Project 1: YouTube Traffic

Project and Submission rules

- Each group can consist of a maximum of 4 students.
- Each group must create a zip file containing a report describing the solution developed and all the code used to solve it.
  - Each student of the group must upload on the didactic portal (Portale della didattica) in the section Work Submission (Elaborati) **before the deadline**. Each student can perform multiple uploads **before the deadline** - In case of multiple uploads, **the last** one be considered.
  - The report must include the title and name of the students in the group. The report must be no more than 6 pages long, including textual descriptions of the solutions developed for all tasks, tables, and plots used for the project.
  - The code must be running and cover the entire project. The code must be running because it can be used during the oral part to assess how each group deployed its solution.
- The 1^{st} deadline for the project submission is 1^{st} February 2022 23:59 (CET).
- The 2^{nd} deadline for the project submission is 24^{th} February 2022 23:59 (CET).

Introduction

YouTube is one of the most popular and sophisticated Internet services. It counts 1 billion users worldwide who watch 6 billion hours of videos per month. Due to its popularity and the type of content it distributes, the load it has to handle is enormous and ensuring a satisfactory Quality of Experience (QoE) for users is a challenging task. To this end, YouTube uses a huge, globally distributed content delivery network (CDN). It consists of hundreds of *edge-nodes* scattered across the Internet **geographically around the world**. Each node hosts hundreds of video *servers*, or *caches* which can potentially deliver any video that a user requests.

Changes in the YouTube CDN occur frequently and can include changes in infrastructure, such as enabling a new cache, or in load balancing algorithm decisions, such as a sudden change in caches to serve requests. Conversely, the ISP's policies are often static and do little justice to the continuous evolution of the Google CDN: any sudden change can render the ISP's optimization obsolete and thus ineffective, leading to abrupt interruptions or QoE degradation. This is a problem for the ISP, as its reputation deteriorates when a change occurs, even if Google caused it.

The goal of this project is to develop an unsupervised machine learning method to monitor changes in the YouTube CDN and determine what is the difference between the most important features when the CDN is in a stable situation or when major changes occur. To this end, the unsupervised method must first group the YouTube *caches* into clusters that represent the *edge-nodes*. Different cluster results will be created by using data from different weeks. Second, students need to calculate the difference between the clustering results from different weeks to identify similar weeks and weeks with different clusters. Finally, students will use a supervised algorithm on the two most divergent clustering results to determine the most important features in the different situations.

For this, students will need to solve the following tasks:

- **Task 1 Data characterization**: explore the dataset and learn about the behavior of features at different levels, e.g., flow, cache, or edge-node level.
- **Task 2 Unsupervised learning**: group caches into clusters representing the edge-nodes.
- **Task 3 Difference between clustering results**: identify the clustering results that differ the most from a *baseline week*.
- **Task 4 Supervised learning**: investigate the importance of features in the *baseline week* and the week with the most different clustering result.

Dataset

Using a tool called Tstat[1], we collect a dataset that describes statistics about traffic flows with YouTube videos. Tstat observes the packets, rebuilds and tracks each TCP flow, and logs detailed statistics at the end of the flow.

For each flow, it logs i) the anonymized client IP address, ii) the server IP address, iii) the minimum TCP Round Trip Time (RTT), i.e., the time for the traffic to get from the server to the client, iv) the IP Time-To-Live (TTL) of packets received by the client, v) the number of bytes received by clients, vi) the time at which the TCP connection begins, vii) the response time as the time the server used to send the first data, viii) the average download throughput, ix) the amount of retransmitted data by the server, and x) a textual code used to identify the *edge-node*,

The data provided covers 6 weeks:

- Week 1 **baseline**: This week must be used as a baseline because the YouTube infrastructure was in a stable situation.
- Week 2-6 *changes*: These weeks are used to identify the different edge-node infrastructures and the week with the biggest differences.

Task 1: Data Characterization

The first task of the project is to present the dataset through various data visualization techniques and statistical analysis.

For this, students must

- extract features at different levels (aggregating at flows into single lines) and produce different visualization techniques and statistical analysis.
- understand which could be the most promising features.

The goal is to identify the behavior of the dataset at different levels, such as the data flow level, the cache level (aggregating all flows of same cache, i.e., s_ip into a single row), and the edge-nodes level (aggregating all flows of same cache, i.e., e_node into a single row). This step is fundamental to understand which could be the most promising features for the following tasks. As outcome of this step the students will produce in the report plots describing the dataset and considerations about the features.

[1]`http://tstat.polito.it/`

Task 2: Unsupervised Learning

The second task requires the creation of the clustering algorithm to group caches into edge-nodes.

For this, students must

- identify the best solution to **represent cache** by means of features.
- design a clustering algorithm that uses the numeric feature **without** using the textual code used to identify the edge-nodes.
  - **WARNING:** This is mandatory because the textual code can provide misleading information in some cases. The textual code can be considered in the following cluster analysis.
  - **HINT:** Since the task requires **grouping of caches** into edge-nodes that should be in different dense geographic areas, information such as RTT, TTL, can help in describing the caches and edge-nodes, but other features can also be considered.
- Week 1 is used as a baseline to identify and optimize the best clustering algorithm. Namely, this week is used to decide the clustering algorithm and the hyper-parameters of the algorithm.
  - **HINT:** Both metrics, which consider only numerical features and labels, can be used for tuning operations.
- week 2 to 6 are used to determine new clustering results. The clustering algorithm and hyper-parameters are set on the previous point and new clustering 5 different clustering results are created (one per week) to describe the different edge-node mappings.

The goal of this task is to group the caches into clusters representing the edge nodes. This step is essential to identify different edge-node configurations. As a result of this step, students will produce plots describing the clustering tuning process, the results of clustering in different weeks, and textual considerations about the entire process.

Task 3: Clustering Evolution

The third task is to study the evolution of clusters over time.

For this, students must

- given a clustering result, identify how to describe it in a numerical space.
- given two clustering results (*C1*, *C2*), each consisting of multiple clusters, calculate the similarity between the clustering results
  - **EXAMPLE:** Take each cluster in *C1* from week 1 and the cluster in *C2* from week 2.
  - **HINT:** For each cluster in *C1*, find the most similar cluster in *C2* and compute a distance metric. Then, perform the same procedure to search for each cluster in *C2* the most similar cluster in *C1* and compute a distance metric. Finally, sum all distances to get an estimate of the difference between the two clustering results

- Calculate the difference between each week and all other weeks and draw a matrix with all distances.
- Identify the week that is most different from the **baseline week**.

The goal of this task is to identify the evolution of edge-nodes. This step is important to identify changes in edge-node configurations. As a result of this step, students will produce plots describing the clustering distance, identification of the week with the greatest deviation from the baseline week, and textual considerations about the overall process.

Task 4: Interpretation and Visualization

The fourth and final task is to examine the importance of the features and how it changes depending on the week under consideration.

For this, students must

- Train and validate a Decision Tree model and a Random Forest model to find the best hyper-parameter configuration using a grid search approach.
  - The models should be trained using the data from the first week (the baseline), using as labels the clusters ID for the caches.
  - Visualize the best model performance.
- Using the best hyper-parameter configurations, train new models using data from the baseline week and the week that is most different from the **baseline week**.
- Extract the feature importance of the four models.
- Compare the feature rankings of the Decision Tree and the Random Forest of the **baseline week**.
- Compare the feature rankings of the baseline weeks with the week that is most different from the **baseline week**.

The goal of this task is to use a classification algorithm to interpret the clustering result and rank the most important features. As a result of this step, students will create plots describing the grid search trend, the best model performance using the confusion matrix, plots describing the different feature rankings, and textual reflections on the overall process.