this number is, the faster a visit might be, thus resulting in a lower value of onLoad or Speed Index. Consistent with what we is seen in Figure 4.10, *s_ttl_min_list_0* has the maximum correlation (0.4) with the onLoad metric. In general, all features except for the *first_rel* ones have sufficient correction with both the target metrics.

4.9 Models Parameters

Choosing machine learning parameters is an important step in building an effective and accurate machine learning model. In fact, most machine learning models have parameters that when set correctly improve the performance of the algorithm itself. Hyperparameters are parameters that are not learned by the model during

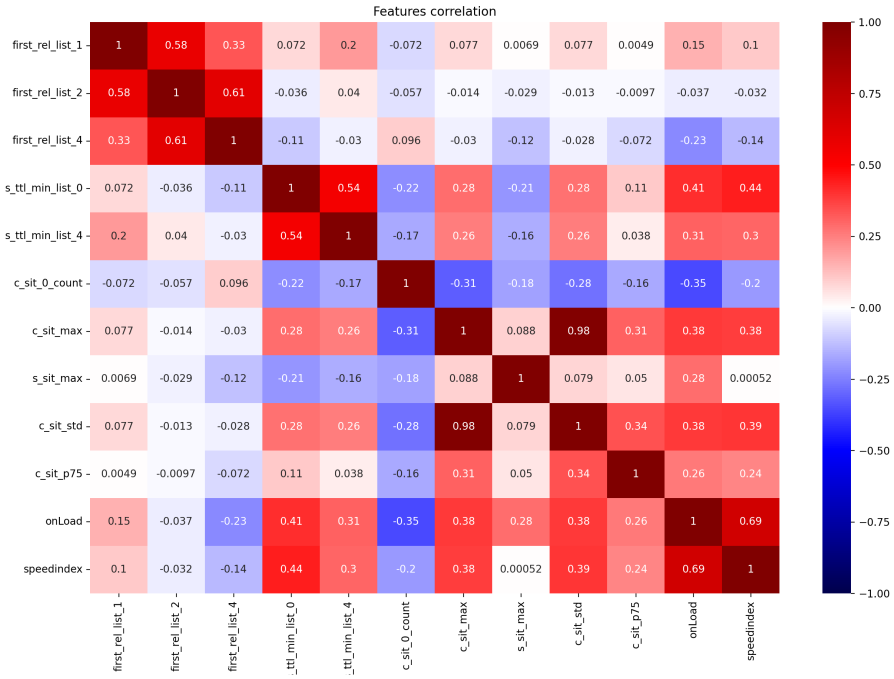

Figure 4.11. Heat map showing the Pearson correlation coefficient between 10 features selected with RFE and the two target metrics.

training, but instead are set manually before training.

In this context, Grid search technique is used. It search for the best combination of hyperparameters for a machine learning model. In a grid search, a grid of possible hyperparameter values is created, and the model is trained and evaluated for each combination of hyperparameters in the grid. The hyperparameter combination that produces the best performance on a validation set is chosen as the final set of hyperparameters for the model.

In a grid search, a grid of possible hyperparameter values is created, and the model is trained and evaluated for each combination of hyperparameters in the grid. The hyperparameter combination that produces the best performance on a validation set is chosen as the final set of hyperparameters for the model.

In building a Random Forest Regressor or Classifier model, the best combination of hyperparameters is sought. A grid search with cross-validation is created and it includes the following hyperparameters and their possible values:

- max_depth: 1, 2, 4, 8, 16, None;
- max_features: "auto", "sqrt", "log2";

- min_samples_leaf: 1, 2, 4;
- min_samples_split': 2, 4, 8;
- n_estimators': 10, 50, 100, 200, 400.

Instead, for Support Vector Machine the hyperparameters and their possible values are:

- C: 0.1, 1, 5, 10;
- coef0: 0.01, 0.1, 1, 10;
- gamma: 'auto', 'scale'.

The grid search train and evaluate the model for each combination of hyperparameters using cross-validation. Then the best hyperparameters are chosen for the different models.

Chapter 5

Metodologies and Results

In this chapter the results of the different models are presented. The chapter is divided into three sections, in the first a description of the results of the models applied on the no cache dataset, together with some considerations, are reported. The second one instead contains the results of the models applied on the dataset with cache and accepted cookies. The last section describe the results on the parallel dataset, the one closest to the actual case.

5.1 No cache dataset

As first results, regression models (Random Forest Regressor and Linear Regression) are reported. OnLoad and Speed Index are considered as target variable by building one model for each of them and a stratified 3-fold approach is used for validation. Stratified sampling is a sampling technique where the samples are selected in the same proportion as they appear in the population. In this case, for each fold, data for a given site are distributed equally between the train and the test set.

Target metric	Model	R2	MAE	MAPE	RMSE
onLoad	RFR	0.86	2245.28	13.15%	5049.78
	LR	0.44	6054.76	40.25%	10228.35
Speedindex	RFR	0.91	1351.96	11.64%	3084.08
	LR	0.55	3845.00	35.12%	6939.83

Table 5.1. Results of regression models on the no cache dataset. Both models (Random Forest Regressor and Linear regression) are used for both metrics (onLoad and Speed Index). A stratified (per site) 3-fold validation approach is used.

Random Forest Regressor was found to offer the best compromise in terms of R-squared, Mean Absolute Percentage Error and Root Mean Square Error. The superiority of this algorithm over Linear Regression lies in the ability to capture non-linear relationships between the dependent and independent variables, it is less sensitive to outlier and it can often provide better accuracy than Linear Regression, especially for datasets with complex relationships between the variables.5.1. With both target metrics, RFR performs very well having an R2 Score of 0.86 for onLoad and 0.91 for Speed index. Also MAPE, that is easier to understand than the other metrics, is very low, showing a good performance of the model.

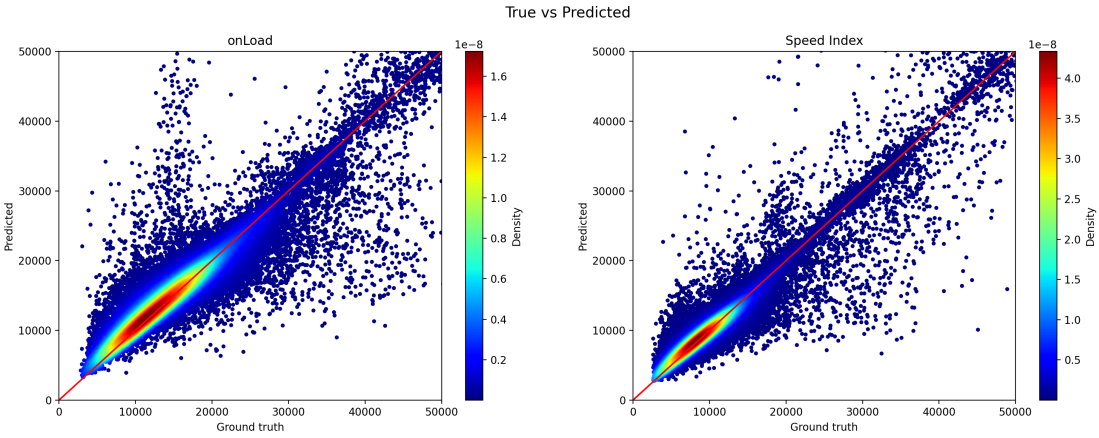

Figure 5.1. True vs predicted density plot, result of regression models on the no cache dataset. The model used is a Random Forest Regressor and it is validated through a stratified (per site) 3-fold validation.

The density plots in Figure 5.1 show the predicted vs. actual values for the two regression models on the two target metrics. The true value is the actual value of the onLoad or Speed Index, while the predicted value is the value estimated by the models. The accuracy of the prediction can be assessed by comparing the position of the point relative to the line in red. In fact, all points lying on this line have the true value and the predicted value equal, the further the point is from the line, the larger the error will be.

In addition, point density information is also reported. A **Gaussian KDE** technique is used for this purpose. It is a specific type of Kernel Density Estimation (KDE) that uses a Gaussian kernel to estimate the probability density function. The kernel density at any point is calculated by summing up the values of all the Gaussian kernels placed at each data point. The final estimate of the density function is obtained by normalizing the sum of the Gaussian kernels by dividing by the total number of data points and the volume of the kernel. In this case, a

red color means high density, blue color means low density.

Website	Experiments	Target	R2	MAE	MAPE	RMSE
accuweather.com	665	onLoad	0.33	2229.93	16.01%	2626.38
		Speed Index	0.76	550.32	6.31%	765.35
amazon.com	683	onLoad	0.41	1587.85	11.79%	2980.38
		Speed Index	0.43	1014.83	9.79%	1649.01
google.com	212	onLoad	0.80	443.72	5.64%	606.79
		Speed Index	0.67	225.01	5.40%	368.92
indeed.com	700	onLoad	0.90	564.55	4.26%	785.24
		Speed Index	0.94	345.81	3.20%	470.51
lefigaro.fr	208	onLoad	0.57	1678.77	10.10%	2140.76
		Speed Index	-0.00	3359.26	38.60%	4203.67
lequipe.fr	666	onLoad	0.88	573.96	4.00%	838.85
		Speed Index	0.45	564.23	12.19%	835.94
pornhub.com	712	onLoad	0.92	685.63	5.24%	1374.05
		Speed Index	0.75	413.95	5.71%	788.82
repubblica.it	680	onLoad	0.88	565.56	3.29%	971.63
		Speed Index	0.81	462.74	5.36%	662.38
wikipedia.org	132	onLoad	0.71	172.92	4.31%	268.01
		Speed Index	0.67	168.76	4.69%	261.20
xnxx.com	688	onLoad	0.94	385.18	2.71%	619.96
		Speed Index	0.90	388.04	4.00%	637.46
xvideos.com	699	onLoad	0.92	447.93	2.95%	844.99
		Speed Index	0.84	494.84	5.29%	785.52
youtube.com	216	onLoad	0.48	1785.59	11.54%	2302.63
		Speed Index	0.75	495.13	5.86%	688.53

Table 5.2. Results on the no cache dataset. For each website a model is created and trained and tested on the data of that website. The model used is Random Forest Regressor and the target metrics are both onLoad and Speed Index. Each model is evaluated through a 3-fold validation approach.

Another important experiment is to see how a model made for only one site behaves. Therefore one model per each site is built, once again with onLoad and Speed Index as target variable. The validation used is a 3-fold approach, not stratified in this case since for each model the dataset contains only one site. In particular, it can be seen in Table 5.2 that the performance of the model depends strongly on the site. In some sites, such as *indeed.com* or *pornhub.com*, the model performs quite well on both the target metrics; but for other sites, like *amazon.com*,

it does not work very well, showing an R2 of 0.41 on the prediction of the onLoad.

5.2 Dataset with cache and accepted cookies

5.2.1 Regression

Like what was seen above, this dataset is collected using the browser cache and having accepted cookies. For this reason it contains fewer flows within an experiment and therefore less information are available making prediction more difficult.

One Model for all websites

Target metric	Model	R2	MAE	MAPE	RMSE
onLoad	RFR	0.76	3261.44	30.57%	5545.61
	LR	0.49	5615.41	55.11%	8051.17
Speedindex	RFR	0.63	1351.96	50.84%	4538.96
	LR	0.43	2841.38	35.12%	6939.83

Table 5.3. Results of regression models on the dataset with cache and accepted cookie. A unique model for all the website in our dataset is created. The model used is Random Forest Classifier and the target metrics used are both onLoad and Speed Index. Each model is evaluated through a stratified (per site) 3-fold validation approach.

Indeed, in Table 5.3 we can see that worse result are obtained. For both model and for both target metrics a lower R2 is obtained. Specifically, in the prediction of the Speed Index, Random Forest Regressor reaches an R2 of 0.63, much lower than that seen for the no cache dataset (0.91). Again, in both cases, LR gets worse results than RFR. Since onLoad metric seems to be more stable it is used as target variable from here on. Speed Index suffers with web pages with video or moving content making the value high. In Table 5.4, the same model as in Table 5.3 is used (where the model is RFR and target metric is the onLoad). However, in this case, the results are reported by site.

One Model per website

In Table 5.5 there are the results of how model behaves when trained and tested only on data from one site. Once again one model for each website is built and a 3 fold approach is used as validation. As for the more generic model, results are worse when compared with those of the no cache dataset. The model still works well on sites like *accuweather.com*, *lequipe.fr* and *pornhub.com* but suffers

Website	Experiments	R2	MAE	MAPE	RMSE
accuweather.com	1650	0.66	3533.30	16.22%	5469.03
amazon.com	3723	0.38	5445.82	64.09%	7783.18
google.com	3399	-0.10	4208.92	115.51%	5439.98
indeed.com	3761	0.33	3914.83	26.88%	6105.96
lefigaro.fr	5032	0.33	2414.88	27.46%	4443.41
lequipe.fr	4882	0.76	1510.64	5.87%	3624.23
pornhub.com	4261	0.74	1313.69	9.44%	2454.15
repubblica.it	4427	0.27	6989.46	20.71%	10616.29
wikipedia.org	4100	0.01	2173.08	45.54%	3307.49
xnxx.com	3880	0.66	2486.99	22.87%	3582.16
xvideos.com	3856	0.57	2670.65	21.65%	3830.98
youtube.com	3033	0.16	3249.12	23.74%	4893.63

Table 5.4. Results of regression models on the dataset with cache and accepted cookie. A unique model for all the website in our dataset is created. The model used is Random Forest Classifier and the target metric is the onLoad. The model is evaluated through a stratified (per site) 3-fold validation approach. Per site results are reported.

Website	Experiments	R2	MAE	MAPE	RMSE
accuweather.com	1659	0.73	3273.53	16.53%	4600.99
amazon.com	3808	0.44	5005.04	57.80%	7145.39
google.com	3479	0.08	3530.63	86.08%	4665.82
indeed.com	3796	0.49	3247.04	22.22%	4589.70
lefigaro.fr	5026	0.55	1878.77	20.57%	3066.82
lequipe.fr	4922	0.86	1246.27	5.16%	2642.32
pornhub.com	4282	0.75	1179.17	8.35%	2336.71
repubblica.it	4428	0.36	6547.28	20.02%	9916.32
wikipedia.org	4902	0.20	1811.44	37.11%	2630.79
xnxx.com	3915	0.68	2424.05	22.75%	3430.08
xvideos.com	3871	0.57	2721.06	22.75%	3717.80
youtube.com	3026	0.34	2914.87	23.34%	4172.74

Table 5.5. Results of regression models on the dataset with cache and accepted cookie. For each website a model is created. The used model is Random Forest Classifier and the target metric is the onLoad.

greatly with sites like *google.com* and *youtube.com*. The R2 of *wikipedia.org* is also very low, this could be due to the fact that, as we saw in Section 4.2, the internal

pages of the site are very different from each other and tend to be fast loading. Comparing Table 5.5 and 5.4 we can see that making one model per site improves the results of all sites. This is reasonable because within the same website we have less variation in the onLoad value than we would for different sites.

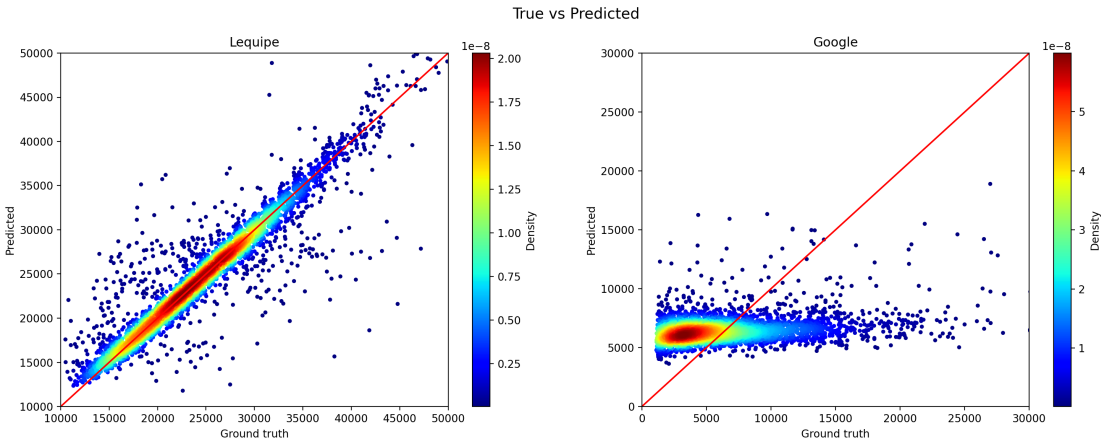

Figure 5.2. True vs predicted density plot, results of regression models on two sites: lequipe and google.

The density plot in Figure 5.2 shows the true vs predicted values for the regression models of two different sites: *lequipe.fr* and *google.com*. From this Figure we can clearly see that the model on lequipe data works very well, quite the opposite for google. In fact, on the latter the values predicted by the model are all between 5 s and 10 s, while the actual values have a wider range.

In Figure 5.3 we have the same heatmap shown in Section 4.2. This time, however, the value within the cell is the R2 recorded on that given URL. Internal pages are still sorted by increasing average onLoad value. Here we can see how in sites where performance is good, this is good for almost all internal pages; in contrast, sites such as *wikipedia.org* and *amazon.com*, which also showed significant onload variation within pages, perform very differently on different URLs and therefore generally much worse.

One Model per website: TCP + UDP

Since google and youtube massively use HTTPv3, which as a transport protocol uses QUIC, it is interesting to analyze also the UDP features extracted by the information of Tstat. In addition, the case where both types of flows are combined is also investigated. The target metric used is still the onLoad and the estimator used is still a Random Forest Regressor.

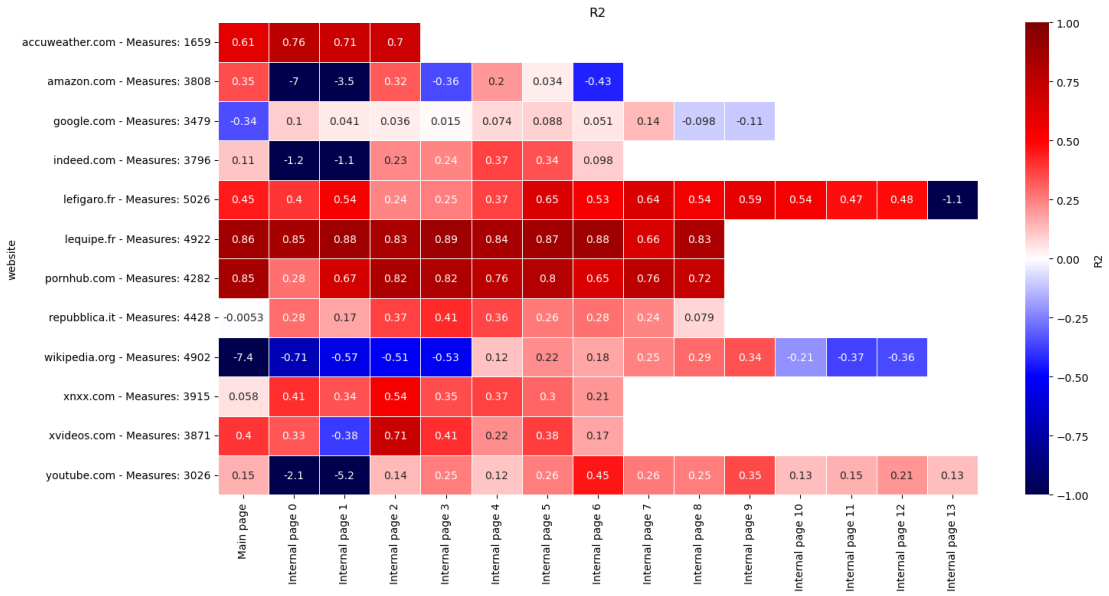

Figure 5.3. Heatmap: Summary of R2 across different internal pages for each website. The data come from the dataset with cache and accepted cookies. Results of regression models on the dataset with cache and accepted cookie. For each website a model is created. The used model is Random Forest Classifier and the target metric is the onLoad.

Website	**Type**	**Experiments**	**R2**	**MAE**	**MAPE**	**RMSE**
google.com	TCP	3479	0.08	3530.63	86.08%	4665.82
	QUIC	3377	0.11	3273.03	77.78%	4272.59
	QUIC+TCP	1996	0.13	3264.13	79.95%	4381.73
youtube.com	TCP	3026	0.34	2914.87	23.34%	4172.74
	QUIC	4590	0.46	2228.24	19.47%	3453.61
	QUIC+TCP	2004	0.55	2112.41	15.71%	3442.11

Table 5.6. Results of regression models on *google.com* and *youtube.com*, comparison between the use of the different type of flows: TCP, QUIC and TCP+QUIC. The used model is a Random Forest Classifier and the target metric is the onLoad. Each model is evaluated through a 3-fold validation approach.

As we can see in Table 5.6, the use of QUIC improves results for both websites. The use of only QUIC flows is the best model in terms of R2 for google, but it still does not get satisfactory results; indeed MAE, MAPE and RMSE shows that the model using QUIC and TCP data together brings a better result. Youtube

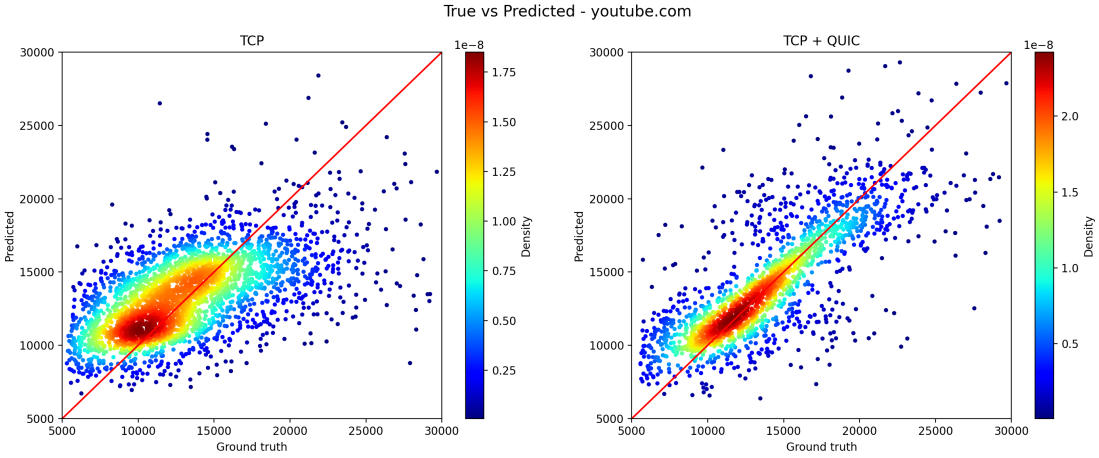

Figure 5.4. True vs predicted density plot, results of regression models on youtube data. Difference between the results obtained using only TCP data and using TCP and QUIC data combined.

benefits a lot from using QUIC data, the model goes from being poor to being good, it manages to achieve an R2 of 0.55 and a MAPE of 15.84%. In Figure 5.4, the dots in the right image tend to be closer to the red line showing good prediction ability. However, it is visible that with values that deviate enough from the mean value the predictor has more difficulty.

Past data for Training Future for Testing

Since websites change and update the content within them over time, it is important to check whether the model is efficient in predicting along time. Content updates are the main factor that can contribute to changes in a website, but these changes are often isolated to only one part of the page. A more complex case may be design changes, in fact a website's design may need to be updated over time to keep up with changing design trends, improve usability, or to better align with the brand's visual identity; this type of updates imply a big change in the behaviour of the site. In such extreme cases the model may work wrong since they may look like two totally different sites.

In this context, another experiment is done: for each website a model is built, training it on data belonging to a period from 23 December 2022 to 9 January 2023, and testing it on data ranging from January 24 to February 8 2023.

Results in Table 5.7 shows that for some sites, there is no clear deterioration in performance. In fact, if we compare the results with those in Table 5.5 we see that for some sites the R2 deviates little from the value in this table. However,

Website	Experiments	R2	MAE	MAPE	RMSE
accuweather.com	589	0.72	3469.48	16.47%	4596.22
amazon.com	1384	0.13	6885.32	86.72%	9103.91
google.com	2526	-0.86	4476.46	143.06%	5138.09
indeed.com	1448	0.04	4965.86	27.69%	6659.37
lefigaro.fr	1165	0.50	2096.91	19.60%	3323.89
lequipe.fr	1849	0.86	1162.45	4.51%	2250.96
pornhub.com	1548	0.81	927.59	5.52%	1889.89
repubblica.it	1747	0.22	6878.45	17.62%	10163.59
xnxx.com	1458	0.75	2084.79	17.88%	2860.52
xvideos.com	1451	0.59	2517.79	20.40%	3452.18
wikipedia.org	3489	-2.61	3004.75	82.17%	3541.37
youtube.com	1818	0.04	3550.46	31.59%	4817.84

Table 5.7. Results of regression models on the dataset with cache and accepted cookies. The dataset is splitted in two parts that differ from the collection period. The training is done on the data of older period, and the testing on the latest data. The used model is Random Forest Regressor and the target metric is the onLoad.

for sites like *amazon.com*, *indeed.com*, *google.com* and *wikipedia.org*, there is a big deterioration in the results. This suggests that these websites need update data for training, then the model, as time goes by, must perform training with new data.

Predict Unseen websites

Now let is see how new sites, that are not within the train set, perform. For each website, the training is done on the data of the dataset with cache and accepted cookies that contain all the experiments of all websites except that one. The testing is done on the data of that website. This test tells us whether we can apply the model (trained on our data) on generic sites, not within our dataset.

From results in Table 5.8 we can see that for most sites the model produces bad results and it is unable to generalize. The models cannot predict unseen websites. Thus, the idea of using the model on generic data in the network is not feasible, it is always necessary to train the model on the data of the site on which we want to make predictions.

The model seems to work decently only on porn sites. The reason is that they are very similar sites, so the data of one website is not so different from another site's data. Particularly, *xnxx.com* and *xvideos.com* have a server IP with the same IP location (*Netherlands Amsterdam Serverstack Inc.*) and the same Autonomous System Network (ASN): *AS46652 SERVERSTACK-ASN, US.*

Website	Experiments	R2	MAE	MAPE	RMSE
accuweather.com	1683	0.05	6120.40	24.21%	9105.45
amazon.com	3818	-4.41	20600.04	290.33%	22815.51
indeed.com	3815	-1.77	9814.35	72.29%	12595.95
lefigaro.fr	5033	-2.97	9497.16	132.11%	10835.81
lequipe.fr	4935	0.43	3261.90	12.27%	5650.68
pornhub.com	4289	0.56	2117.84	16.78%	3182.40
repubblica.it	4432	-1.06	14415.30	37.00%	18037.93
xnxx.com	3922	0.63	2719.20	26.72%	3762.17
xvideos.com	3879	0.44	3057.68	25.84%	4378.64
youtube.com	3033	-0.63	5163.36	35.98%	6806.94
google.com	3487	-2.10	8008.24	253.19%	9101.62
wikipedia.org	4913	-2.53	5436.59	133.84%	6350.87

Table 5.8. Results of regression models on the dataset with cache and accepted cookies. For each site the model is trained on the dataset containing all the sites except that one and tested on that site. The used model is Rnadom Forest Regressor and the target metric is the onLoad.

5.2.2 Classification

Since as an end result we need to know whether the QoE of navigation in that visits is good or not, the regression problem was simplified by turning it into a classification problem. This section analyzes the results of the classification models.

Based on numerical data of Table 5.9, taking only into consideration the weighted average of F1 score, some comparison can be made. Already from this first analysis it emerges that, among the two used models, the Random Forest Classifier provide better results for our research, and therefore the Support Vector Machines model must be excluded.

Moreover, the sites on that the models perform well are the same as those seen in the regression problem; in this case in the data of *pornhub.com* we have the best results and in *google.com* the worst results in terms of weighted F1 score.

Taking in consideration also Precision and Recall scores, we can see that in the class 0 (good) the models always perform better, this is because class 0 is the most populated class since, as it is explained in Section 4.6, is twice as large as the other two classes. Class 1 (medium) is the most challenging, since it lies in between the other two, misclassification errors are more frequent. In fact, in Figure 5.5 we can see that in the *lequipe.fr*'s results the misclassified errors come from neighboring classes. Errors from predicting poor class instead of good class and vice versa are very low. These errors come from neighboring onLoad values that have been

Website	Experiments	Model	F1 weighted avg	Precision			Recall		
				0	1	2	0	1	2
accuweather.com	1683	RFC	0.73	0.80	0.57	0.74	0.91	0.40	0.77
		SVM	0.65	0.69	0.65	0.66	0.91	0.25	0.64
amazon.com	3818	RFC	0.67	0.71	0.58	0.72	0.88	0.40	0.57
		SVM	0.66	0.68	0.62	0.72	0.92	0.37	0.51
google.com	3487	RFC	0.40	0.54	0.28	0.31	0.92	0.04	0.11
		SVM	0.38	0.52	0.48	0.41	0.98	0.01	0.05
indeed.com	3815	RFC	0.65	0.69	0.58	0.68	0.92	0.36	0.50
		SVM	0.60	0.65	0.56	0.67	0.92	0.37	0.33
lefigaro.fr	5033	RFC	0.68	0.71	0.57	0.76	0.87	0.35	0.70
		SVM	0.62	0.66	0.54	0.67	0.88	0.27	0.57
lequipe.fr	4935	RFC	0.76	0.82	0.60	0.81	0.91	0.50	0.76
		SVM	0.77	0.83	0.60	0.83	0.90	0.56	0.73
pornhub.com	4289	RFC	0.80	0.84	0.73	0.81	0.92	0.58	0.82
		SVM	0.77	0.83	0.62	0.79	0.87	0.64	0.69
repubblica.it	4432	RFC	0.59	0.64	0.45	0.71	0.89	0.24	0.46
		SVM	0.49	0.55	0.55	0.83	0.98	0.06	0.30
wikipedia.org	4913	RFC	0.45	0.56	0.31	0.34	0.79	0.16	0.22
		SVM	0.37	0.52	0.00	0.55	0.98	0.00	0.06
xnxx.com	3922	RFC	0.72	0.84	0.55	0.66	0.88	0.48	0.68
		SVM	0.72	0.85	0.52	0.66	0.86	0.53	0.62
xvideos.com	3879	RFC	0.66	0.77	0.50	0.61	0.83	0.37	0.67
		SVM	0.64	0.75	0.56	0.55	0.83	0.33	0.66
youtube.com	3033	RFC	0.52	0.60	0.34	0.59	0.79	0.09	0.71
		SVM	0.50	0.55	0.34	0.60	0.84	0.09	0.56

Table 5.9. Results on different sites of classification models on the dataset with cache and accepted cookies. Two models are used: Random Forest Classifier and Support Vecotor Machines. The metrics used are: the weighted average F1 score, the precision and the recall of the three classes. Each model is evaluated through a stratified (per class) 3-fold validation approach.

assigned to different classes. Different discussion regarding *google.com*, in fact here the model does not work. It mostly predict all class as good class, this leads to bad results for the other two classes.

In creating a single model for all sites in our dataset, it would not make sense to use the previously created classes. For exmaple, an experiment that belongs to class 2 of *wikipedia.org* could have an onLoad value of 10 seconds, *repubblica.it* instead could have a value of 50 seconds. So it would not make sense to consider them as belonging to the same class.

Therefore, the idea is to normalize the data by site and then create a single classification model. A Standard Scaler is used to normalize the data. It is a preprocessing technique used in machine learning to transform numerical data so that it has a mean of 0 and a standard deviation of 1. This technique is commonly used when working with datasets that contain features with different scales or units of measurement, which could cause problems for certain machine learning algorithms. The standard scaler works by subtracting the mean value of each

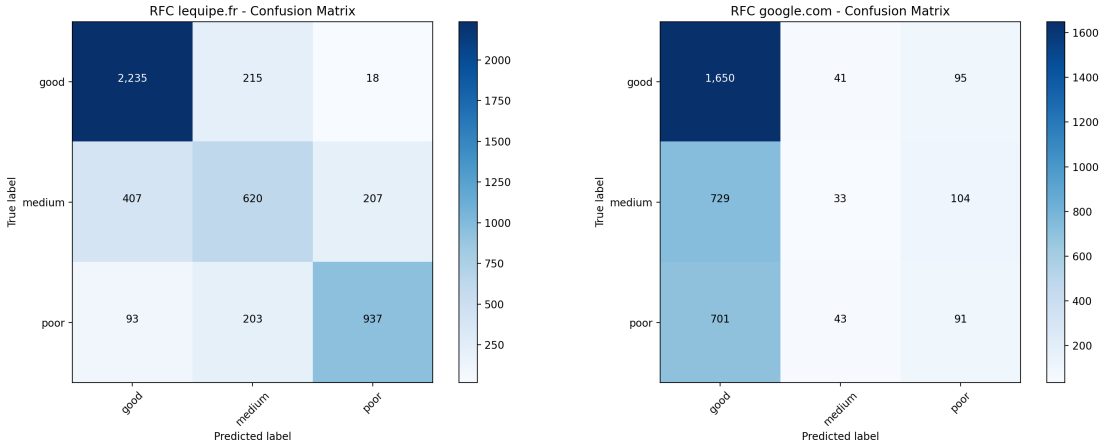

Figure 5.5. Confusion Matrix of the results of RFC applied on the data of *lequipe.fr* and *google.com* of the dataset with cache and accepted cookies. The model is validated through a stratified (per class) 3-fold approach.

feature from each data point, and then dividing by the standard deviation of the feature. This ensures that each feature has a similar scale and range of values, making it easier for machine learning algorithms to work with the data.

For each site, all features and also the onLoad are normalized, then the data of all sites are aggregated and the dataset is splitted into three class according to the normalized onLoad value. The splitting is done again through percentiles, as described in Section 4.6.

Model	**F1 weighted average**	**Precision**			**Recall**		
		0	**1**	**2**	**0**	**1**	**2**
RFC	0.65	0.70	0.56	0.68	0.89	0.35	0.57
SVM	0.62	0.68	0.49	0.65	0.88	0.31	0.50

Table 5.10. Results of classification models on the dataset with cache and accepted cookies. All features are normalized through a Standard Scaler. Also onLoad are normalized, before creating the classes. Two models are used: Random Forest Classifier and Support Vecotor Machines. The metrics used are: the weighted average F1 score, the precision and the recall of the three classes. Each model is evaluated through a stratified(per class) 3-fold validation approach.

In Table 5.10 we can see the results of the models Random Forest Classifier and Support Vector Machine applied on this dataset. Looking at the F1 score, the results are not so good. Also here, RFC outperforms SVM, proving that

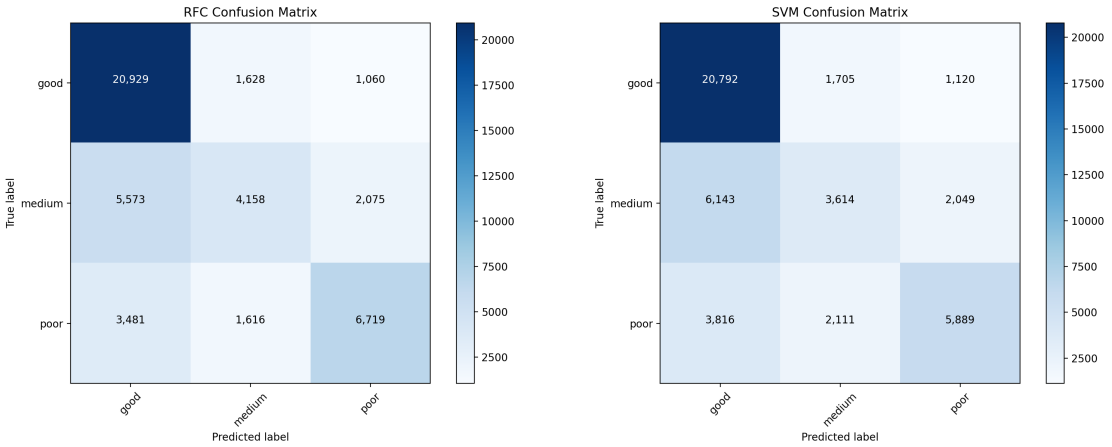

Figure 5.6. Confusion Matrix of the results of RFC and SVM applied on the dataset with cache and accepted cookies. The data are normalized through Sandard Scaler and also onLoad are normalized, before creating the classes.

models with decision trees are better for our purpose. In Figure 5.6 we can see the confuzion matrix of the two models. As for the models seen above, many of the misclassification errors are those that the moldels predict as good instead of medium or poor.

Turning problem into classification brought no benefit over regression, in fact the problem arises that two neighboring onLoad values can be assigned to two different classes. Also, for data where it is more difficult to make a prediction, the model almost always tends to predict the experiment as class 0. So there is no point in investigating further the above problem.

5.3 Parallel dataset

With this dataset we get even closer to the real case. There are only two website: *lequipe.fr* and *pornhub.com*, which are the sites with the best performance by the model in the previous tests. As described in Section 4.3,in this dataset the flows belonging to a single *lequipe.fr* experiment might actually be the flows from *pornhub.com*. This introduces inaccuracies in the data, since information of a flow of *pornhub.com* could be included in *lequipe.fr* data.

The idea is to filter flows based on the Server Name Idndication (SNI) field. It is an extension to the Transport Layer Security (TLS) protocol that allows multiple domain names to be served over HTTPS from the same IP address. SNI enables a client to indicate the hostname it is attempting to connect to, which allows the server to present the appropriate digital certificate for that hostname during the

TLS handshake.

For our purpose, The flows are not filtered based on the full SNI but only on the domain name. For example, if the SNI is *www.lequipe.fr* the domain name is *lequipe.fr*. For each site a list is created with the best domain names and then the TLS flows are filtered based on that list. In this context, as introduced in Section 3.5, Term Frequency-Inverse Document Frequency (TF-IDF) can help in selecting the best domain names. Starting from the dataset with cache and accepted cookies 4.2, a rank of the domain names with highest TF-IDF value is created for each website. In addition, it is also created a rank of the domain names with highest TF value. Then, taking into consideration only the n_{DN} with highest value, the flows of each experiment in the parallel dataset are filtered.

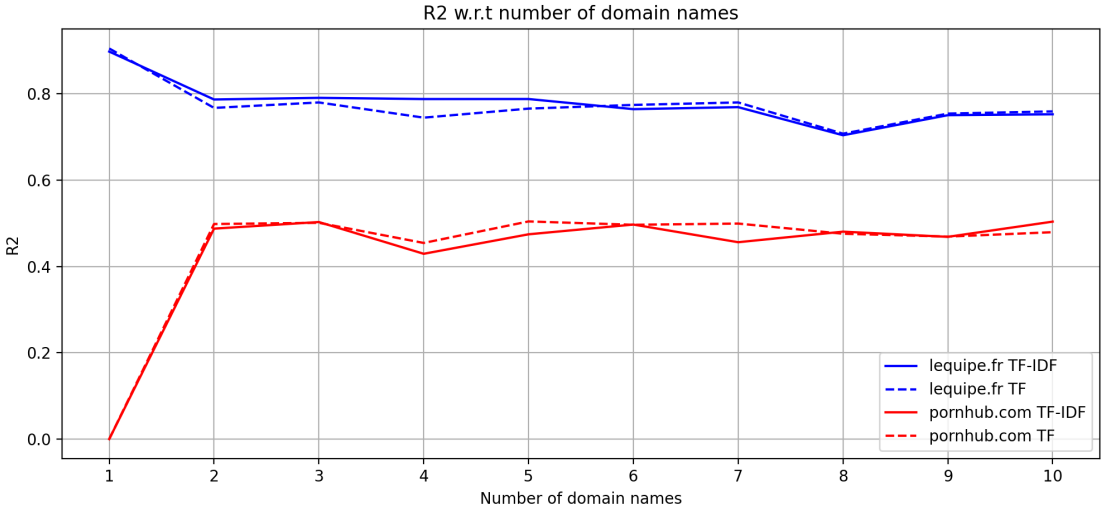

Figure 5.7. Performance comparison between different values of the number of domain names for both methods for selecting the best domain names (TF and TF-IDF). A Random Forest Regressor is trained and tested for each site through a 3-fold validation on the parallel test.

In Figure 5.7 several possible values of n_{DN} are tested. The domain name of the site is always taken: for example, in *lequipe* the domain name *lequipe.fr* is always taken. So, when $n_{DN} = 1$, in the case of *lequipe.fr*, the flows taken are only those with the domain name as *lequipe.fr*. In the case of *pornhub.com*, it is impossible to produce results with $n_{DN} = 1$, there are not 5 flows for each experiment with the domain name equal to *pornhub.com*. Even when adding the first domain name, the number of experiments with at least 5 flows are less than half. For this reason, it is chosen to use $n_{DN} = 3$ (belonging domain name + the two best domain names).

Table 5.11 reports a comparison between the performance of the model over different data. Results are also reported for the dataset with cache and accepted

Site	dataset	type	Experiments	R2	MAE	MAPE	RMSE
lequipe.fr	cached	RAW	4922	0.86	1246.27	5.16%	2642.32
		TF-IDF	4878	0.91	1101.36	4.58%	2123.67
	parallel	RAW	5473	0.61	1983.32	10.71%	3146.32
		TF	5384	0.74	1485.58	8.17%	2552.32
		TF-IDF	5382	0.77	1377.57	7.71%	2438.32
pornhub.com	cached	RAW	4282	0.75	1179.17	8.35%	2336.71
		TF-IDF	4278	0.82	1003.30	7.18%	1965.88
	parallel	RAW	6461	0.35	1375.72	16.19%	2552.76
		TF	6426	0.47	1244.14	15.05%	2240.43
		TF-IDF	6426	0.48	1236.09	14.96%	2229.75

Table 5.11. Performance comparison over different dataset (dataset with cache and accepted cookies and the parallel dataset) and over different ways to filter the flows for each website. The model used is the Random Forest Regressor, validated with a 3-fold approach.

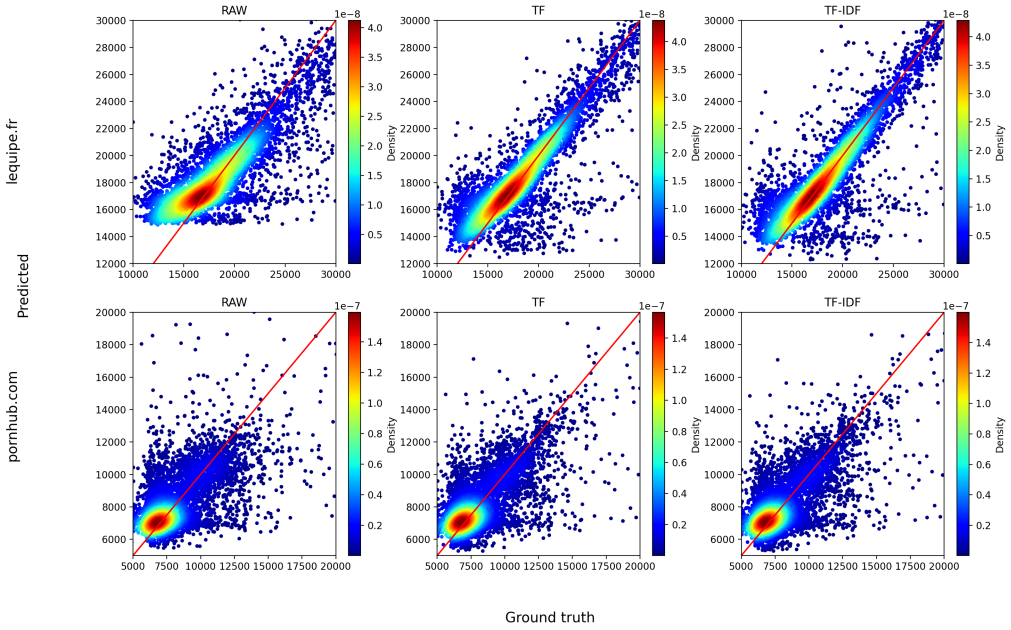

Figure 5.8. True vs Predicted density plot, results of regression models on the parallel dataset. Results obtained from the different ways of filtering flows through the value of the domain name. Random Forest Regressor is used and the target metric is the onLoad.The model is validated through a 3-fold approach.

cookies in case the flows are filtered through the value of the domain name. Already with this dataset we can see a slight improvement on all evaluation metrics for

both sites. Turning to the parallel dataset, there is a clear improvement in results, whether using TF or TF-IDF to filter the domain names. For *lequipe.fr*, TF-IDF seems to provide better results, while for *pornhub.com* TF and TF-IDF select the same best domain names.

In Figure 5.8 we can see the differences between *lequipe.fr* and **pornhub.com** results. Although the results of data filtered with TF and TF-IDF are very similar they visibly improve over those without filtering data. The model still works well on *lequipe.fr* but not as well on *pornhub.com*. Thus, proper filters may improve the performance and only on a subset of the website can good performance be achieved, based on the SNIs that characterize it.

Chapter 6

Conclusions

The limitations introduced by satellite navigation such as limited bandwidth and high latency create the need to measure the user satisfaction while browsing the web. In this context the work done is intended to create a mechanism that allows the ISP to know the quality of a given user's browsing from data passing through its network. The approach proposed involves the prediction of WebQoE metrics, as onLoad and Speed Index, from the network data after ISP ground station.

The works started by studying the several metrics used for measuring the costumer satisfaction in web browsing. Speed Index and onLoad, the widely used metrics in this context, are used because they have proven to be the most suitable and the most reliable in measuring the QoE. But Speed Index showed inaccuracy in measurement in the case of web pages with video or moving content.

To study and create machine learning models, a measurement collection has to be done. So a testbed for automatic collection of network and QoE data in web visits is created. Browsetime is used for collecting timing metrics of web sites since it is easy to deploy using docker and it is highly configurable. Tstat is used for collecting the network flows. For each visit on a website an experiment is created by aggregating different information of the belonging flows. A large number of features are extracted from this dataset, which will later be selected through feature selection techniques such as RFE.

The work then turned forward finding the preferred model for this type of dataset. Random Forest Regressor was found to offer the best compromise in terms of R-squared, Mean Absolute Percentage Error and Root Mean Square Error. The superiority of this algorithm over Linear Regression lies in the ability to capture non-linear relationships between the dependent and independent variables, it is less sensitive to outlier and it can often provide better accuracy than Linear Regression, especially for datasets with complex relationships between the variables.

In addition, the use of the browser cache led to a degradation of the result, many objects are downloaded directly from the cache, without taking them from the

network, decreasing the number of flows per experiment. Having less information it is logical that the model works worse, negatively impacting the results. Per-site results show that the proposed model works well with only some sites such as *lequipe.fr*, *pornhub.com* or *accuweather.com*, while for others it does not prove to be a reliable method for QoE prediction. *Google.com* and *Youtube.com* make heavy use of HTTPv3 protocol, which is why a model was created on UDP data. This models gives acceptable results on *youtube.com*, achieving a great improvement over the previous model created on TCP data.

Moreover, to approach the real case scenario tests have been done showing that the per-site model woks well on unseen data of the same site, in fact, the second dataset contains data from different time periods. On the other hand, in the case of the one model for all websites, the predictions of the data of a website that is not in the training data are inaccurate, thus the model can only be applied to sites within the dataset used in the training phase.

Finally, transforming the problem from regression to classification does not bring benefits, in fact it leads to worse results than the original problem. Classification models perform quite well on the sites where regression models perform well. However, even here the random forest proved to be the best model, outperforming SVM.

Despite the contributions that this work has made towards our understanding on the correlation between the SatCom network flows and the WebQoE, there remain several areas that require further investigation. One limitation of the study is the limited sample size, which may have influenced the generalizability of our findings. Future research should aim to replicate our study with a larger and more diverse sample to assess the validity and reliability of our results. For example, we need to check whether the model also works on the data of other users, who are in a different place from the client used to create our dataset.

Another area of future research could involve examining the long-term results of the machine learning models on new data. This would require continuing to collect data on the same sites in our dataset and testing the models on these.

A lot of work still have to be done, in order to allow ISPs to predict the QoE of the users, anyway, as always, the best has yet to come.

Bibliography

[1] Mohammed Alreshoodi and John Woods. «Survey on QoE\QoS correlation models for multimedia services». In: *arXiv preprint arXiv:1306.0221* (2013).

[2] Enrico Bocchi, Luca De Cicco, and Dario Rossi. «Measuring the Quality of Experience of Web Users». In: 46.4 (Dec. 2016), pp. 8–13. ISSN: 0146-4833. DOI: 10.1145/3027947.3027949. URL: https://doi.org/10.1145/3027947.3027949.

[3] Leo Breiman. «Random Forests». In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[4] *Browsertime Scripting:* https://www.sitespeed.io/documentation/sitespeed.io/scripting/.

[5] Patrick Le Callet, Sebastian Möller, and Andrew Perkis. «Qualinet White Paper on Definitions of Quality of Experience». In: *European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003)* (Mar. 2013).

[6] Pedro Casas and Sarah Wassermann. «Improving QoE prediction in mobile video through machine learning». In: *2017 8th International Conference on the Network of the Future (NOF)*. 2017, pp. 1–7. DOI: 10.1109/NOF.2017.8251212.

[7] *Docker:* https://www.docker.com/.

[8] Diego Neves da Hora, Alemnew Sheferaw Asrese, Vassilis Christophides, Renata Teixeira, and Dario Rossi. «Narrowing the Gap Between QoS Metrics and Web QoE Using Above-the-fold Metrics». In: *Passive and Active Measurement*. Ed. by Robert Beverly, Georgios Smaragdakis, and Anja Feldmann. Cham: Springer International Publishing, 2018, pp. 31–43.

[9] *How the browser cache works:* https://pressidium.com/blog/browser-cache-work/.

[10] Deutschmann J, Hielscher K-S, and German R. «Performance of modern web protocols over satellite links». In: *38th International Communications Satellite Systems Conference (ICSSC 2021)*. Vol. 2021. IET. 2021, pp. 154–158.

[11] Jo Kenneth. «Satellite Communication Network Design and Analysis». In: Artech house, 2011. URL: https://ieeexplore.ieee.org/document/9100166.

[12] Huan Liu. «Feature Selection». In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 402–406. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_306. URL: https://doi.org/10.1007/978-0-387-30164-8_306.

[13] Maria Papadopouli, Paulos Charonyktakis, Maria Plakia, and Ioannis Tsamardinos. «On User-Centric Modular QoE Prediction for VoIP Based on Machine-Learning Algorithms». In: *IEEE Transactions on Mobile Computing* 15 (Jan. 2015), pp. 1–1. DOI: 10.1109/TMC.2015.2461216.

[14] Daniel Perdices, Gianluca Perna, Martino Trevisan, Danilo Giordano, and Marco Mellia. «When Satellite is All You Have: Watching the Internet from 550 Ms». In: *Proceedings of the 22nd ACM Internet Measurement Conference*. IMC '22. Nice, France: Association for Computing Machinery, 2022, pp. 137–150. ISBN: 9781450392594. DOI: 10.1145/3517745.3561432. URL: https://doi.org/10.1145/3517745.3561432.

[15] Gianluca Perna, Dena Markudova, Martino Trevisan, Paolo Garza, Michela Meo, Maurizio M. Munafò, and Giovanna Carofiglio. «Online Classification of RTC Traffic». In: *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. 2021, pp. 1–6. DOI: 10.1109/CCNC49032.2021.9369470.

[16] *Satellite Communication scheme*. https://byjus.com/physics/satellite-communication/.

[17] *Sitespeed.io:* https://www.sitespeed.io/.

[18] Robertson Stephen. «Understanding inverse document frequency: on theoretical arguments for IDF». In: *Journal of Documentation* 60.5 (Jan. 2004), pp. 503–520. ISSN: 0022-0418. DOI: 10.1108/00220410410560582. URL: https://doi.org/10.1108/00220410410560582.

[19] Robert C. Streijl, Stefan Winkler, and David S. Hands. «Mean Opinion Score (MOS) revisited: Methods and applications, limitations and alternatives». In: *Multimedia Systems* 22 (2016), pp. 213–227.

[20] G S Suma, S Dija, and Arun T Pillai. «Forensic Analysis of Google Chrome Cache Files». In: (2017), pp. 1–5. DOI: 10.1109/ICCIC.2017.8524272.

[21] Alaa Tharwat. «Classification assessment methods». In: *Applied Computing and Informatics* 17.1 (Jan. 2021), pp. 168–192. DOI: 10.1016/j.aci.2018.08.003. URL: https://doi.org/10.1016/j.aci.2018.08.003.

[22] Martino Trevisan, Alessandro Finamore, Marco Mellia, Maurizio Munafo, and Dario Rossi. «Traffic analysis with off-the-shelf hardware: Challenges and lessons learned.» In: *IEEE Communications Magazine* 55.3 (2017), pp. 163–169.

[23] Martino Trevisan, Danilo Giordano, Idilio Drago, and Ali Safari Khatouni. «Measuring HTTP/3: Adoption and Performance». In: *2021 19th Mediterranean Communication and Computer Networking Conference (MedComNet)*. 2021, pp. 1–8. DOI: 10.1109/MedComNet52149.2021.9501274.

[24] *Tstat - TCP STatistic and Analysis Tool*. http://tstat.polito.it/.

[25] Hongdong Wang, Lei Mingfeng, Chen Guanhua, Li, and Zou. «Intelligent Identification of Maceral Components of Coal Based on Image Segmentation and Classification». In: *Applied Sciences* 9 (Aug. 2019), p. 3245. DOI: 10.3390/app9163245.

[26] *WebPageTest by catchpoint:* https://www.webpagetest.org/.