POLITECNICO DI TORINO

Master's Degree Course in Computer Engineering

Master's Degree Thesis

Automated Data Integration and Machine Learning for Enhanced Social Media Marketing Analysis

Supervisor
prof. Daniele Apiletti

Candidate
Matteo Borzi

Company supervisor
dott. Donato Chiarello

October 2023

Abstract

In today's digital era, the abundance of data generated on social media platforms presents a valuable resource for extracting marketing insights. This thesis, born from an internship project at Mediamente Consulting s.r.l., addresses the pressing need of a customer for efficient data integration and automation in the context of social media marketing campaign analysis. The primary objective is to develop a robust Data Integration model to streamline the visualization of social media marketing campaign performance.

The traditional approach of manually downloading and aggregating standard reports from YouTube and LinkedIn analytics tools is time-consuming and error-prone. To replicate the company workflow, our thesis outlines a comprehensive framework comprised of four key stages.

The initial stage, Data Ingestion, involves the extraction of data from various sources with the utilization of REST APIs provided by the respective social media platforms. In cases where APIs are not available, a web scraper is employed, ensuring a comprehensive data collection process. Following the Extract, Transform, Load (ETL) design pattern, the data is prepared for analysis. This phase involves the integration of data from diverse sources into a unified model, ensuring consistency and coherence for subsequent analyses. Leveraging supervised and unsupervised Machine Learning techniques, data points are clustered into meaningful groups. Each record is then assigned a label based on the corresponding marketing campaign, enriching data and facilitating richer insights into campaign performance. The reconciled and enriched data is stored in a Google BigQuery dataset, serving as a centralized Data Warehouse. This repository allows for complex querying and facilitates both Data Visualization and further Machine Learning analyses.

The thesis acknowledges the potential for future enhancements to the developed system, including the incorporation of additional data sources such as Google Ads for comprehensive cost and revenue analysis. Moreover, consideration is given to the prospect of a full migration to a serverless cloud platform solution to enhance scalability and reliability, ensuring the system's long-term viability.

In summary, this thesis presents an innovative approach to addressing the automation of data integration challenges inherent in social media marketing campaign analysis. By optimizing the collection, preparation, and enrichment of data from wide-ranging sources and with the advantage of Machine Learning techniques, it offers a powerful decision-support resource for the customer who is seeking deeper insights and greater efficiency in their campaigns.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Social media

Social media are dynamic Web applications where users can interact with others through personal profiles and user-generated content, thus developing social networks. These platforms have gained popularity with the wide spread of smartphone technologies and they now represent a central tool for both entertainment and working purposes, especially in marketing, for which data collection and analysis are crucial.

1.1.1 A brief history of social media platforms

In the late 1990s the *World Wide Web* was mainly made of static Web pages in which content generation was meant to be performed beforehand in order to offer the same information to all the visitors. The first social Web platforms like *classmates.com* (launched in 1995) and *SixDegrees* (born in 1997) allowed users to get in touch with each other through personal Web pages or email addresses. The concept of "friendship", to be intended as a connection between two users via their profiles, was the core feature and the main focus of these services.

A major shift in technologies and development frameworks was brought by the introduction of new techniques such as asynchronous programming[1] and standardized Web protocols and architectures in the early 2000s: this change of paradigm is called *Web 2.0.* Such evolution enabled websites content to be dynamically generated by users, leading to the birth of several social media and networking platforms. The main difference between the two resides in the purpose of the platforms. Social

[1]The major advancement with Web 2.0 paradigm can be ascribed to user's continuous interaction with the Web page on the client while the server processes the requests at its own pace. AJAX (*Asynchronous JavaScript and XML*) is one of the most prominent examples of asynchronous programming techniques.

media encompass a wide range of digital tools and technologies that facilitate the creation of content and interaction among users in relation to the produced content. In contrast, social networks represent specific online platforms where the aim of the individuals is interconnectivity itself.

Platforms such as *Friendster* and *MySpace* laid the foundation for social media, after which the advent of *Facebook* in 2004 brought the phenomenon to a wider public. Following the diffusion of new technologies and approaches, further innovations such as microblogging with the advent of *Twitter* in 2006 and visual content sharing through *Instagram* and *Snapchat* have been possible. Within this environment, YouTube has redefined content sharing through its video-centric approach. Meanwhile, LinkedIn, another pivotal social media platform, has revolutionized professional networking and engagement.

Inaugurated in 2005, YouTube serves as an example of how content creation has become accessible to a wide public. Being the world's largest video-sharing platform, it has given users the ability to create their content, from educational tutorials to entertainment.

LinkedIn, established in 2002, occupies a niche in the realm of social media by focusing on professional networking. LinkedIn is designed to support career advancement, job search, and business connections by letting users create digital résumés and cultivate relationships with colleagues and professionals.

1.1.2 Social media marketing

The rise of social media platforms has led to the birth of social media marketing, a discipline aimed at exploiting these platforms for advertising purposes. Social media marketing entails the creation of content designed to achieve many marketing goals, including *brand awareness*, *user engagement*, and *lead generation*. Its growing importance is linked to the vast number of users inhabiting these online environments. Businesses have come to understand that establishing a presence on social media is not just advantageous but, in fact, necessary for effectively connecting with and engaging their target audience.

1.2 Case study

Mediamente Consulting s.r.l. is a Business Analytics and Big Data company born as an innovative start-up in 2012 at Politecnico di Torino's I3P incubator, then acquired by Var Group (SeSa S.p.A.) under their Data Science unit. It specializes in Technological Infrastructure, Data Integration and Management, Data Visualization, Business Intelligence, and Corporate Performance Management (CPM), which are the key Business Units for a comprehensive facilitation of a shift towards a data-driven approach for customer companies.

This work is carried out with the collaboration of the Data Integration business unit, which is mainly responsible for migrating existing on-premise *On-Line Analytical Process (OLAP)* systems to cloud-based solutions and designing data pipelines, including data ingestion, transformation, and storage into centralized Data Warehouses and Data Marts to empower Business Intelligence operations.

The practical aim of this thesis originates in the request of a customer who actively uses their social media platforms, such as YouTube and LinkedIn, to generate engagement and involve new audiences in their products and services by publishing platform-specific content. They manually download tabular data about user interaction provided by the standard reports of the respective platforms on a daily basis and collect them in a spreadsheet which supplies the daily metrics to a Business Intelligence (BI) dashboard for *Key Performance Indicators (KPIs)* visualization. The current workflow has many downsides:

- **Handmade data ingestion is a mechanical activity** and can be entirely performed and managed by a machine
- **Human operations are frequently subject to errors**. Updating and transcribing many records by hand can be a source of data redundancy or inconsistency
- **Multidimensional Analysis requires complex data structures** to frame a model that can coherently describe the underlying data patterns and enable the extraction of meaningful information.

Hence, the project's main business focus lies in the entire automation of the data ingestion process following the *ETL (Extract, Transform, Load)* pattern and the integration of the manifold sources into a centralized Data Warehouse.

The academic purpose of the work revolves around the Data Enrichment phase during ETL. Feeding BI dashboards and Data Visualization tools with reconciled and integrated data allows significant information for decision processes to be gathered and represented. Nonetheless, we can leverage the most recent advancements in Machine Learning to implement a solution that yields additional insightful knowledge that is inaccessible otherwise. *Unsupervised Learning* is a compelling tool for a preliminary analysis of the marketing campaigns' effectiveness, e.g., unsupervised techniques like *clustering* provide a partition of the available data points into closely related subsets. *Supervised Learning* techniques, such as *binary* and *multi-class classification*, facilitate both a qualitative and quantitative representation of performance through the assignment of a label to each data point.

1.3 Chapters description

A list of the following chapters with a brief outline is presented.

- Chapter 2 portrays the current state of the art in Data Integration patterns and technologies, with a focus on Data Warehouse models, and Machine Learning techniques.
- Chapter 3 delves into the Data Integration framework developed at Mediamente Consulting. Three layers are defined within the ETL process: the Staging Area, where data is replicated from the sources; the Relational Data Storage, which serves as a central reconciled data repository; the Dimensional Data Storage, where the actual integration operations are performed and data is prepared to be loaded into the Data Warehouse. An extra layer is added at the foundation of the framework in order to manage metadata and the automation process.
- Chapter 4 describes the case study and proposes a Data Warehouse model for the purpose analysis following the best practices in in preparation for the implementation of a customized version of the company framework. Here the employed tools and resources are listed.
- Chapter 5 presents an extensive description of the ingestion and data enrichment steps of the designed ETL process. Two data sources are analyzed: YouTube APIs and LinkedIn reports, which are provided by a web navigation automation mechanism such as Selenium. The integration step is illustrated in all its phases and the Load step on Google BigQuery is described.
- Chapter 6 sets up two Unsupervised Learning models in order to extract underlying patterns from YouTube data. The main purpose is defining clusters based on interactions with the YouTube platform in order to assign class labels according to clusters. These labels will be employed for Supervised Learning tasks on new records in the data enrichment phase.
- Chapter 7 present the conclusion of the thesis and treats possible future implementations to enhance the work.
- Appendix A provides a comprehensive list of the designed tables for the Data Warehouse model.

Chapter 2

State of the art

Business Intelligence and Advanced Analytics play a significant role in the Big Data scenario and deliver functional tools to gather information for core marketing decisions. On the other hand, a relevant amount of knowledge remains undiscovered with the traditional BI approach: Machine Learning algorithms are considerable resources for data enrichment and outcome forecasting. In this chapter, we delve into the current Data Ingestion and Integration patterns, Data Warehousing models, and the bedrock of the chosen predictive analysis techniques.

2.1 Data Analytics and Big Data

The field of *Big Data* deals with large datasets originating from many sources, which can be either structured like in transactional records coming from Relational Database Management Systems (RDBMSs), Internet-of-Things (IoT) log files, sensors measurements, and industrial automated monitoring systems or unstructured data and documents from non-relational databases containing users' interactions with websites and mobile devices or content from social media platforms (e.g. textual posts, pictures, videos, audio recordings).

Doug Laney defined the three fundamental concepts of Big Data in 2001 known as *«the Three V's»* (*Volume*, *Velocity*, *Variety*) [1], which later served as a basis for organizations to appoint several additional qualities that fully describe Big Data [2].

- **Volume.** The first characterizing element of Big Data resides in the data size, which holds an inherent value for statistical analysis: as of April 2023, roughly 328.77 Exabytes of data are generated every day, with videos and social media interactions being responsible for approximately 66% of the data traffic [3]. An average-sized dataset can reach the magnitude of terabytes, even petabytes, and tends to be stored in large distributed data clusters rather than centralized servers, moving in the direction of *scalable* cloud-based solutions.

- **Velocity.** Data is produced continuously at an increasing pace: in 2022, the average content upload rate was nearly 500 hours per minute, while around 231 million emails were sent each 60 seconds [4]. For this reason, data processing is shifting towards a real-time or near real-time approach to supply the constant demand for information in crucial processes and urgent decision contexts like anomaly detection.

- **Variety.** Gathering highly diverse data sources implies the need to reconcile and combine strictly structured data, organized in records with a rigid and coherent schema, with unstructured data such as images, videos, text files, emails, and non-relational database objects. Albeit RDBMSs perform thorough management of structured data, they are not suitable for handling semi-structured and unstructured entities. Data Lakes and Data Warehouses are more appropriate models for storing all the different kinds of data.

- **Veracity.** Data quality is essential in a BI scenario, where the business future relies on the correctness and dependability of processed information [5]. After collection and integration, data must be coherent and meaningful, such that the decision process is backed by a trustworthy source in which data is reliable, does not present inconsistencies or contradictions with its meaning, and it is robust to noise.

- **Value.** The derivation of insights from data offers decisive advantages for business and research fields. Furthermore, data possesses the capacity for versatile utilization across different contexts, transferring its significance from the data itself to its inherent potential for reuse [6]. This concept also holds in Machine Learning and other predictive analysis scenarios, where comprehensive modeling, which often requires substantially large datasets, is not always feasible for quality data scarcity.

The Big Data paradigm firmly reflects in the data pipeline architectures: centralized RDBMSs are replaced by Data Lakes, Data Marts, and distributed Enterprise Data Warehouses. Plenty of sources are concerned in the Data Ingestion (e.g. records from Enterprise Resource Planning systems or Customer Relations Management systems, social media REST API responses for user interaction requests, log files from IoT devices, real-time web crawling results), while the Extract, Transform, and Load (**ETL**) phase is frequently reinforced with the implementation of *MapReduce* and *Hadoop* frameworks.

2.1.1 Data integration design patterns

Data pipelines prepare the data obtained from various sources to be placed in definite data storage architectures, enabling eventual reporting, visualization, and further processing. The increasing importance of Data Integration in the research field has brought its role from a marginal task in data warehousing to a crucial element of Data Warehouse projects, changing the standard approaches in their development. Currently, Data pipeline design patterns can be grouped depending on the stages that compose the pipeline and their execution order.

Extract, Transform, Load

ETL is the Data Warehousing process that performs the Extraction, Transform, and Load stages in order. After the Extraction phase, the Transform step is performed on data in the same place as the staging area. The advantages of traditional ETL principally concern the realization workflow overhead:

- A backward approach is usually adopted when implementing an ETL pipeline by designing the output first, thereby speeding up the realization process [7].
- Limiting implementation to the needed resources and business rules enables time and complexity savings [7].
- Data collected in the Data Warehouse has already been transformed and does not require further processing before its visualization [8].

On the other hand, the ETL pattern has many downsides:

- The data transformation step is the most intensive section in terms of computational resources and it can be a potential bottleneck when performing row-wise data quality checks [7].
- ETL is a monolithic process and requires a re-design whenever the requirements change and involve new data or transformations, thus leading to increasing costs and development time [9].
- Data transport is suboptimal since it is performed twice - first from the data sources to the ETL server and then from the ETL server to the target storage. Furthermore, data processing with an ETL approach requires a dedicated server, resulting in higher hardware costs [8].

Extract, Load, Transform

The **ELT** approach derives from ETL by inverting the Transform and Load steps: records and documents are gathered from the data sources and directly loaded into the target system, where they undergo validation and enrichment. Among the ELT pipeline features we can find:

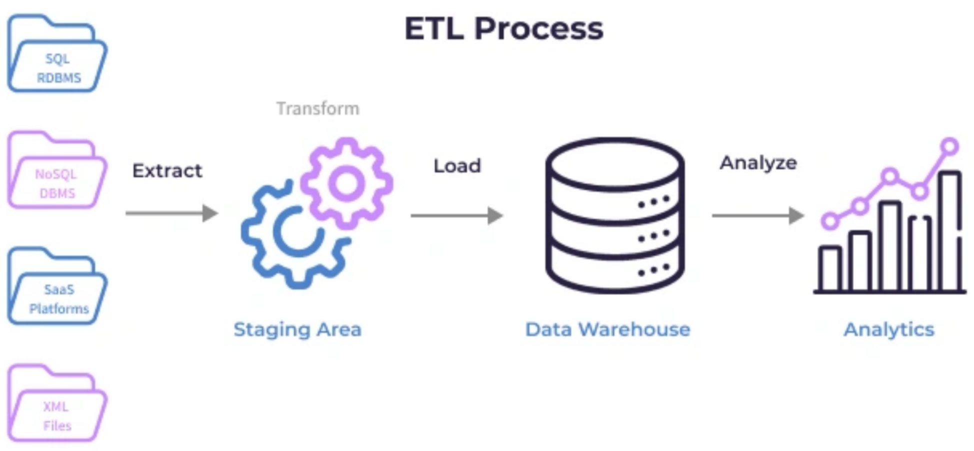


Figure 2.1: Basic ETL process in Data Warehouses. Data is ingested from many sources into the staging area, then cleansing, integration and enrichment are performed. Finally data is loaded in a Data Warehouse for subsequent uses such as BI analysis, reports or Machine Learning. © D. Tobin, 2022

- **Improved network management**. Data is transferred in a single step before transformations and business rules are applied, reducing the overall network traffic [10].
- **Performance and scalability**. ELT architectures do not require external technologies, servers, or tools, and process each step with a single query, thus leading to improved performance and easier scalability management [10].
- **Flexible implementation.** In ELT implementations, the Data Warehouse contains all the untreated data as a result of the Extract and Load stages, hence enhancing the implementation of future requirements [7].

However, ELT is not recommended for lower volumes of data and more complex data transformation operations are better performed by ETL mappings.

Extract, Load

In the Extract, Load (EL) pattern data is first pulled from the source, then stored as is, without any further transformation. Although EL is fairly uncommon in data warehousing, Data Lake architectures exploit this kind of pattern for serving as raw data repositories [11].

Other patterns

Under some circumstances, traditional architectures are not viable, and application-specific data paths are developed. For instance, *H. M. N. Dilum Bandara et al.* suggest an inversion of the Extract and Transform steps in the ETL process to overcome the incompatibility between source and target endpoints in a blockchain migration scenario [12]. *Umeshwar Dayal et al.* provide another example by proposing an ETLT (Extract, Transform, Load, Transform) pipeline in which the transformation phase is divided in two: the former is executed after the extraction, while the latter exploits Data Warehouse servers' scalability and parallelization capabilities [13].

2.1.2 Data sources and ingestion

The first phase in data processing for visualization and analysis is Data Ingestion. Unprocessed Data is acquired from a large variety of sources and collected in the *staging area*, which serves as the primary source for the subsequent steps in the Data Integration stage.

In business scenarios, Enterprise Resource Planning (ERP) systems provide traditional operational databases with structured data organized in tables and records like sales invoices, inventory snapshots, customer information, medical records, and bank transactions.

When dealing with websites, more specifically with social media platforms, the principal data source is a web server database, which is queried through the usage of *Representation State Transfer Application Programming Interfaces* (REST or RESTful APIs). The web server supplies functions that web clients invoke in the form of HTTP requests, which typically provide a response in standard formats such as *JavaScript Object Notation* (JSON) or *Extensible Markup Language* (XML). Client applications elaborate API responses for user visualization, interaction, or further processing.

While APIs are functional instruments for web data collection, several websites do not offer accessible means for structured data collection, thus requiring external methods. Web scraping is a data science technique that responds to this need by deploying scripts for the extraction of structured data from websites [14]. Scrapers combine an in-depth web page exploration (*crawling*), which involves capturing and recursively vising the web pages in a list of websites, to data research in each gathered resource, followed by a preliminary data preparation before the actual ingestion in the pipeline.

There exist two ingestion methodologies, with distinct use cases and procedures: static and incremental ingestion.

Static ingestion

Static ingestion (*"FULL mode"*) consists in gathering all the available records in a snapshot from a source and transferring it to the succeeding steps of the ETL process. Static ingestion is performed during the first iteration of the pipeline (*Initial Load*) and it is generally slower than incremental ingestion, since it needs to read massive amounts of records periodically. Therefore, it is suited for smaller tables with few records, such as dimensional tables, or when incremental ingestion is not available.

Incremental ingestion

Incremental ingestion is performed through the *Change Data Capture* (CDC) mechanism: only new, updated or newly deleted records are acquired through each iteration, reducing the number of records that need to be processed, speeding up the capture phase and reducing the quantity of data that needs to be stored and updated when being subject to changes.

Multiple sources directly provide CDC mechanisms at the top of the ingestion phase, while others allow an incremental collection of records through source-dependent timestamp metadata. If CDC operations are not feasible, a similar solution can be reached by capturing the difference between the current snapshot and the previous one (*delta* operation).

2.1.3 Batch and streaming ETL architectures

Data pipeline architectures can be classified as *batch* or *streaming* by how data flows through during each step.

A *batch* pipeline processes data in groups or *chunks*, which are periodically collected and elaborated at scheduled intervals. Each batch flows through the pipeline independently from others. When dealing with Big Data, batch processing is the most suitable solution for non-critical periodically scheduled analytics.

Streaming pipelines deal with real-time data or near real-time data, enabling streams with a low latency: records are picked from the source continuously, then they are aggregated in a *window*, depending on the extraction logic and sent through the subsequent steps of the pipeline as soon as the windows are marked as ready. There are three principal types of windowing aggregations:

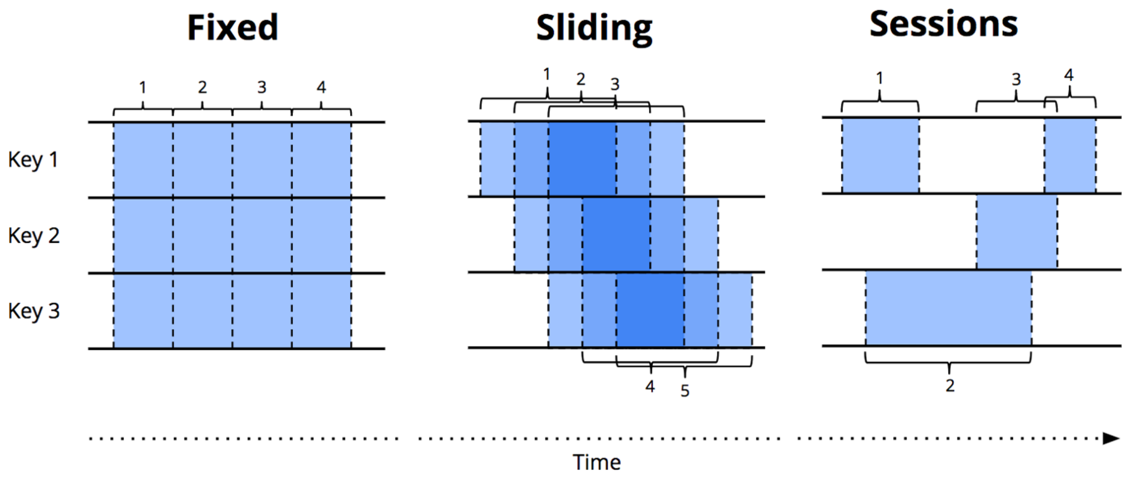
- **Fixed time windows**, or **tumbling windows**, are disjoint windows with a fixed, constant duration T. Given a timestamp t_i, data is assigned to its corresponding window W_k represented by the time interval that encompasses its value:

$$t_i \in [kT, (k+1)T) \Rightarrow t_i \in W_k$$

- **Sliding time windows** or **hopping windows** are overlapping windows with constant duration T and a period n, which represents the frequency at which each window W_k starts capturing data:

$$W_k = [t, t + T)\,, \qquad W_{k+1} = [t + n, t + n + T)$$

- **Session windows** define a time *gap duration* between data after which records are assigned to the next window. In session windows *data keys* are relevant, since each session window is key-dependent, namely session with different keys can overlap.



(a) Fixed time windows **(b)** Sliding time windows **(c)** Sessions time windows

Figure 2.2: Types of windowing aggregations

Due to their nature, streaming pipelines need a *watermark* to deal with late or missing data. Moreover, data is not guaranteed to arrive in order. A watermark is a threshold that represents the moment in which all the data belonging to a window should have been collected [15]. Each data point is attached with a timestamp and sent through the pipeline: if the timestamp exceeds the watermark, the relative record is considered as late and must be handled with a specific policy.

Unlike batch pipelines, where data is sent to the next stage when the entire batch has been processed, a data stream can adopt more diverse pipelining strategies. *Triggers* define the rules that allow data to be forwarded to the next steps. For instance, in Apache Beam [15] conditions can depend on the timestamp value, the processing time or the number of elements collected at a given step in a window.

2.1.4 Data integration and enrichment

The Transformation step is the central phase of the data reconciliation process in which raw data is given more value by combining, enhancing, and reformatting it to meet specific business needs. In ETL architectures, this is the step in which raw data is refined and prepared for storage in a Data Lake, a Data Mart, or a Data Warehouse; in ELT, data is transformed after being stored in a Data Hub or another model inside the Data Warehouse.

Data Quality

Data quality is a capital step in the data enrichment process, since it lays the basic bricks for reliable and valuable insights. Ensuring that the data being enriched is accurate, complete, and consistent is crucial to avoid making decisions based on flawed or incorrect information. Data quality checks encompass a range of activities, such as data validation to identify missing or inaccurate values, data cleansing to revise inconsistencies and formatting issues, and data standardization to give data a uniform structure.

For instance, when integrating customer records, data quality checks might involve verifying that all the addresses are in a standardized format, correcting misspellings in names or other textual fields, and ensuring that birth dates are within coherent ranges. Guaranteeing accurate and consistent data is critical for obtaining trustworthy data assets and maximizing the utility of enriched data for further analytics, reporting, and decision-making processes.

Data integration

Data integration involves adapting the diverse data into a single, coherent schema or model. This process encompasses several techniques, including data joins, unions, and merges, depending on the nature of the data. For example, in a retail context, data integration might involve combining sales transaction data from point-of-sale systems with customer data from CRM platforms. By performing data joins based on unique identifiers, the resulting model can bring in-depth insights into customer purchasing behavior, product preferences, and revenue trends. Effective data integration not only provides a comprehensive view of business operations but also enables cross-functional analysis, aiding decision-makers in optimizing strategies, improving customer experiences, and identifying new business opportunities.

In addition, data enrichment commonly includes a data augmentation process: further sources than the ones provided by the customer can be included in the integration step to supply contextual information and details to enrich data in the model. For instance, finer details of geographic locations can be joined to geolocation coordinates in order to enhance spatial analysis of job offering applicants in a LinkedIn campaign.

Machine Learning and data enrichment

Data enrichment can be carried out leveraging Machine Learning techniques as well. In this context, Machine Learning plays a pivotal role by automating the process of feature extraction, prediction, and trend analysis. Clustering algorithms, such as K-means or hierarchical clustering, can be employed to group similar data points together, thereby revealing hidden patterns within the data. Furthermore, classification and regression models offer the capability to predict missing values or forecast future trends based on historical data. For instance, in business analytics, Machine Learning models can predict the trend of specific point-of-sales or foresee a product shortage. Similarly, in e-commerce, recommendation systems can enhance the customer experience by suggesting relevant products based on their past purchasing behavior.

2.1.5 Data Lakes, Data Marts, Data Warehouses

There are multiple storage technologies for Advanced Analytics which play a significant role in the Big Data scenarios. In particular, Data Hub and Data Lake architectures are employed to store both structured and unstructured data in a common repository for further processing, while Data Marts and Data Warehouses serve as a centralized and complete historical data storage for multidimensional analysis. Hybrid solutions like Data Lakehouses are available.

Data Lakes

The aim of Data Lakes is the storing large amounts of data with their native format in a flat architecture in order to function as a primary source for all the analytics, reporting and visualization purposes in a company. The data stored in a Data Lake does not require to be directly valuable to the organization, since its meaning may be unrecognized at the time of design. However, Data Lakes reduce the cost of data ingestion and provide an exhaustive view on company data with a flexible and scalable solution [16].

Data Lakes exist in four stages:

1. **Data Puddle.** A moderate sized collection of unstructured and unorganized data is loaded with an EL pattern in a Data Puddle, which is generally owned and managed by a single team.
2. **Data Pond.** Multiple Data Puddles are gathered in Data Ponds, where data presents a mild data types organization and a raw structure with little analysis and conditioning [17].
3. **Data Lake.** The proper Data Lake is stored in a non-relational DBMS (e.g. *NoSQL* technology) or a distributed file system such as HDFS (*Hadoop Distributed File System*) with an extensive supporting documentation through

metadata. Business users can access via both classical query languages (SQL) and data reporting and visualization tools (Microsoft PowerBI, Tableau) [18].

4. **Data Ocean.** An expanding Data Lake that reaches a remarkable size is called Data Ocean. While including a wide range of sources and offering a broad availability, Data Oceans are subject to the risk of degenerating into *Data Swamps*, which are unorganized collections with usually obsolete data due to the difficult elaboration.

Data Marts

In a Data Warehouse scenario, a *Data Mart* is a model that encompasses a single aspect of the entire business architecture. For instance, a supermarket Enterprise Data Warehouse may be formed by smaller Data Marts for each domain of competence, such as the logistic area or inventory snapshots.

Data Warehouses

The *Data Warehouse* (DWH) is the main data management system for storing the reconciled data for supporting downstream analytics, reporting, data visualization and other Business Intelligence tasks. Data Warehouses house historical data that has been processed by ETL/ELT pipelines and certified through data quality checks in order to furnish a reliable support for the *knowledge workers* [19] and decision-making processes. The Data Warehouse design can follow a *top-down* or *bottom-up* strategy:

- **Top-down.** The initial phase of a top-down design process involves planning the Data Warehouse model with the objective of encompassing all business data, thereby offering a holistic view on company data. Subsequently, the design includes the creation of business area-specific Data Marts. This method complies with the backward methodology for ETL and enables the representation of diverse perspectives at a later stage.

- **Bottom-up.** The bottom-up approach consists in planning many Data Marts to be aggregated in the main DWH. The bottom-up strategy enables a quick access to specific business area information.

2.1.6 OLTP and OLAP systems

Data Warehouses are a widespread solution for On-Line Analytical Processing (OLAP), which empowers the analysis of large amounts of data from a large number of perspectives for decision-making purposes. On the other hand, On-Line Transactional Processing (OLTP) is the standard operational databases pattern, which supports frequent data alteration with the optimization of multi-access write operations. Table 2.1 provides an overview on the main differences between OLTP and OLAP systems [20].

	OLTP	**OLAP**
Purpose	Transactions and operational support	Support in decision-making process
Query pattern	Simple, repetitive	Complex, ad-hoc
Query operations	Insert, Update, Delete, Select (read-write)	Select (read-only)
Amount of records	Low (hundreds of records per query)	High (millions of records per query)
Time window	Limited to current data	Historic (coverage of years)
Refresh rate	Immediate, per transaction	Scheduled, mostly daily

Table 2.1: Main differences between OLTP and OLAP systems

2.1.7 Cloud Data Warehouse solutions

Data Warehouses have benefited from the advent of cloud technologies, since the availability of massive computational and storage resources has empowered their availability, scalability, and reliability, with a variety of cost-effective solutions. Unlike on-premise architectures, which require the management of the whole infrastructure, cloud platforms provide three possible services depending on the level of flexibility and responsibility:

- **Infrastructure-as-a-Service (IaaS)** is the cloud model that provides access to a set of on-demand storage and computational resources on a remote server, with basic low-level services such as virtualization and networking tools. IaaS is the least expensive architecture, due to the fact that the customer is responsible for the management of the remaining components (e.g. Operating System, applications, middleware) [21].

- **Platform-as-a-Service (PaaS)** model offers an higher-level solution than IaaS, supplying the operating system and further software resources without the need of building and maintaining the whole infrastructure. The customer's responsibilities concern the development, the deployment, and the maintenance of software applications [22].

- **Software-as-a-Service (SaaS)** solutions furnish on-demand access to an application. The entire underlying stack is managed by the service provider, entrusting the mere usage of the software to the customer, generally through a subscription to a web application [23].

Plenty of Data Warehouse and ETL cloud services that leverage the three cloud architectural patterns are available. Here are some examples of the major ETL cloud solutions:

- Oracle Data Integrator
- Microsoft Azure Data Factory
- Google Cloud Dataflow
- Pentaho Data Integration
- Apache Spark
- Apache Kafka

The first three are proprietary SaaS solutions and require a subscription, while Pentaho Data Integration Community Edition, Apache Spark and Apache Kafka are open-source projects and require a PaaS or a IaaS architecture. Hybrid solutions for processing and storage are also widely used in order to meet technological and economic constraints of the projects.

2.2 Machine Learning foundations

In 1997, Tom M. Mitchell presented a functional definition for Machine Learning:

> *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E"* [24].

This definition underscores the idea that machines can autonomously extract patterns, make predictions, and adapt their behavior based on past data. Machine Learning encompasses diverse techniques and methods, with two fundamental paradigms being *Supervised Learning* and *Unsupervised Learning*, which serve distinct purposes and present unique characteristics.

2.2.1 Supervised Learning

Supervised learning involves training a model using labeled data. In this setup, the algorithm generates a function that maps inputs to the desired output [25], enabling it to make predictions or classifications when presented with unseen data. This form of Machine Learning is based on the presence of output values called *labels*, which instruct the model to recognize patterns and *infer* the label of unlabeled new data points, particularly suited for tasks such as image recognition, spam email detection, and sentiment analysis, where the algorithm can learn from historical examples and their associated correct outcomes. Supervised Learning algorithms can be either classified by the output domain in *classification* and *regression* tasks.

The *training* of a Supervised Learning model is the process of determining the mapping, also called *decision function*, from the m-dimensional *feature space* to a label space in order to generalize the problem given a set of labeled samples.

The type of decision function determines another categorization of the models in *discriminant models*, *discriminative models*, and *generative models*.

- **Discriminant models** shape the decision function by directly mapping the feature vector to its label [26]
- **Discriminative models** construct a function that maps the input to a set of scores and subsequently use that set to assign new samples to one of the classes [26].
  - When the score has a probabilistic interpretation, it is the class-posterior probability $P(\mathcal{C}_k \mid x)$ which represents the probability of belonging to the class $\mathcal{C}_k$ for the sample x. *Logistic Regression* classifiers are an example of discriminative probabilistic classifiers.
  - In a non-probabilistic model, like *Support Vector Machines* (SVMs), the score is employed to assign labels without having a probabilistic interpretation. In the case of SVMs, the score is related to the distance of the data points from the decision surface.
- **Generative models** determine both the joint distribution of the feature vector and each class label $P(x, \mathcal{C}_k) = P(x \mid \mathcal{C}_k)P(\mathcal{C}_k)$, then extract the class-posterior probability by applying the Bayes theorem.

Classification

Classification is the Machine Learning task of finding a mapping from of the input values, represented by vectors, to the space of labels, which is a discrete set that constitutes the output values, in order to predict the label of unseen input vectors.

Machine Learning models enable multiple classification tasks:

- **Binary classification.** The output space is represented by a set of cardinality 2 $L = \{\mathcal{H}_T, \mathcal{H}_F\}$. Binary classifiers model a set of problems where, given an input vector, its label represents a true hypothesis ($\mathcal{H}_T$) or a false one ($\mathcal{H}_F$), such as in intrusion detection or identity verification scenarios.

- **Multi-class classification.** The output set of a multi-class classifier is the set $L = \{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_N\}$ with cardinality $N > 2$. Multi-class classification is employed in assignments like object categorization, image recognition, and speech decoding.

- **Closed-set classification.** The aim of a closed-set classifier is recognizing a set of categories that does not change from training to testing [27], meaning that all the unseen data points will fit one of the available classes. This holds true for scenarios in which the model is accurate enough to be trained on a label space that covers all the possible values.

- **Open-set classification.** The open-set problem assumes that the knowledge of the world is incomplete at training time and that unknown classes can be submitted to the model during inference [28]. An open-set classifier is therefore required to provide an additional label for the data points that do not fit any of the other classes.

Among the classification models, generative examples are Naïve Bayes classifiers, Variational Autoencoders (VAEs) [29], Generative Adversarial Networks (GANs) [30], and Generative Pre-trained Transformers (GPT models) [31].

Support Vector Machines [32], Logistic Regression classifiers [26], and traditional Neural Networks such as Feed-Forward Neural Networks [33] are examples of discriminative classifiers.

Regression

Regression tasks require the output domain to be defined as continuous, since each data vector is assigned to a real value. Some examples are *linear* and *polynomial regression* models [26], *Autoregressive Integrated Moving Average* (ARIMA) models [34], Support Vector Regression (SVR) [35], Long Short-Term Memory Recurrent Neural Networks (LSTM RNNs) [36]. Their applications include modeling continuous-valued functions such as in stock market and other time-series forecasting settings, and physical phenomena modeling.

2.2.2 Unsupervised Learning

Unsupervised Learning operates in scenarios where the data lacks explicit labels or categories: the machine receives feature vectors as inputs without a label as output, with the aim of identifying inherent structures, patterns, or relationships within the data itself [37]. Clustering and dimensionality reduction are common techniques within Unsupervised Learning.

Clustering

Clustering aims at grouping data points together based on the similarity of their properties. Its purpose is aiding in the discovery of natural groupings or hidden insights within the data. Clustering algorithms are subdivided in various categories depending on their characteristics:

- **Exclusiveness.** Determines whether a data object can belong to more than one cluster.
  - In *Exclusive Clustering*, or *Hard Clustering*, each data point is assigned exactly to a cluster, resulting in non-overlapping partitions of the starting dataset.
  - *Inclusive Clustering*, also called *Soft* or *Fuzzy*, models the affinity of every element into each cluster. In particular, the degree of membership to each cluster is assigned to each item.
- **Nesting.** It is the clustering algorithms property that describes the possibility of defining nested clusters.
  - *Partitional Clustering* algorithms extract disjoint partitions of the dataset, prohibiting nested clusters.
  - *Hierarchical Clustering* group data elements into trees of clusters, either in an agglomerative (bottom-up) or a divisive (top-down) fashion [38].
- **Completeness.**
  - A *Complete Clustering* algorithm assigns each data point to at least a cluster, covering the whole starting dataset.
  - *Partial Clustering* permits to leave data points unassigned, resulting in a spare partition without common properties.

The main two clustering algorithms are presented for each of the two existing cluster assignment techniques: K-Means algorithm for *distance-based* clustering and DBSCAN for *density-based* clustering.

K-Means clustering

The K-Means clustering algorithm is a partitional hard clustering technique that revolves around the concept of *centroid* (or *center*) of a cluster, which is defined as the mean of the distances among all the cluster points.

The algorithm implements an Expectation-Maximization technique in order to minimize the objective function defined through the *sum of the squared error* measure:

$$J = \arg\min_{C} \sum_{i=1}^{K} \sum_{x \in \mathcal{C}_i} \|\mathbf{x} - \mu_{\mathbf{i}}\|^2, \qquad \mu_i = \frac{1}{|C_i|} \sum_{x \in \mathcal{C}_i} \mathbf{x}$$

where μ is the cluster C_i centroid, i.e. the center of the cluster. The algorithm requires the number of clusters K to be provided a priori as a parameter. The Algorithm 1 illustrates the basic K-Means procedure [39].

Algorithm 1 K-Means clustering algorithm

```
Elect $K$ points as initial centroids.
repeat
    assign each point to the cluster with the closest centroid.
    recompute the centroid of each cluster.
until Centroids do not change
```

Time Complexity: $\mathcal{O}(n \cdot K \cdot I \cdot d)$

Centroids are calculated on a distance measure, which determines the degree of closeness of the points. *Euclidean distance* is the most common measure, however other measure functions can be employed: *Cosine distance*, *correlation*, *Manhattan distance* [40]. Centroid initialization is generally performed by electing the points randomly, thereby producing different clusters for each run.

K-Means complexity is $\mathcal{O}(n \cdot K \cdot I \cdot d)$, where:

- n is the cardinality of the dataset
- K is the number of clusters
- I is the number of iterations
- d is the dimensionality of the feature space

The algorithm usually converges in the first few iterations, followed by an adjustment stage. By relaxing the stopping condition to the variation of the assignment for a limited number of data points, the effective complexity is reduced.

DBSCAN clustering

The *Density-Based Spatial Clustering of Applications with Noise (DBSCAN)* is a partitional clustering algorithm. Its gist is forming one or more clusters in which the points *density* is sufficiently higher with respect to the density of other regions. More specifically, the algorithm defines two parameters: the minimum number of points *minPts* required to form a cluster and the distance ϵ that serves as a proximity upper threshold for the cluster creation [41].

DBSCAN algorithm subdivides the data points into three groups: *core points*, *border points*, *noise points*.

- **Core points** are the points that constitute the interior of a cluster. A point is assigned to the core points if it has at least *minPts* neighbor points within the distance ϵ.

- **Border points** are neighbor points of a core point that do not meet the *minPts* constraint within the distance ϵ.
- **Noise points** are the remaining points, which do not belong to either of the other two groups.

The algorithm selects an unlabeled point and expands its neighbor points in the radius ϵ in order to define the border points and the noise points. If there are at least *minPts* within ϵ, a new cluster is defined and expandend subsequently, until all the points have been reached. The initial point is generally decided randomly. The complexity of DBSCAN is $\mathcal{O}(n^2)$ with n being the dataset size, while it can be lowered to $\mathcal{O}(n \log n)$ by employing a spatial index when looking for the neighbor points [38].

Algorithm 2 DBSCAN clustering algorithm

```
$C \leftarrow 0$
for each unvisited point $P \in$ dataset do
    mark $P$ as visited
    $Neighbors \leftarrow$ neighbor points of $P$ in range $\epsilon$
    if number of $Neighbors < minPts$ then
        mark $P$ as noise
    else
        $C \leftarrow C + 1$
        add $P$ to cluster $C$
        for each point $P' \in Neighbors$ do
            if $P'$ is not visited then
                mark $P'$ as visited
                $Neighbors' \leftarrow$ neighbor points of $P'$ in range $\epsilon$
                if number of $Neighbors' > minPts$ then
                    $Neighbors \leftarrow Neighbors \cup Neighbors'$
                end if
            end if
            if $P'$ has no label then
                add $P'$ to cluster $C$
            end if
        end for
    end if
end for
```

Time Complexity: $\mathcal{O}(n^2)$

DBSCAN is more robust to noise and allows to handle the outliers. Unlike K-Means, which performs best with globular-shaped clusters, DBSCAN treats more complex cluster shapes.

Dimensionality Reduction

Dimensionality reduction helps in simplifying complex datasets by reducing their feature space while preserving essential information. High-dimensional feature spaces introduce the phenomenon known as *curse of dimensionality*: an increasing number of attributes (i.e. dimensions) flattens the distance between different data points, making it harder to separate them into categories or clusters depending on their value [26]. Therefore, Dimensionality Reduction techniques such as *Principal Component Analysis* (PCA) contribute to simplify the Machine Learning models by removing the less informative dimensions while retaining the most useful information.

Chapter 3

Company Data Integration framework

Mediamente Consulting developed a Data Integration framework for ETL and Data Warehousing which enables a standardized and easily maintainable data pipeline through the implementation of *metadata tables*. The company framework develops in three layers and an additional management layer. The Staging Area stores data from the sources without any transformation and feeds the following steps of the ETL process. The Relational Data Storage serves as a central reconciled data repository and houses a cleansed and normalized version of original data. The Dimensional Data Storage is the layer where integration among sources is performed and data is prepared to be loaded into the Data Warehouse or another dimensional analysis storage system. The management layer houses metadata and information about the process [42].

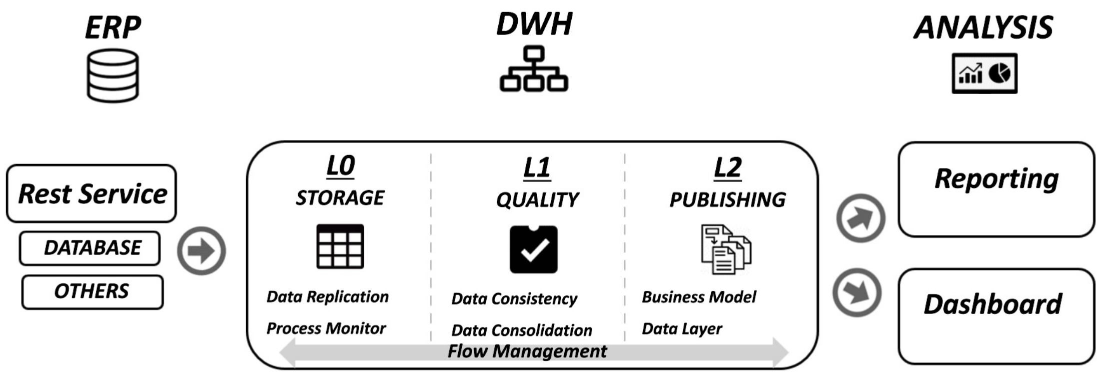


Figure 3.1: The company framework is composed of a distinct layer for each data preparation step: storage, data quality and integration, publishing.

3.1 L0 - Staging Area

The staging area houses a replica of the raw data from the diverse sources, such as relational DBMSs and files provided by the customers. Given that records are replicated without modifications, there exists the potential to store inaccurate or incomplete data; nevertheless, data cleansing is not required at this stage. Data Ingestion is performed in batches on a periodical schedule for each table or document inside a job identified by a *JOBID*. Each job is independent and performs an iteration of the load operations for a single table of a specific source and it is managed by the metadata tables *FLOW_MANAGER* and *TABLE_MANAGER*. Three kinds of tables are implemented in the ingestion stage: delta tables, staging tables, error tables.

DLT tables

Delta tables (DLT tables) store the incremental updates from the sources. Each source table has a corresponding DLT table with the same schema and provides new records through three mechanisms:

- **CDC.** A Change Data Capture system allows to intercept only the data variations (*delta*) between the current iteration and the previous one. In this scenario, DLT tables are ready for the next steps. CDC is generally employed for optimizing the load jobs from tables with abounding records .
- **MINUS.** When CDC is not available on densely populated tables, changes can be detected with a MINUS set operation. For this purpose, a staging table must store temporarily the current snapshot and the one from the previous loading iteration.
- **FULL.** The import operation in FULL mode involves a copy of the whole table or file for each iteration. This pattern suits the ingestion of small-sized sources and records with low variability, for which frequent updates are not required. A FULL load is performed when an Initial Load is required for tackling new sources or populating a Data Warehouse for the first time (*refresh*).

In order to address possible data losses or a partial loading during the ETL process due to errors, a historicized version of the DLT table is required. A straightforward approach can be adopted by keeping the history directly on the DLT tables and partitioning by JOBID. Another solution is the creation of DLT shadow tables marked by the prefix "DLT_HIS", which are partitioned by JOBID with a retention policy determined by the amount of ingested data. Since records from previous iterations are stored in separate tables, DLT tables do not require to retain the whole history of data, hence loading can be performed with a *truncate/insert* pattern, hastening the ingestion phase.

STG tables

Staging tables (STG) contain the records acquired in FULL mode from documents and tables for the specific purpose of feeding DLT tables with the result of a MINUS set operation between the current snapshot of the source and the antecedent one.

The MINUS operation is obtained by two distinct steps. First, the *positive* and the *negative* set subtractions are performed on two instances of the same table in order to extract the new or updated records from the positive MINUS and the deleted records from the negative MINUS[1]. Afterwards, the set union gathers the results of the two operations in a single place and the flow of data is redirected to the corresponding DLT table.

The output DLT table schema bears an additional field (the flag *FLG_NEG*) in order to differentiate the new records from the deleted ones.

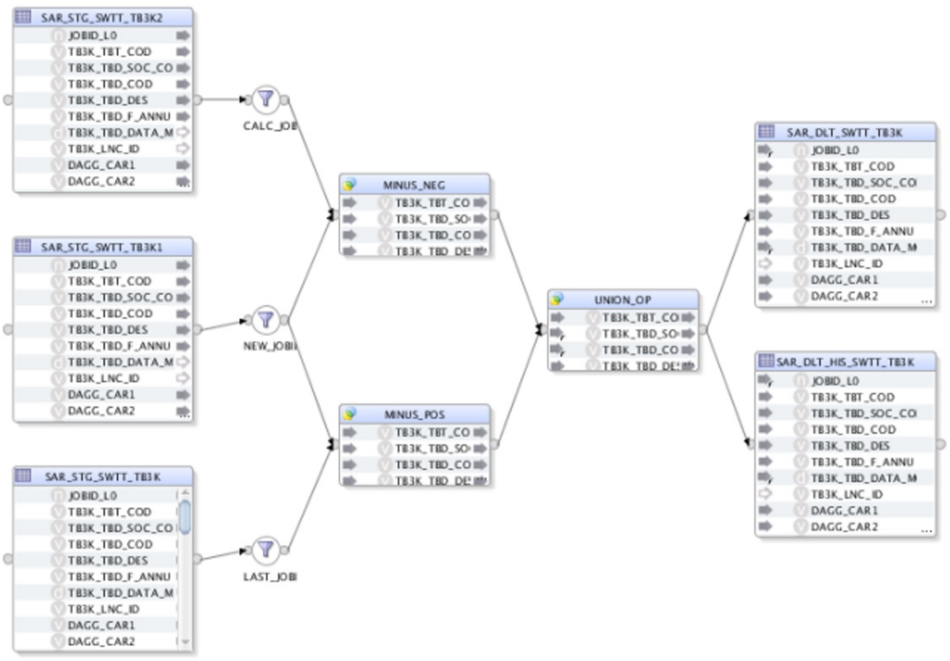


Figure 3.2: An example of a MINUS operation performed in an Oracle Data Integrator environment.

[1]The set subtraction operation is not commutative, i.e. given two sets A and B, $A \setminus B \neq B \setminus A$

3.2 L1 - Relational Data Store

The L1 section is the effective core of the ETL process, since the most complex and computationally expensive procedures and transformations are carried out here. Data is subject to data quality checks and business rules to ensure its integrity, correctness, and dependability; subsequently, a process of normalization is applied with the aim of populating a relational storage with the most granular level of data available. Finally, the effective integration is performed among the various data sources in order to prepare data for the subsequent visualization and dimensional analysis.

OK tables

The Data quality process evaluates data freshness, accuracy, completeness, consistency, and trustworthiness in a specific context in order to empower business decisions along with *data governance* programs [43]. The aforementioned activities of data quality and the application of business rules are performed on data coming from DLT tables and result in the population of new tables with records that are functional to the business needs in the means of data effectiveness. These tables are called *OK tables*. Three classes can be identified in this step:

- **Referential integrity checks.** For each table that contains at least a foreign key, a join operation with the corresponding reconciled table (L1) is executed.
- **Quality checks.** Field-dependent constraints are validated through multiple steps. For instance:
  - empty *NOT NULLABLE* fields are replaced with default values in order to avoid potential errors due to missing data
  - data types are aligned to the ones specified in the requirements
  - length constraints, string encoding, and invalid values are checked to comply to the model: for instance, the date *'2023-02-29'* is inadmissible, since 2023 is not a leap year
- **Business rules validation.** Raw records can undergo additional checks that depend on specific requirements, e.g. an integer field is required to be inside the range $[0, 10]$, therefore a record having the value 11 in that field must be discarded

Records that are unsuited for reconciliation are treated in expressly defined tables for error management.

ERR tables

Although error tables (ERR) are included in the L0 layer, their discussion and relevance are being examined within L1 data quality step. Error tables are geared to store records that do not meet the data quality rules. Unsuitable data points are discarded temporarily for the purpose of supplying them to the L1 mappings during the next iteration. For instance, a record can fail referential integrity checks as a result of the absence of the records to which it should be joined, hence it is kept in the error table until the missing reference issue is resolved.

Error management is treated within the ERR tables by extending the original schema with a description of the error, along with the technical fields inherited from DLT tables. Moreover, a retention period is defined after which the ERR table records are permanently discarded in order to avoid potential bottlenecks.

In a production scenario, errors are currently reported to the person in charge of manually revision, however error correction tools powered by Machine Learning can be employed to automate this step.

ODS tables

Operational Data Store (ODS), or *Relational Data Store* tables are the implementation of the conventional L1 layer. ODS tables provide and establish a centralized and normalized data model that encompasses all the aspects of the company requirements. L1 tables aim to provide a consolidated and historicized version of the relational source data ready for integration. Since Relational Data Store needs to be historicized, ODS tables are populated with MERGE statements in an *Incremental Update* fashion. In order to allow the records update, a primary key constraint must be defined on the physical key of the table. Moreover, tables are not subject to compression since frequent updates are performed, especially when many iterations are carried out in a single load process.

MDM tables

Master Data Management (MDM) tables fulfill two critical roles in the ETL process: the integration from different sources and data enrichment. Data integration in the current stage consists in gathering matching data by joining two or more ODSs, therefore accumulating attributes from many sources in a central table.

Since integration is already performed at a transaction level in RDBMSs in real use case scenarios, Master Data Management tables integrate only the new sources in order to avoid redundancy [8]. Moreover, integration can prioritize a source over another depending on the type of performed join: for instance, when joining an ERP table with an API responses table, the former can be modeled as more relevant by being on the left side of a LEFT OUTER JOIN.

MDM tables which contain descriptive attributes (*dimension tables*) serve the other purpose of assigning a *surrogate key* as well. A Surrogate Key (SK) is a unique identifier assigned to each record of a table with the aim of superseding the natural key. The SK has no meaning outside its role of record identification and it is independent from all the other attributes inside the table. The most prominent gain from the use of a surrogate key is the increase in performance, since querying tables through surrogate keys is faster than compound or more descriptive keys, especially in expensive join operations. In addition, SK provide a solution which is efficient in terms of storage and more robust to changes.

Surrogate keys are generally defined with compact data types, such as integers, with a growing pattern. For instance, in an Oracle environment, surrogate keys are defined as *sequences*, while in a SQL Server scenario the surrogate key field is declared as an *IDENTITY*.

Surrogate keys are not assigned to measurement records according to the dimensional model later treated in Section 3.3.

OUT tables

After integration and enrichment, records are prepared for the publishing phase by collecting the required information for the multidimensional schema and transforming the records in order to fit the Data Warehouse model. OUT tables do not manage physical keys since data representation is not transactional: they acquire the surrogate keys, preparing records for the visualization layer, especially for snowflake schemas. By merging various sources into one, multiple representations of the same value can be met, therefore the general pattern is choosing a common representation to prevent misalignment errors.

3.3 L2 - Dimensional Data Storage

The visualization layer consists in the Data Warehouse, where the model for multidimensional analysis is stored. The L2 layer serves both as the central storage for business data to be queried for decision-making processes and as a source for smaller Data Marts or subsequent analyses.A Data Warehouse model is composed by one or more *fact tables*, which store the measurements of a business application, and more *dimension tables*, which describe the dimensions of interest for the analysis.

3.3.1 Fact tables

A fact table stores all the measurements of interest in relation to the dimensions of analysis. Each record is uniquely identified by a compound primary key, which is composed by all the surrogate keys that link the table to each dimension of analysis, and a time reference (usually modeled as an extra dimension). Records can be interpreted as data points in a n-dimensional space where each dimension is a finite and discrete domain defined by the dimension tables.

Measurements, or *metrics*, are quantitative descriptions that address specific business information, empowering dimensional analysis on a variable level of granularity. Whereas dimensions are always required to define a data point, presence and granularity of the metrics are determined by the type of the fact table:

- *Transaction* fact tables store the most fine-grained data (records are stored with a transaction level granularity)
- *Snapshot* fact tables store metrics which have been aggregated in time windows
- *Factless* fact tables model an "attendance" fact, with no need for metrics, since information is represented by the mere existence of the record

Fact tables are fed with an append pattern, since they are required to ingest large quantities of records for each iteration, and are generally partitioned by the time dimension in order to speed up time-based queries, which are the most frequent in a decision-making application.

3.3.2 Dimension tables

Dimension tables collect descriptive information of each dimension of the analysis. The records of a dimension table are identified by a simple primary key, which is the surrogate key assigned in the MDM stage, and contain textual fields that entirely characterize an aspect of the analysis. These tables are usually denormalized, trading a negligible increase in stored data with a significant improvement in query time performance.

3.3.3 Star schema and snowflake schema

Two architectural patterns can be adopted in a Data Warehouse model planning: the **star schema** or the **snowflake schema**.

In a star schema, the fact table is the only table to use foreign key constraints on the surrogate keys in order to be joined with other tables, presenting a denormalized structure.

The snowflake schema presents a fully normalized structure for dimension tables, introducing a greater complexity in the DWH model. Querying many lookup tables requires a JOIN clause for each level introduced in the hierarchy, leading to a slowdown in visualization, but the trade-off may be relevant for specific Business Intelligence tools that favor normalization.

	Star schema	**Snowflake schema**
Dimension tables	One table for each dimension containing all the information	Each dimension has a hierarchical structure with many tables
Normalization	Denormalized	Highly normalized
Redundancy	High	Absent
Storage performance	High due to redundancy	Low due to normalization
Query performance	High (few join operations)	Low (many joins required)

Table 3.1: Main differences between star schema and snowflake schema

3.4 Metadata layer

In addition to the three standard layers, and additional layer is implemented in order to describe and manage metadata, which is descriptive information that pertains to other data.

Metadata can be classified either on its employment as *internal* or *external* metadata, or depending on its scope as *global* metadata or *process* metadata [42]:

- **Internal** or **structural** metadata refers to data that portrays the structure and organization of data, such as table schemas, field data types, transformation status and outcomes, etc. Internal metadata provides information to the manager about the architecture of the system.
- **External** or **content** metadata is user-scoped data about the context of data. For instance, content metadata supplies data definitions, measurement units, and information about the meaning of its content.
- The scope of **global** metadata is the whole system and involves all the processes in order to provide a common reference for the synchronization of the activities and a shared granularity among data
- **Process** metadata is determined by the data source or the process to which it refers, outlining its characteristics and status.

The company framework encloses two metadata tables: *FLOW_MANAGER* and *TABLE_MANAGER*. Moreover, additional technical fields are declared in each table:

- `JOBID`, contains the load iteration identifier; it is split into `JOBID_INS` and `JOBID_UPD` for tables populated with MERGE operations with the aim of tracking both the first insertion and the last update

- `INS_TIME` is the insertion timestamp in the table. Just as JOBID, the additional field `UPD_TIME` is employed for tables that allow updates.

3.4.1 Flow management table

The *FLOW_MANAGER* table stores information about the history and the unfolding of ETL iterations, detailing the current state and the outcome of each ETL layer for every ingestion flow. Its purpose is synchronizing the diverse ETL stages and processes and monitoring the advancement of the pipeline.

Each record of the FLOW_MANAGER table is uniquely identified by the iteration *JOBID*, an *IDENTITY* and a *GROUP*, and its *LEVEL*.

- **JOBID.** Describes an ETL iteration instance with a timestamp in the format `YYYYMMDDHHmmss` (`YYYYMMDDHH24MISS` for Oracle systems).

- **IDENTITY.** The identity defines and aggregates flows with a common working area, e.g. the ingestion source, the business area.

- **GROUP.** The group field (named *GRP_NAME*) assembles the tables involved in the pipeline into functional entities, which collect objects with functional dependencies, such as all the tables related to the product, or the point-of-sales description tables.

- **LEVEL.** Defined by the field `NUM_LEVEL`, it is a numerical representation of the ETL stage performed (L0, L1, or L2).

Records gather information about the current **STATUS** of the process with a numeric value in the domain defined in Table 3.2 and the advancement of the **LOAD** operations in the next stage with a binary value (0 or 1). Furthermore, a timestamp for the beginning of the process (`START_DATE`) and one for its termination (`END_DATE`) are present.

Value	Status meaning
0	The job is complete without errors
1	The job is currently running
2	Job data is ready to be loaded in the next stage
-3	There are errors, job is aborted

Table 3.2: Possible values of the STATUS field in FLOW_MANAGER table.

3.4.2 Table management table

The *TABLE_MANAGER* table is responsible of tracking information about the iteration flow with regards to the table. Records are characterized by the following fields, which constitute the primary key of the table:

- `IDENTITY`
- `GRP_NAME`
- `NUM_LEVEL`
- `TABLE_NAME`
- `JOBID`

Four fields are defined in order to supply insights on the ETL processes in relation to each table:

- `CAP_ID` is an integer identifier for the timestamp of the most recent data point in the source table; it shares the same format as JOBID and serves as a reference for the synchronization of delta operations performed during the ingestion phase.
- `NUM_ROWS` is a counter of the number of rows produced as output from the current stage; a misalignment in the number of rows between two different stages or an incoherent value among the steps may indicate the presence of ingestion or integration errors, or the failure of data quality checks for some records.
- `LOAD_DATE` stores the timestamp at which the table population has been completed.

3.4.3 Additional metadata tables

In specific major projects, further metadata tables can be implemented in the model in order to furnish a more extensive view on data sources, staging tables and Data Warehouse schemas and execution flow. For instance, data types can be widely described in documentation tables; an error management table can provide a collection of the errors obtained during the data quality checks.

Metadata tables can be subject to business rules and supplementary checks as well, thereby playing a fundamental role during the deployment and maintenance phases of a project. The error management table can be queried in order to retrieve a subset of errors which have been deemed critical by the customer from a business perspective, then consequently processed.

Chapter 4

Case study

The customer requires the automation of a data pipeline for visualizing and performing Business Intelligence tasks on data acquired from two different social media platforms: YouTube and LinkedIn. The current infrastructure relies on the manual intervention of a user, who has access to the social media reporting web pages and daily downloads the required data to obtain the KPIs which have been determined in a previous assessment.

Three purposes can be ascribed to this thesis:

- designing an automated ETL pipeline for the task
- improving the existing Data Visualization and reporting activities by leveraging a Data Warehouse architecture
- providing a solution based on Machine Learning for Data Enrichment, enhancing future analyses and BI reports

The accomplishment of the first two objectives, involving the ETL and Data Warehouse realization, is reached by observing a top-down approach: the DWH model is designed beforehand, followed by the ETL process implementation. The last goal is achieved through the means of Machine Learning Python libraries.

4.1 Data Warehouse model

In order to provide an integrated model to support the business analytics performed by the customer, the Data Warehouse implements a star schema with two distinct fact tables: `FACT_YT_METRICS` for YouTube data and `FACT_LI_METRICS` for LinkedIn data. An exhaustive fact table has been considered; the two data sources present relevant differences and cannot be integrated in a single target nevertheless. In Table 4.1 an exhaustive list of the designed entities is provided.

Table name	**Type**	**Description**
`YT_METRICS`	FACT	YouTube insight measurements
`LI_METRICS`	FACT	LinkedIn insight measurements
`CALENDAR`	DIMENSION	Description of time dimension
`VIDEO`	DIMENSION	YouTube video information
`DEVICE_TYPE`	DIMENSION	Device that generates the insight
`TRAFFIC_SOURCE`	DIMENSION	Source of the YouTube insight
`SUBSCRIBED`	DIMENSION	Status of the user's subscription
`LINKEDIN_PAGE`	DIMENSION	Page visited on the LinkedIn profile

Table 4.1: List of the designed table for the Data Warehouse proposed model.

FACT_YT_METRICS

The YouTube fact table is designed to store metrics at a granularity defined by five dimensions:

- the **date** of the measurement, retrieved by `DIM_CALENDAR`
- the **video** to which the insight is referred, from `DIM_VIDEO`
- the state of a user's **subscription** at the insight time
- the **device type** from which the measurement is recorded, from `DIM_DEVICE_TYPE`
- the insight **traffic source**, which represents the origin of the interaction with the video

An additional dimension is added for the **campaign label** which will be the subject of the subsequent realization of the thesis in Chapter 6.

Metrics can be grouped in three categories:

- **Basic metrics** such as *views*, *estimated minutes watched*, *likes*, *dislikes*, *shares*, which are available for all the videos.
- **Annotation metrics**, which refer to the insights related to the annotation function. Annotations have been discontinued by 2019 and YouTube APIs report annotation data up to 2013 [44]; however, these fields have been included in the analysis in order to provide a more scalable solution for a further ingestion of historical data. *CTR*, the *close rate* and the number of *impressions* are some examples of the available metrics.

- **Card metrics**, which supply values for the Info Cards function. Even though the customer does not make use of info cards at design time, there are plans to implement the function, therefore the metrics have been kept in the final model.

Field name	**Data type**	**Primary key**
`DATE_SK`	int	Yes
`VIDEO_SK`	int	Yes
`SUBSCRIBED_STATUS_SK`	int	Yes
`DEVICE_TYPE_SK`	int	Yes
`TRAFFIC_SOURCE_SK`	int	Yes
`CAMPAIGN_LABEL_SK`	int	Yes
`VIEWS`	int	No
`RED_VIEWS`	int	No
`ESTIMATED_MINUTES_WATCHED`	int	No
`ESTIMATED_RED_MINUTES_WATCHED`	int	No
`LIKES`	int	No
`DISLIKES`	int	No
`VIDEOS_ADDED_TO_PLAYLIST`	int	No
`VIDEOS_REMOVED_FROM_PLAYLIST`	int	No
`SHARES`	int	No
`AVERAGE_VIEW_DURATION`	int	No
`AVERAGE_VIEW_PERCENTAGE`	numeric(38,10)	No
`ANNOTATION_CTR`	numeric(38,10)	No
`ANNOTATION_CLICKS`	int	No
`CARD_CLICK_RATE`	numeric(38,10)	No
`CARD_TEASER_CLICK_RATE`	numeric(38,10)	No
`CARD_IMPRESSIONS`	int	No
`CARD_CLICKS`	int	No

Table 4.2: Brief table description for FACT_YT_METRICS model. Some fields have been omitted.

FACT_LI_METRICS

The fact table which encloses LinkedIn insights is described by three dimensions:

- the **date** to which metrics refer, from `DIM_CALENDAR`

- the **type of device** used, from `DIM_DEVICE_TYPE`. It is available exclusively for the metrics relative to visitors, thus requiring a dummy value when inserting new followers and content-related metrics
- the **page** visited by the user. Another pseudo-value used for followers and content metrics is manually inserted

In a manner analogous to the YouTube fact table, the **campaign** dimension is added to the other three.

Time and *device type* dimensions are shared between the two fact tables, enabling joint analyses for a comprehensive view on the evolution of the social media marketing campaigns towards the business goals.

Field name	**Data type**	**Primary key**
`DATE_SK`	int	Yes
`DEVICE_TYPE_SK`	int	Yes
`LINKEDIN_PAGE_SK`	int	Yes
`CAMPAIGN_LABEL_SK`	int	Yes
`PAGE_VIEWS`	int	No
`PAGE_UNIQUE_VISITORS`	int	No
`ORGANIC_FOLLOWERS`	int	No
`SPONSORED_FOLLOWERS`	int	No
`ORGANIC_IMPRESSIONS`	int	No
`ORGANIC_UNIQUE_IMPRESSIONS`	int	No
`ORGANIC_CLICKS`	int	No
`ORGANIC_REACTIONS`	int	No
`ORGANIC_COMMENTS`	int	No
`ORGANIC_POST_DIFFUSIONS`	int	No
`ORGANIC_INTEREST_PERCENTAGE`	numeric(38,10)	No
`SPONSORED_IMPRESSIONS`	int	No
`SPONSORED_CLICKS`	int	No
`SPONSORED_REACTIONS`	int	No
`SPONSORED_COMMENTS`	int	No
`SPONSORED_POST_DIFFUSIONS`	int	No
`SPONSORED_INTEREST_PERCENTAGE`	numeric(38,10)	No

Table 4.3: Brief table description for FACT_LI_METRICS. Technical fields have been omitted.

4.2 Employed tools

The model implementation is performed with an hybrid approach in order to comply with the diverse requirements at design time:

- a solution that can provide an immediate improvement in the automation of the data ingestion step
- a model that can operate in continuity with the already existing dashboards and Data Visualization tools (such as Microsoft Power BI)
- the cost-effective choice to employ open-source instruments which are compatible with the chosen cloud platform

Among the open-source options, the most suitable pipeline design software is **Pentaho Data Integration** due to its compatibility with the underlying SQL Server architecture. Pentaho jobs are planned to be scheduled on a daily basis on the on-premise server of the customer for the ETL section, where a temporary replica of the Data Warehouse is stored.

The Load operation stores data into a Data Warehouse created on Google BigQuery as a first step of the full migration to Google Cloud Platform services.

All the code sections are developed in *Bash* and *Python 3.11.6* with the aid of the modules:

- `google-api-python-client` for the API calls
- `google-auth-oauthlib` for Google APIs authorization
- `google-cloud-bigquery` for loading data into Google BigQuery
- `selenium` for the WebDriver implementation
- `pandas` and `numpy` for data internal representation
- `scikit-learn` for clustering algorithms
- `seaborn` and `matplotlib` for data visualization

Chapter 5

Data ingestion and integration

The first step in the data pipeline planning is the ingestion from the data sources. In the case study, the customer needs to integrate two sources, which are two different social media platforms: YouTube and LinkedIn. The former provides an easy access to its own APIs, while the latter requires further methods to automate the extraction process, such as the webdriver Selenium. The ingestion and the integration layers are encompassed in a Pentaho Data Integration job, which partially replicates the company framework in order to perform ETL operations. The resulting Data Warehouse is loaded on Google BigQuery as a first step for a migration on Google Cloud Platform.

5.1 Data ingestion layer

The ETL pipeline initially launches the L0 layer, which consists of three distinct jobs that acquire all the records from the two YouTube channels and from the LinkedIn profile produced from the parameter *CAP_DATE* of the last job, which is the last date present in the records of the most recent incremental update (stored for each table in the *CAP_ID* field of TABLE_MANAGER). Since both YouTube and LinkedIn do not provide data for the current day, the latest data point is dated one day before the request timestamp. The CAP_DATE parameter is assigned to the variable *START_DATE* and the variable *END_DATE* is calculated; for the initial load, the default value "2000-01-01" is chosen in order to guarantee a complete history of the available data.

Each table load process is uniquely identified in the TABLE_MANAGER, so that the fields *IDENTITY* and *GRP_NAME*, passed to the jobs as parameters, allow them to identify the operation in progress.

The Bash scripts *run_youtube_api.sh* and *run_linkedin_checks.sh* are mutually exclusively called depending on the data source ingested by the current job:

- **run_youtube_api.sh**, the *"YouTube load script"*, sets up a virtual environment, acquires the parameters and log options, and finally runs the Python script `youtube_main.py`, which is responsible of providing a given DLT table with the most recent records by calling the YouTube API. For instance, the command line to execute the script in order to insert the metrics of the channel "YTA" in a date range between July 2023 and August 2023 into the table DLT_YT_MAIN_METRICS is

  ```
  python youtube_main.py -t DLT_YT_MAIN_METRICS -u YTA \
  --start_date 2023-07-01 --end_date 2023-08-31
  ```

- **run_linkedin_checks.sh**, the *"LinkedIn load script"*, runs two scripts. First, the `linkedin_main.py` Python script is launched to simulate the user interaction with LinkedIn website in order to download the current delta of data. Afterwards, the same operations as the YouTube load scripts are performed by executing the `li_checks.py` Python script for each table. Since a single LinkedIn user is included in the data sources, the user parameter is not present; in addition, a single date acting as CAP_DATE is passed as parameter to both scripts.

  E.g. The commands executed by the LinkedIn load script for the staging table DLT_LI_FOLLOWER_METRICS starting from July 2023 is

  ```
  python linkedin_main.py -d 2023-07-01
  python li_checks.py -t DLT_LI_FOLLOWER_METRICS -d 2023-07-01
  ```

Both scripts produce one or more JSON files containing input data for the subsequent step of the pipeline. Each job's flow is put into a wait state until the resulting JSON files are ready for elaboration, and calls the DLT table mapping thereafter. The last operation performed by load jobs is the insertion of the outcome of the job and additional metadata in the management tables through the execution of the EndProcessTable transformation. Figure 5.1 illustrates the complete visual mapping of the load jobs.

5.2 YouTube data sources

The primary requirement for data ingestion from YouTube is the availability of source data for the objective KPIs, which need metrics such as the number of views for a certain video, the daily amount of likes, shares and comments for a video depending on the subscription status. YouTube exposes two types of API services for

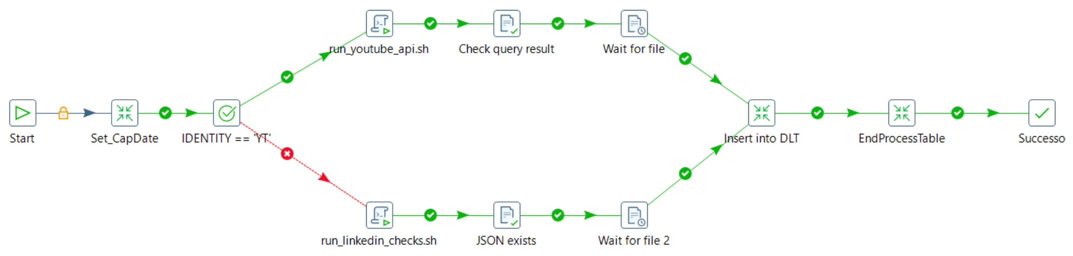


Figure 5.1: Load job for each DLT table in Pentaho. Cap date is set first, then a different script is called depending on the source. Finally, data is written in the DLT tables and metadata is stored in management tables.

the purpose: *YouTube Data API* and *YouTube Analytics and Reporting API*. API requests, which are performed at the L0 layer of the ETL model, employ OAuth 2.0 authorization methods to grant access to private data and provide records containing both metrics and descriptive attributes. The ingestion is designed to follow an incremental pattern and API responses are stored in temporary JSON files in the staging area, which are pre-processed before feeding the ETL pipeline.

The YouTube load script is the primary responsible of the ingestion phase by instantiating one of two possible API flows, depending on the table execution parameter.

The first flow grants access to the YouTube Data API service in order to retrieve a comprehensive list of videos from the channel with relative descriptive information and stores it in the staging JSON file `tmp_DLT_YT_VIDEO_parsed.json`; a cleansed version of the obtained video identifiers is memorized as a list in the temporary file `videos.json`, which is provided as input for the second flow.

The second kind of flow queries the YouTube Analytics API service for obtaining the metrics for each video in the temporary file and stores the results in two staging files:

- `tmp_<table-name>.json`, which contains the raw API response as a backup (e.g. `tmp_DLT_YT_VIEWS_PER_DEVICE.json`)
- `tmp_<table-name>_parsed.json`, which feeds the next steps of the pipeline (e.g. `tmp_DLT_YT_VIEWS_PER_DEVICE_parsed.json`)

Once the temporary files are ready, a table-specific `API_to_DLT` transformation loads the records into the respective DLT table.

5.2.1 OAuth Authentication

Since both Data and Analytics API services access private user data on behalf of its owner, *OAuth 2.0* authentication and authorization mechanisms must be implemented. A pair of strings, the *client ID* and the *client secret*, is generated in advance to identify the application that invokes the APIs and is stored on the endpoint. Applications query the OAuth server requesting an *authorization code*, a unique string that identifies a set of permitted operations for an API service user for a specific application. Therefore, the user authenticates on its Google login page and provides consent on the API access scope. The authorization code is thereby returned to the application, which exchanges it with the authorization server for another couple of strings, the *access token* and the *refresh token.* The access token is memorized by the application until its expiration and it is attached to each API call to the YouTube server, which supplies its response without the user's need to authenticate again. Since access tokens have a limited validity in time, the refresh token must be exchanged for a new access token when the previous one expires.

In a server-to-server architecture, the implementation of an automated authentication flow involves the employment of a *service account*, which is an application account that serves as a mediator between user and server, in a two-legged OAuth flow [45]. Despite service accounts empower a communication pattern with YouTube services without requiring user's interaction, their usage is limited to Google Workspace domains, which must be directly managed by the responsible organization. The customer aims to fully migrate on Google Cloud Platform, however they requested to configure their domain at a second time, leading to choose the standard authorization approach for the current project.

5.2.2 YouTube Analytics and Reporting API

YouTube exposes two kinds of reporting and analyzing views, interactions and other performance metrics on a given YouTube channel, called *YouTube Analytics API* and *YouTube Reporting API.* The main difference between the two services resides in the availability of reports related to system-managed ad campaigns [46]. Despite YouTube Reporting API being the most suitable API service, since it targets data mining applications and tools, it only provides predefined daily reports, being unsuitable for the planned in-dept analysis. YouTube Analytics API service has therefore been chosen with the aid of *google-api-python-client* module.

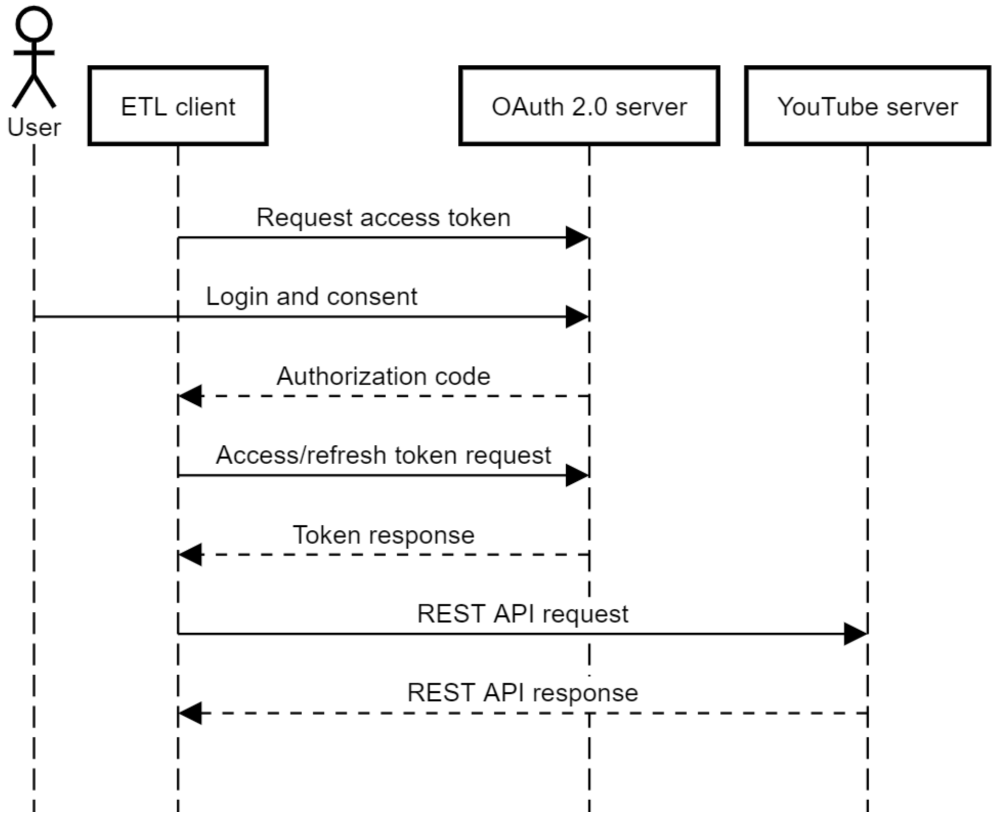


Figure 5.2: Client-Server pattern for YouTube API services

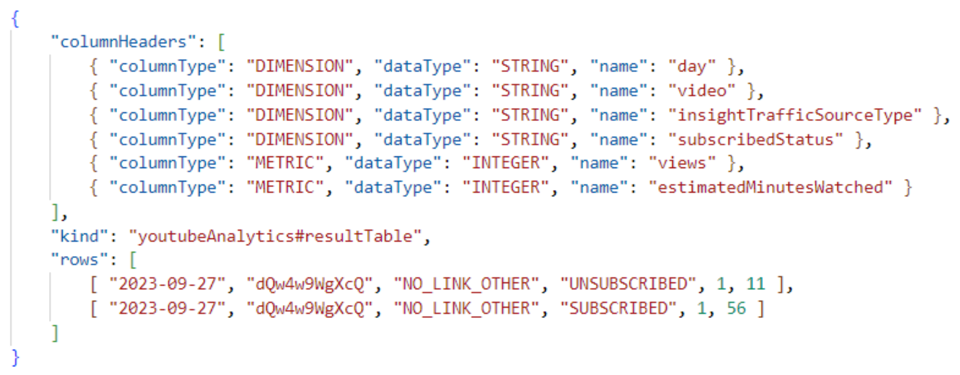
Every YouTube Analytics API call requires the following parameters:

- **ids** the channel identifier, set to `"channel==MINE"` for the channel owned by the authenticated user
- **video** the unique video identifier, applied as a filter
- **dimensions** the comma-separated list of dimensions that define the response granularity
- **metrics** the list of measurements of interest, separated by commas
- **sort** the key of the response sorting, required by all the API reporting queries, set to `"day"` (the ascending order is set when not specified)
- **startDate** defines the time interval lower bound of the incremental update
- **endDate** represents the last day of the time interval, generally set to one day before the execution date

The two JSON files that store a YouTube API response have substantially different formats.

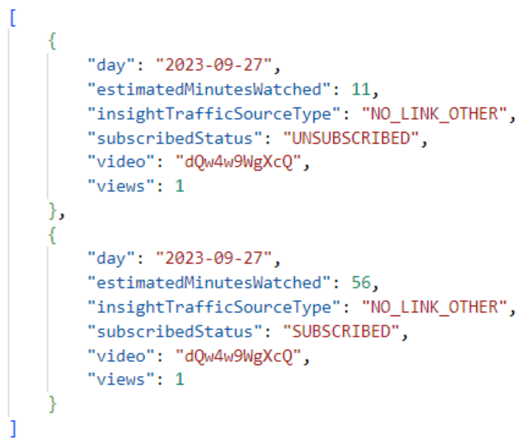
The raw version of the response is a JSON object with three key-value pairs: a list of objects that define the column names and types, an API tag, and a list of records (Figure 5.3a).

The cleansed version houses a list of objects with the same structure as the respective DLT table in a key-value pair fashion (Figure 5.3b). Since the quantity of requested fields is limited, data types are dropped when parsing the raw response and tackled at a later stage, during data quality checks.



```
{
    "columnHeaders": [
        { "columnType": "DIMENSION", "dataType": "STRING", "name": "day" },
        { "columnType": "DIMENSION", "dataType": "STRING", "name": "video" },
        { "columnType": "DIMENSION", "dataType": "STRING", "name": "insightTrafficSourceType" },
        { "columnType": "DIMENSION", "dataType": "STRING", "name": "subscribedStatus" },
        { "columnType": "METRIC", "dataType": "INTEGER", "name": "views" },
        { "columnType": "METRIC", "dataType": "INTEGER", "name": "estimatedMinutesWatched" }
    ],
    "kind": "youtubeAnalytics#resultTable",
    "rows": [
        [ "2023-09-27", "dQw4w9WgXcQ", "NO_LINK_OTHER", "UNSUBSCRIBED", 1, 11 ],
        [ "2023-09-27", "dQw4w9WgXcQ", "NO_LINK_OTHER", "SUBSCRIBED", 1, 56 ]
    ]
}
```

(a) Raw JSON file



```
[
    {
        "day": "2023-09-27",
        "estimatedMinutesWatched": 11,
        "insightTrafficSourceType": "NO_LINK_OTHER",
        "subscribedStatus": "UNSUBSCRIBED",
        "video": "dQw4w9WgXcQ",
        "views": 1
    },
    {
        "day": "2023-09-27",
        "estimatedMinutesWatched": 56,
        "insightTrafficSourceType": "NO_LINK_OTHER",
        "subscribedStatus": "SUBSCRIBED",
        "video": "dQw4w9WgXcQ",
        "views": 1
    }
]
```

(b) Parsed JSON file

Figure 5.3: Examples of a raw JSON file and the relative parsed version for YouTube API.

5.3 LinkedIn

LinkedIn provides a visualization page on the Analytics section of the website for reporting data about visitors, followers and content engagement. At project time, customer manually downloaded reports from the export function on the page, thereby obtaining an Excel spreadsheet for each tab of the LinkedIn Analytics tool with a filename containing the name of the LinkedIn profile and the download timestamp in epochs (e.g. `customer-page-name_content_1690808129824.xls` for the content metrics of "Customer Page Name", downloaded on 2023-07-31 at 12:55:29).

Although LinkedIn APIs are available, their usage needs the authorization from the platform: in order to supply an automated time-saving solution while the application is under approval, a demo of the automated LinkedIn ingestion process has been developed by leveraging the implementation of a browser automation agent with the Selenium WebDriver framework in Python.

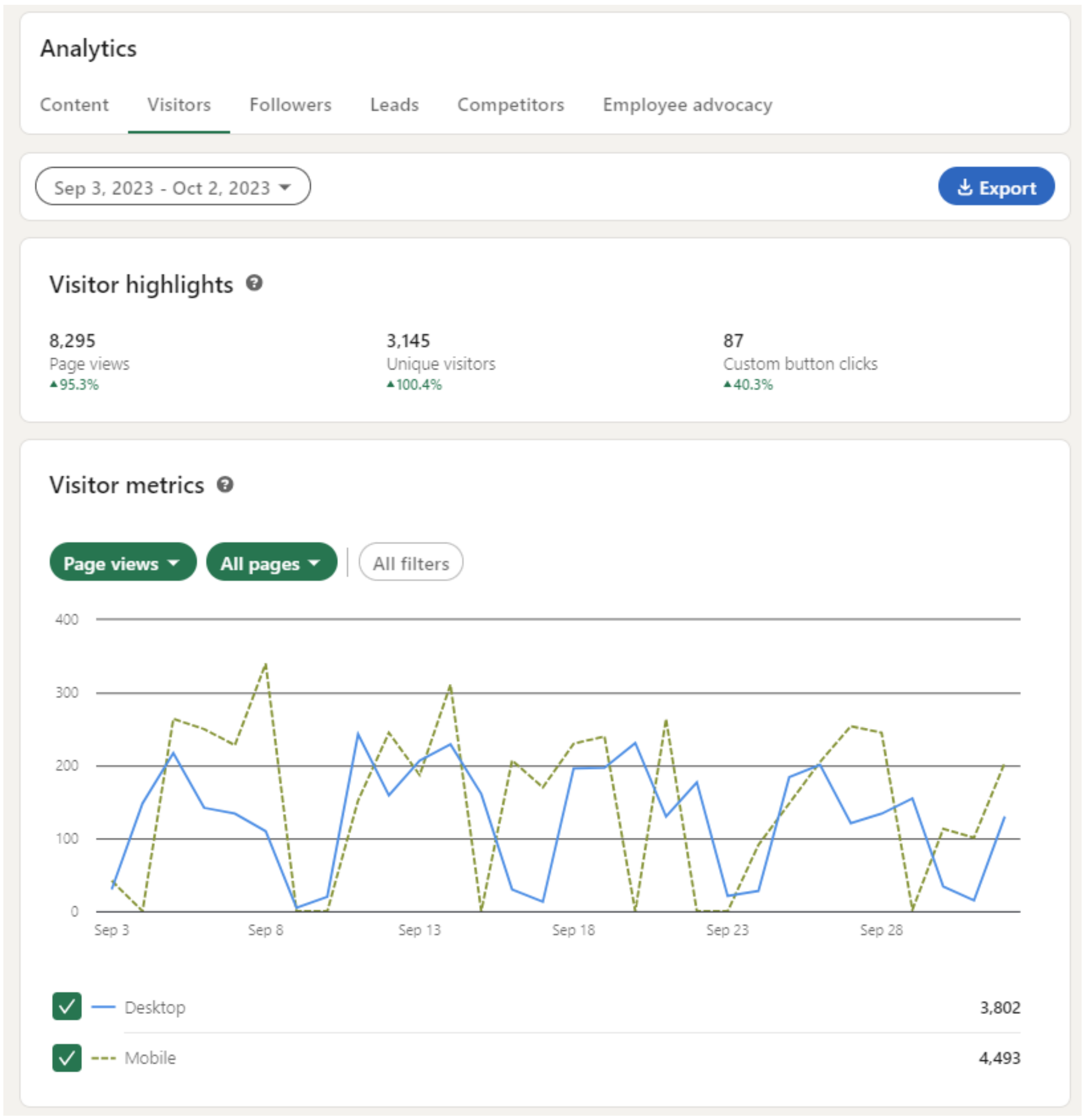
5.3.1 Automation with Selenium

Selenium is a framework that empowers the automation of user interaction with a web browser by defining an instance of the WebDriver interface [47]. The *driver*, which is responsible of the web browser environment initialization, is capable of accessing and manipulating Document Object Model (DOM) of a HTML web page and interacting with it, simulating the user behavior during navigation on a website.

The LinkedIn load script is designed to run seamlessly during the migration from the manual ingestion to the automated procedure, hence a staging folder where all the spreadsheets are stored is created beforehand and it is accessed during the load job.

The script creates an instance of a Google Chrome driver and navigates to the LinkedIn sign in page. The driver interacts with the web page by locating the HTML elements through the *XML Path Language* (XPath) and putting the process into a WAIT status until the target element is *available*. When the pointed object has the requested properties, such as a button being *clickable* or a text box allowing text input, the corresponding action is carried out and the next iteration of the navigation flow is processed. A registered profile with viewer permissions on the company analytics is logged in, resulting in a redirection to the LinkedIn homepage. Security best practices require user's credentials not to be hard-coded, hence the password string is typed in a command line prompt during development. An enciphered version of the sensitive login information is going to be stored at a production stage on the on-premise storage, where other security policies have already been implemented to protect reserved data.

The browser instantiated by the driver redirects to the LinkedIn Analytics page and queries the user interface by clicking the Export button first, then selecting the date range between CAP_DATE and the last functional date for analysis (i.e.



8,295	3,145	87
Page views	Unique visitors	Custom button clicks
▲95.3%	▲100.4%	▲40.3%

✓	Desktop	3,802
✓	Mobile	4,493

Figure 5.4: LinkedIn Analytics web page example

one day before the current timestamp). As soon as the report is ready, the file is downloaded in the staging directory, which has been defined in the driver options. A download operation is performed for all the three required files:

- `<customer-page-name>_content_<epoch-timestamp>.xls`
- `<customer-page-name>_followers_<epoch-timestamp>.xls`
- `<customer-page-name>_visitors_<epoch-timestamp>.xls`

The WebDriver script checks the existence of Excel files for the current day, then it locks a shared resource to ensure files are downloaded exactly once. The script instances that do not perform the download operations are put on a WAIT state until the leading process notifies them that files are ready to be read. By keeping the download and actual ingestion steps decoupled, LinkedIn load processes can benefit from further parallelization.

Finally, the main instance produces a JSON file listing all the spreadsheets involved in the current load, which is fed to the next stages of the pipeline.

5.3.2 Ingestion from spreadsheets

Unlike YouTube API calls, which provide targeted queries for the specific purpose of populating DLT tables, LinkedIn spreadsheets are organized in separate sheets, each one containing a standard report.

- In the *content file* two sheets are available: the first presents the overall daily content engagement metrics with a distinction between organic and sponsored measurements, while the second lists the descriptive attributes of the content published during the selected time interval and some aggregated metrics.
- The *followers file* supplies a main sheet with the daily metrics about organic and sponsored followers, as well as a sheet for aggregate data on different dimensions such as location, job characteristics and the job field of the followers. Despite each sheet in the file being mapped to a separate DLT table, mappings are carried out in a single transformation (`API_to_DLT_LI_FOLLOWER_METRICS`, illustrated in Figure 5.6).
- The *visitors file* presents the same sheets structure of the followers file, apart from the available fields for the daily metrics, which are split according to the visited page and the device used.

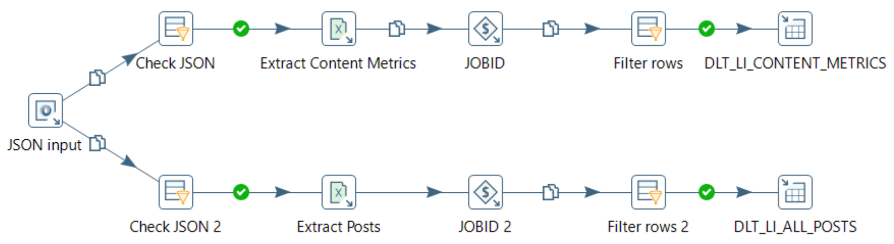


Figure 5.5: LinkedIn content data ingestion from spreadsheets in DLT tables.

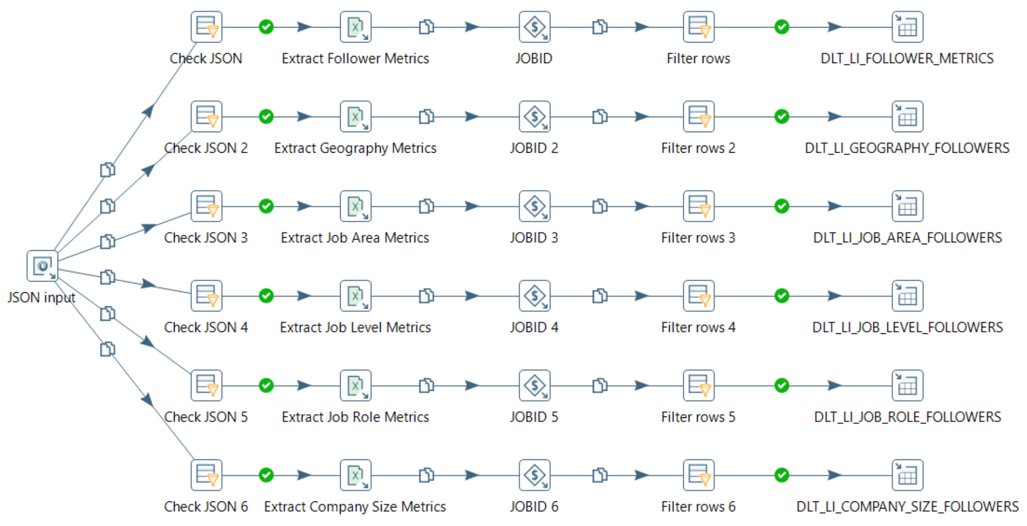


Figure 5.6: API_to_DLT_LI_FOLLOWER_METRICS transformation. The JSON produced by the first script provides a list of Excel files containing six different sheets, each one being processed separately and feeding a different DLT table.

5.4 Data quality

The Data quality step is performed both on DLT tables and records stored in the ERR tables which have been discarded from previous load iterations. The two input sources undergo referential integrity checks, NULL values for NOT NULLABLE fields are standardized and data type constraints are applied. Qualified records are then inserted into OK tables.

Data quality on YouTube tables

For the YouTube branch of L1 layer, every record represents measurements referred to a video in the respective table: since the referential integrity checks are carried out on the video dimension, which is managed by the pipeline, a complete parallelization is not feasible. The most efficient approach consists in applying data quality rules on the video dimension and performing data validation on the ODS_VIDEO table before a parallel execution of the L1 operations on other tables. By these means, a lookup can be performed on *videoId* for the tables DLT_YT_MAIN_METRICS, DLT_YT_VIEWS_PER_DEVICE, and DLT_YT_VIEWS_PER_TRAFFIC_SOURCE, for which the field has a foreign key constraint.

Additional dimensions, such as the viewer's *subscription status*, the *device* used for playing the video and details on the *origin of the traffic* cannot be retrieved by ad-hoc API calls and must be obtained from other staging tables.

The following tables are thereby populated during the step from DLTs to OKs:

- **OK_YT_SUBSCRIBED** describes the available subscription statuses for all the YouTube data points, and its records derive from the staging table DLT_YT_VIEWS_PER_DEVICE. A supplementary record for NULL values is inserted manually during the mapping.
- **OK_YT_DEVICE_TYPE** bears the principal descriptive attribute for metrics that depend on the device type. Its ingestion source is the table DLT_YT_VIEWS_PER_DEVICE and a dummy value for NULL records is added as well. Despite being scalable, the set of possible values is limited by YouTube APIs:
  - `DESKTOP`
  - `GAME_CONSOLE`
  - `MOBILE`
  - `TABLET`
  - `TV`
  - `UNKOWN_PLATFORM`
- **OK_YT_TRAFFIC_SOURCE** stores the descriptive information about the origin of interactions with the videos. The extraction process from the table DLT_YT_VIEWS_PER_TRAFFIC_SOURCE follows the same pattern as the previous ones, with a placeholder value for NULL records. Similar to device types, the insight traffic sources are determined by YouTube APIs, and the following are examples of potential values:
  - `ADVERTISING`
  - `EXT_URL`
  - `NO_LINK_EMBEDDED`
  - `NO_LINK_OTHER`
  - `NOTIFICATION`
  - `PLAYLIST`
  - `SHORTS`
  - `SUBSCRIBER`
  - `RELATED_VIDEO`
  - `YT_CHANNEL`
  - `YT_OTHER_PAGE`
  - `YT_SEARCH`

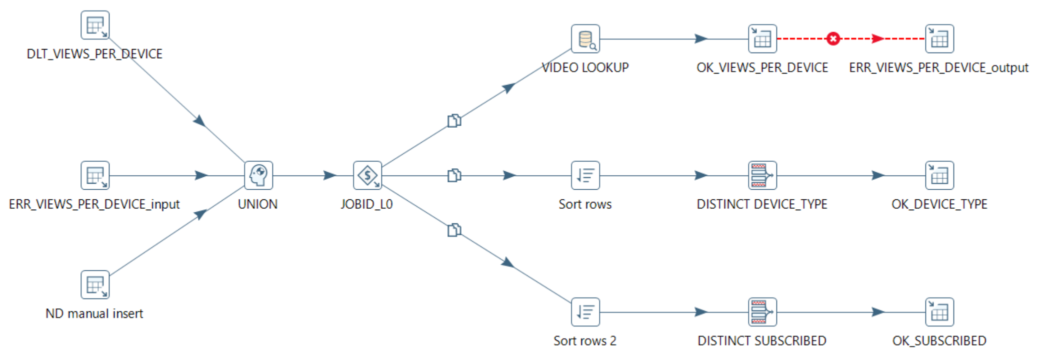


Figure 5.7: DLT to OK table mapping for metrics relative to the device type. Two further OK tables are obtained with this transformation.

Data quality on LinkedIn tables

DLT tables for LinkedIn data do not require referential integrity checks since dimension tables are not available at the data quality stage: they are rather derived from validated records during the Operational Data Store population. Three groups of staging tables are identified depending on the sequence of transformations applied from DLT to ODS:

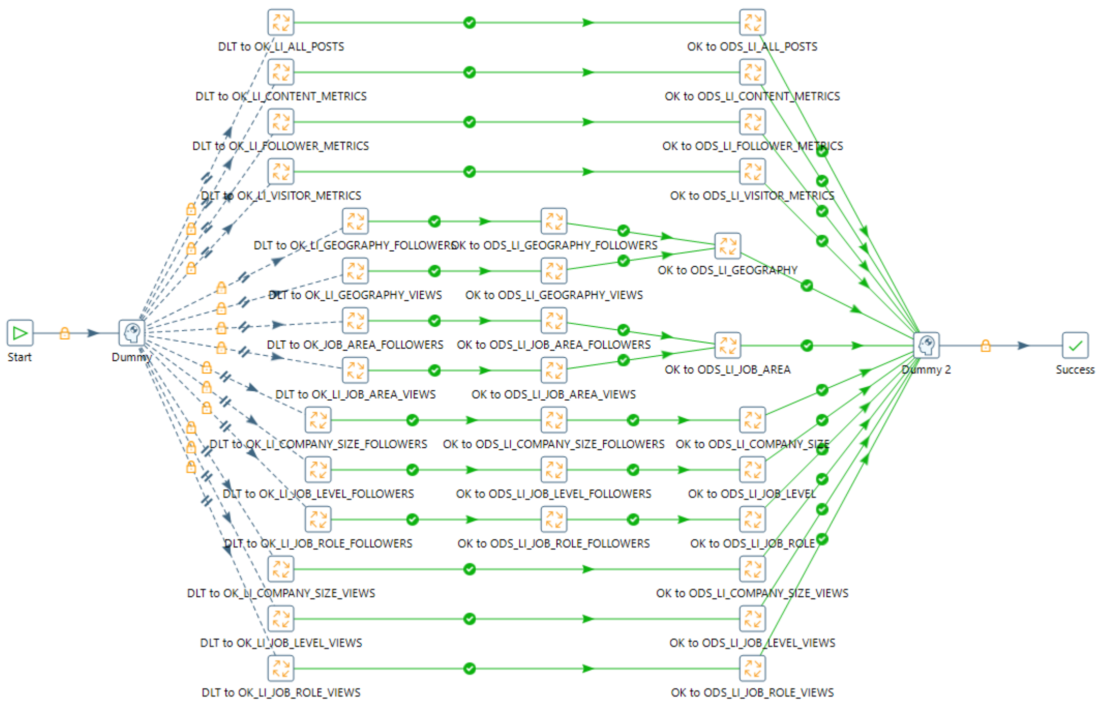
- Tables that enclose metrics with fine granularity that have been extracted from the principal sheet, which are replicated to the OK level after replacing NULL values and performing basic data type checks (e.g. integers are positive and dates exist) and merged into the ODS. Three tables in the L1 layer are populated in this case:
  - `ODS_LI_CONTENT_METRICS`
  - `ODS_LI_FOLLOWER_METRICS`
  - `ODS_LI_VISITOR_METRICS`
- Tables with aggregated metrics for a single dimension follow a standard pipeline from DLT to OK tables and from OK tables to ODS, while the dimension is extracted once the records have been stored in the L1 layer and it is immediately placed in its respective ODS table. Five dimension tables are populated in this scenario:
  - `ODS_LI_COMPANY_SIZE`
  - `ODS_LI_JOB_LEVEL`
  - `ODS_LI_JOB_ROLE`

  - `ODS_LI_JOB_AREA`
  - `ODS_LI_GEOGRAPHY`

  Since followers and visitor metrics tables share the same structure except for the metric they store, ODS tables records are either gathered from a single source, if the dimension descriptive field has a limited set of values, or from both tables when the set cardinality is high and cannot be anticipated beforehand. The firs three tables of the aforementioned list belong to the former scenario, while the others fall into the latter.

- There is a single dimension table at this stage directly obtained from LinkedIn, DLT_LI_ALL_POSTS, that encompasses content information and aggregated metrics. Since no other level of granularity is available, these metrics are kept in the dimension ODS table rather than a separate fact table.

Since there are dependencies among LinkedIn data quality mappings, `DLT_to_OK` jobs can be fully parallelized, whereas the execution order of `OK_to_ODS` jobs is illustrated in Figure 5.8.



Start
Dummy
DLT to OK_LI_ALL_POSTS
DLT to OK_LI_CONTENT_METRICS
DLT to OK_LI_FOLLOWER_METRICS
DLT to OK_LI_VISITOR_METRICS
DLT to OK_LI_GEOGRAPHY_FOLLOWERS
DLT to OK_LI_GEOGRAPHY_VIEWS
DLT to OK_JOB_AREA_FOLLOWERS
DLT to OK_LI_JOB_AREA_VIEWS
DLT to OK_LI_COMPANY_SIZE_FOLLOWERS
DLT to OK_LI_JOB_LEVEL_FOLLOWERS
DLT to OK_LI_JOB_ROLE_FOLLOWERS
DLT to OK_LI_COMPANY_SIZE_VIEWS
DLT to OK_LI_JOB_LEVEL_VIEWS
DLT to OK_LI_JOB_ROLE_VIEWS
OK to ODS_LI_GEOGRAPHY_FOLLOWERS
OK to ODS_LI_GEOGRAPHY_VIEWS
OK to ODS_LI_JOB_AREA_FOLLOWERS
OK to ODS_LI_JOB_AREA_VIEWS
OK to ODS_LI_COMPANY_SIZE_FOLLOWERS
OK to ODS_LI_JOB_LEVEL_FOLLOWERS
OK to ODS_LI_JOB_ROLE_FOLLOWERS
OK to ODS_LI_ALL_POSTS
OK to ODS_LI_CONTENT_METRICS
OK to ODS_LI_FOLLOWER_METRICS
OK to ODS_LI_VISITOR_METRICS
OK to ODS_LI_GEOGRAPHY
OK to ODS_LI_JOB_AREA
OK to ODS_LI_COMPANY_SIZE
OK to ODS_LI_JOB_LEVEL
OK to ODS_LI_JOB_ROLE
OK to ODS_LI_COMPANY_SIZE_VIEWS
OK to ODS_LI_JOB_LEVEL_VIEWS
OK to ODS_LI_JOB_ROLE_VIEWS
Dummy 2
Success

Figure 5.8: A comprehensive representation of the L1 mappings related to LinkedIn source tables.

5.5 Data Integration and enrichment

Data Integration is the ETL step in which information from different sources is gathered and prepared for the visualization layer. ODS tables provide a reconciled version of the transactional records which feed the subsequent MDM stage, where the actual integration is carried out, and the OUT stage, where data is prepared accordingly to the Data Warehouse model.

The main difference between the company framework and the employed pipeline is in the presence of MDM tables. Although two sources are ingested by the ETL application, the metrics they serve are different enough to lean towards a multiple fact table approach: at the end of the process, all the YouTube metrics will be collected into the table `FACT_YT_METRICS`, while the LinkedIn branch of the pipeline will feed the table `FACT_LI_METRICS`. Due to the branching of the flow into two separate data streams, MDM tables are not required and surrogate keys are generated in the ODS tables. Being in a SQL Server environment, surrogate keys are assigned an *IDENTITY* constraint, thus leaving their management entirely to the underlying DBMS.

The absence of MDM tables raises the issue of data enriching: whereas the framework places Machine Learning operations in the MDM mappings, a different architecture with Advanced Analytics at the end of the pipeline has been chosen in order to provide a more comprehensive analysis, following the solution proposed by the Google Cloud best practices.

All the ODS tables are mapped to the OUT tables according to the model provided in Chapter 4.

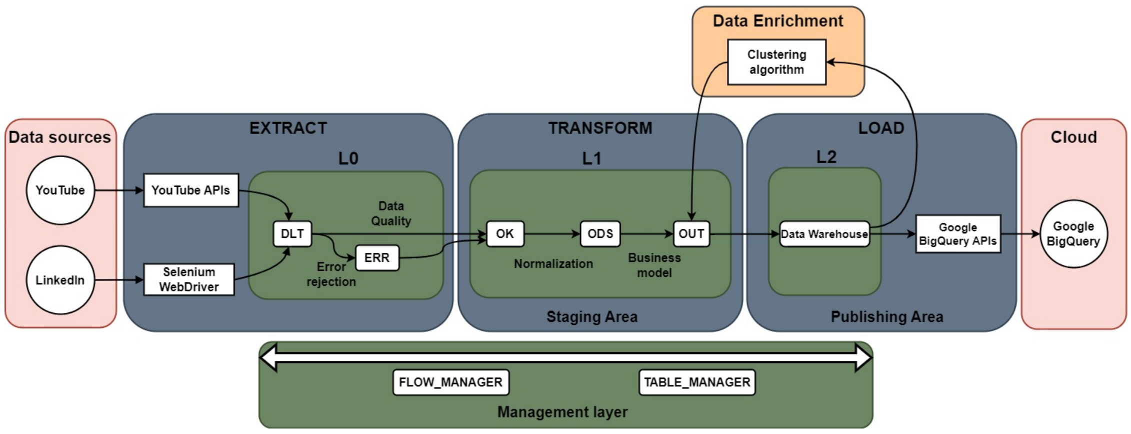
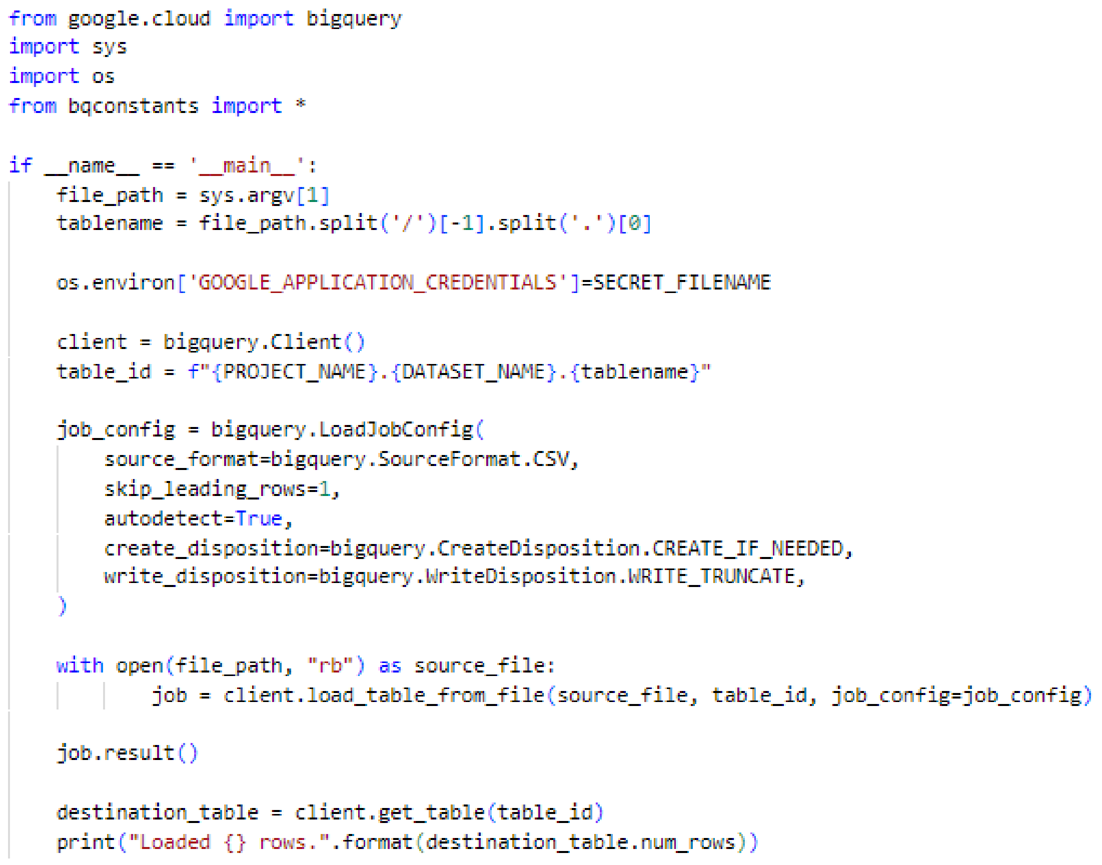


Figure 5.9: Full ETL pipeline architecture. Machine Learning is performed after the Data Warehouse level, before visualization.

5.6 Loading into Google BigQuery

The last step of the pipeline is replicating the delta of data from the local Data Warehouse to Google BigQuery. In order to provide the same output to both targets, the L2 jobs retrieve the delta of data from the OUT tables and fork the flow in two directions:

- Data is written locally in the L2 tables
- A temporary CSV file is created for each table; then the Python script `bqload.py` is run for each of the staging file. The script is responsible of invoking the Google BigQuery APIs to ingest the cloud Data Warehouse with new data. A code sample of the script can be found in Figure 5.10.



```
from google.cloud import bigquery
import sys
import os
from bqconstants import *

if __name__ == '__main__':
    file_path = sys.argv[1]
    tablename = file_path.split('/')[-1].split('.')[0]

    os.environ['GOOGLE_APPLICATION_CREDENTIALS']=SECRET_FILENAME

    client = bigquery.Client()
    table_id = f"{PROJECT_NAME}.{DATASET_NAME}.{tablename}"

    job_config = bigquery.LoadJobConfig(
        source_format=bigquery.SourceFormat.CSV,
        skip_leading_rows=1,
        autodetect=True,
        create_disposition=bigquery.CreateDisposition.CREATE_IF_NEEDED,
        write_disposition=bigquery.WriteDisposition.WRITE_TRUNCATE,
    )

    with open(file_path, "rb") as source_file:
            job = client.load_table_from_file(source_file, table_id, job_config=job_config)

    job.result()

    destination_table = client.get_table(table_id)
    print("Loaded {} rows.".format(destination_table.num_rows))
```

Figure 5.10: Sample code for the script *bqload.py*.

Chapter 6

Machine Learning and Advanced Analytics

The first goal of the thesis is automating the ingestion and the integration of social media reporting data, which is carried out by an ETL pipeline on an hybrid cloud solution. The second aim of the work resides in a deeper study of the current state of the collected data with Machine Learning techniques in order to provide an enhancement in the customer's analysis.

More specifically, the analysis focus dwells in three key points:

1. visualizing the current dataset for performing a qualitative research on the relevant characteristics of the dataset

2. looking for characterizing features which are decisive in partitioning the dataset into classes, leveraging Unsupervised Learning techniques

3. setting up a Supervised Learning model which allows predicting the classes of unseen data depending on the previous knowledge

In order to preserve sensitive information, noise is added to data elements and all the measurement units are omitted. Despite not supplying the actual magnitude of the phenomena may impede a clear dissertation, the patterns will have a central role in the following analysis.

6.1 Dataset overview

The preliminary step of the analysis is the visualization of the current state of the dataset in order to determine whether some inference can be provided before applying computationally expensive and time-consuming techniques, which are sensitive to noise and redundant data. Reducing a Machine Learning model complexity makes it less prone to overfitting, moreover a balanced workload is beneficial for its performance.

The dataset is composed of two subsets, one for each fact table previously populated. All the metrics have been modeled as non-negative additive values, while the dimensions are defined as categorical attributes represented by integers (the surrogate keys). In the Data Warehouse design phase, some fields have been added for legacy purposes, although their content is absent: before visualizing the feature distribution, a targeted query can provide this piece of information. For research purposes, only YouTube data is reported.

The table `FACT_YT_METRICS` has 5 categorical features, which are the modeled dimensions, and 24 numerical attributes. The following SQL query is executed for retrieving a summary about the usage of the latter:

```
SELECT
  SUM(VIEWS) AS TOT_VIEWS
, SUM(RED_VIEWS) AS TOT_RED_VIEWS
, SUM(ESTIMATED_MINUTES_WATCHED) AS TOT_ESTIMATED_MIN_WATCHED
, SUM(ESTIMATED_RED_MINUTES_WATCHED) AS TOT_ESTIMATED_RED_MIN_WATCHED
, SUM(LIKES) AS TOT_LIKES
, SUM(DISLIKES) AS TOT_DISLIKES
, SUM(VIDEOS_ADDED_TO_PLAYLIST) AS TOT_PLAYLIST_ADD
, SUM(VIDEOS_REMOVED_FROM_PLAYLIST) AS TOT_PLAYLIST_REMOVE
, SUM(SHARES) AS TOT_SHARES
, AVG(AVERAGE_VIEW_DURATION) AS AVG_AVERAGE_VIEW_DURATION
, AVG(AVERAGE_VIEW_PERCENTAGE) AS AVG_AVERAGE_VIEW_PERCENTAGE
, AVG(ANNOTATION_CTR) AS AVG_ANNOT_CTR
, AVG(ANNOTATION_CLOSE_RATE) AS AVG_ANNOT_CR
, SUM(ANNOTATION_IMPRESSIONS) AS TOT_ANNOT_IMPRESSIONS
, SUM(ANNOTATION_CLICKABLE_IMPRESSIONS) AS TOT_ANNOT_CLICK_IMPR
, SUM(ANNOTATION_CLOSABLE_IMPRESSIONS) AS TOT_ANNOT_CLOSE_IMPR
, SUM(ANNOTATION_CLICKS) AS TOT_ANNOT_CLICKS
, SUM(ANNOTATION_CLOSES) AS TOT_ANNOT_CLOSES
, AVG(CARD_CLICK_RATE) AS AVG_CARD_CR
, AVG(CARD_TEASER_CLICK_RATE) AS AVG_CARD_TEASER_CR
, SUM(CARD_IMPRESSIONS) AS TOT_CARD_IMPR
, SUM(CARD_TEASER_IMPRESSIONS) AS TOT_CARD_TEASER_IMPR
, SUM(CARD_CLICKS) AS TOT_CARD_CLICKS
, SUM(CARD_TEASER_CLICKS) AS TOT_CARD_TEASER_CLICKS
FROM VGMARKETING.L2.FACT_YT_METRICS;
```

As expected, the result of the query shows that all the fields related to the YouTube annotations and informative cards functions are not used, thereby limiting the analysis to 11 features, listed in Table 6.1 with the average value for each field retrieved by the query.

Field name	**Query output**	**Aggregation**
`VIEWS`	151470	SUM
`RED_VIEWS`	418	SUM
`ESTIMATED_MINUTES_WATCHED`	239338	SUM
`ESTIMATED_RED_MINUTES_WATCHED`	980	SUM
`LIKES`	790	SUM
`DISLIKES`	33	SUM
`VIDEOS_ADDED_TO_PLAYLIST`	341	SUM
`VIDEOS_REMOVED_FROM_PLAYLIST`	55	SUM
`SHARES`	1452	SUM
`AVERAGE_VIEW_DURATION`	49	AVG
`AVERAGE_VIEW_PERCENTAGE`[1]	13.65	AVG

Table 6.1: List of selected features for FACT_YT_METRICS, with relative query output.

Distributions of the selected features are reported in Figure 6.2. For better representation, some features like `VIEWS`, `ESTIMATED_MINUTES_WATCHED`, and `AVERAGE_VIEW_PERCENTAGE` are reported in a logarithmic scale. Here some considerations on the distributions:

- The feature `VIEWS` resembles a Gaussian distribution, with the presence of many outliers that extend significantly the range scale. This feature would benefit from Z-score normalization [48].
- `AVERAGE_VIEW_DURATION` and `AVERAGE_VIEW_PERCENTAGE` are shaped like a mixture of Gaussian distributions in which at least two clusters can be identified with a qualitative approach. Thereby, their features can supply the most prominent contribute to the cluster partitioning.
- Considerable outliers can be identified in `ESTIMATED_RED_MINUTES_WATCHED` and `SHARES`, leading to noise during clustering operations.
- Features related to user's interaction such as `LIKES` and `DISLIKES`, and playlist features (`VIDEOS_ADDED_TO_PLAYLIST` and `VIDEOS_REMOVED_FROM_PLAYLIST`) are fairly evenly distributed in a limited support; further pre-processing may be required in order to provide more efficient Supervised Learning models for classification.

[1]The average view percentage is calculated by YouTube APIs as *estimatedMinutesWatched/videoDuration*. Watch time can be higher than the video duration, since the user can navigate freely in the video during a single session (its domain is not $[0, 100]$, but rather $\mathbb{R}_0^+$).

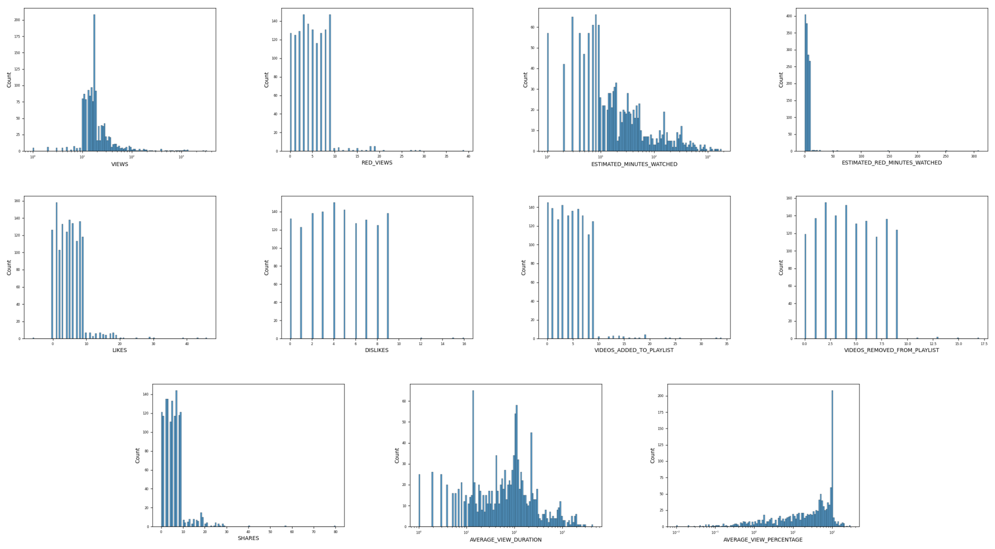


Figure 6.1: Features distribution for the relevant metrics in YouTube data.

As shown in the correlation heatmap in Figure 6.2, features are generally weakly correlated with the exception of:

- `ESTIMATED_MINUTES_WATCHED` and `AVERAGE_VIEW_DURATION`, which are semantically similar
- `AVERAGE_VIEW_DURATION` and `AVERAGE_VIEW_PERCENTAGE`, since their meanings are closely related to each other (an increasing average view duration reflects in an higher view percentage)

Since no relevant correlation between features can be extracted from the correlation matrix, data is ingested in clustering algorithms without further processing.

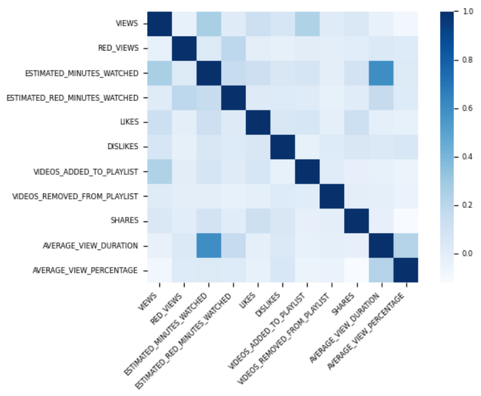


Figure 6.2: Correlation heatmap for the selected features of YouTube metrics

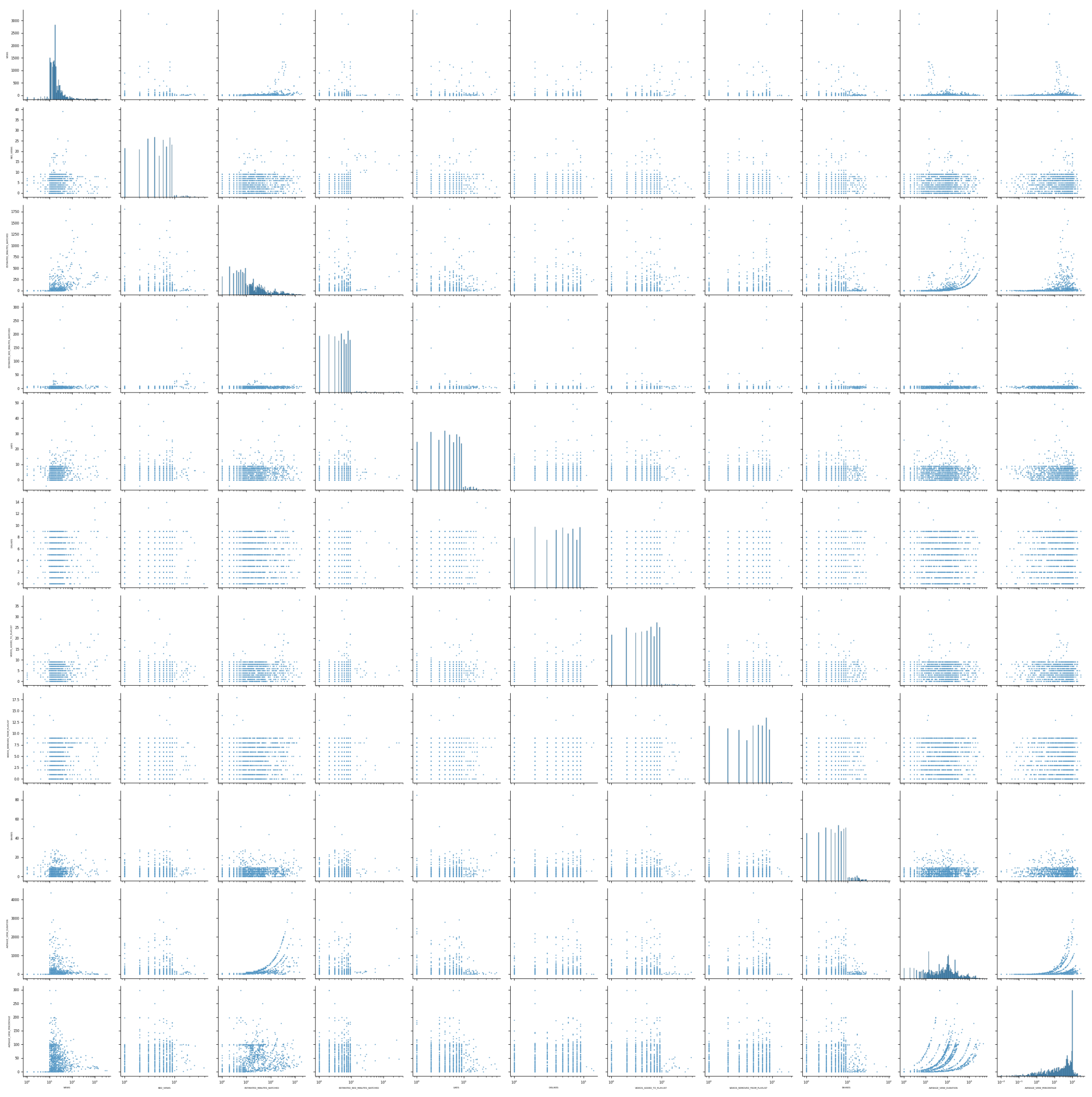


Figure 6.3: Pair-wise distributions and histograms for the selected features. All the axes use logarithmic scales.

6.2 Clustering analysis

Two diverse clustering techniques are applied and compared with different hyperparameters: *K-Means* and *DBSCAN*. From a visual preliminary analysis of Figure 6.3, the most promising results are expected from density-based algorithms such as the latter. Moreover, unlike K-Means, DBSCAN does not require to fix the number of clusters in advance, being more robust to noise and outliers. Conversely, all the DBSCAN clusters are formed using the same hyperparameters, resulting in an inefficient partition for clusters with different densities.

6.2.1 K-Means clustering

Recalling the definition of the K-Means clustering algorithm objective function:

$$J = \arg\min_{C} \sum_{i=1}^{K} \sum_{x \in \mathcal{C}_i} \|\mathbf{x} - \mu_{\mathbf{i}}\|^2, \qquad \mu_i = \frac{1}{|C_i|} \sum_{x \in \mathcal{C}_i} \mathbf{x}$$

where μ_i is the centroid of the cluster C_i.

Since no ground truth is available through labels on training data, clustering performance is tracked by employing the *mean Silhouette Coefficient* (SC) evaluation metric [49, 50]:

$$SC = \max \frac{1}{N} \sum_{i}^{N} \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where $a(i)$ is the average distance of the i-th point from the other points of the cluster, while $b(i)$ is the average distance of the same point from the points assigned to the nearest cluster. In Table 6.2 silhouette values for K-Means are reported in relation to the hyperparameter of the algorithm, which is the number of clusters. A visual representation of the SC values is presented in Figure 6.4.

#Clusters	SC
2	0.8242
3	0.7675
4	0.7835
5	0.7866
6	0.5441
7	0.5591
8	0.5452
9	0.5447
10	0.4451
11	0.4340
12	0.4354
13	0.4523
14	0.4582
15	0.4503
16	0.4632
17	0.4665
18	0.4920
19	0.4823
20	0.4913
21	0.4930
22	0.4540
23	0.4431
24	0.4477
25	0.4501
26	0.4498
27	0.4472
28	0.4440
29	0.4434

Table 6.2: Silhouette Coefficients for K-Means clustering with $k = 2, 3, \ldots, 28$

The three most effective clustering are obtained with the number of clusters $k \in \{2,4,5\}$. However, the most suited hyperparameter will be determined after comparing the results with DBSCAN clustering analysis obtained in Section 6.2.2.

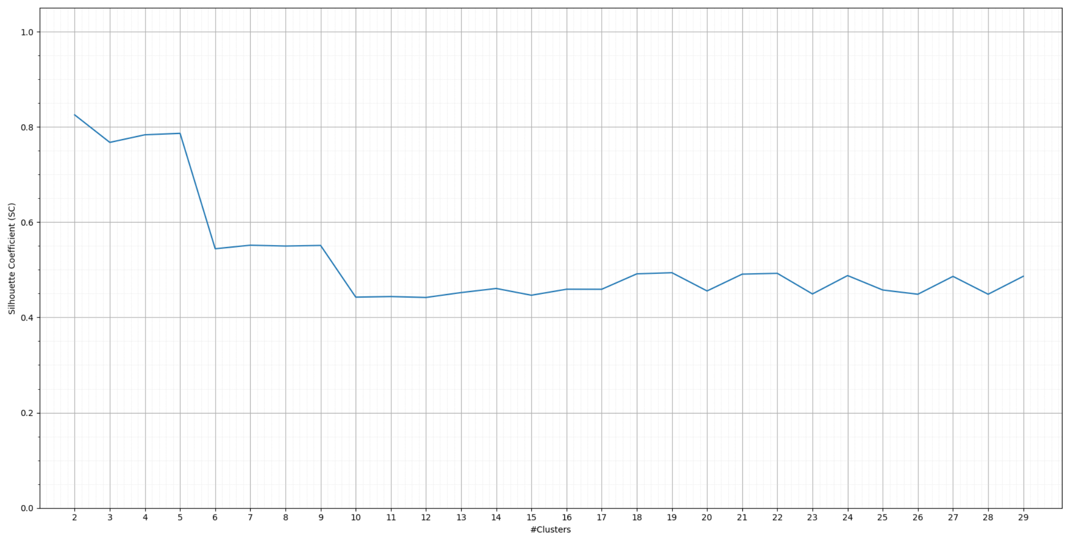


Figure 6.4: Line plot for Silhouette Coefficient values obtained for K-Means hyperparameter tuning performed on YouTube data.

6.2.2 DBSCAN clustering

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm requires the tuning of two hyperparameters:

- $minPts$, the number of points required to form a core point
- $\epsilon(eps)$, the distance in which two points are considered neighbors

The analysis is carried out by performing a grid search on ϵ and $minPts$ on the Z-normalized dataset made of n samples. Values for ϵ have been selected on a logarithmic scale between 10^{-2} and 10^{3} for a first rough search, while $minPts$ is chosen so that $minPts \in \{2,4,8,16\}$. A finer tuning is then performed with a linear scale of ϵ in the most promising interval. The mean Silhouette Coefficient is chosen as main evaluation metric for a direct comparison with the results of K-Means clustering. However, the number of resulting clusters K, the number of noise points N_{np}, and the outlier ratio $\frac{N_{np}}{n}$ are taken into account, in order to prevent the inclusion of too many outliers.

The most notable results for the second tuning phase are reported in Table 6.3 and Figure 6.5, for which $\epsilon \in [1, 10]$.

An increasing value of ϵ generally results in fewer, larger clusters that include the majority of the points. Higher $minPts$ settings bring to the same results for smaller ϵ ranges, converging to a binary partition rather earlier than lower values of $minPts$. For $minPts = 16$ the Silhouette Coefficient can be computed only

minPts	ϵ	**K**	$\mathbf{N_{np}}$	**SC**	**Outlier ratio (%)**
2	3.4	4	27	0.4353	2.0
2	7.6	2	3	0.7317	0.2
4	3.5	2	29	0.5332	2.1
8	2.6	2	74	0.4265	5.5
16	1.3	2	1079	-0.1734	80.0

Table 6.3: Most notable results for grid search on DBSCAN for Z-Score normalized YouTube data. The noise cluster is excluded from the count.

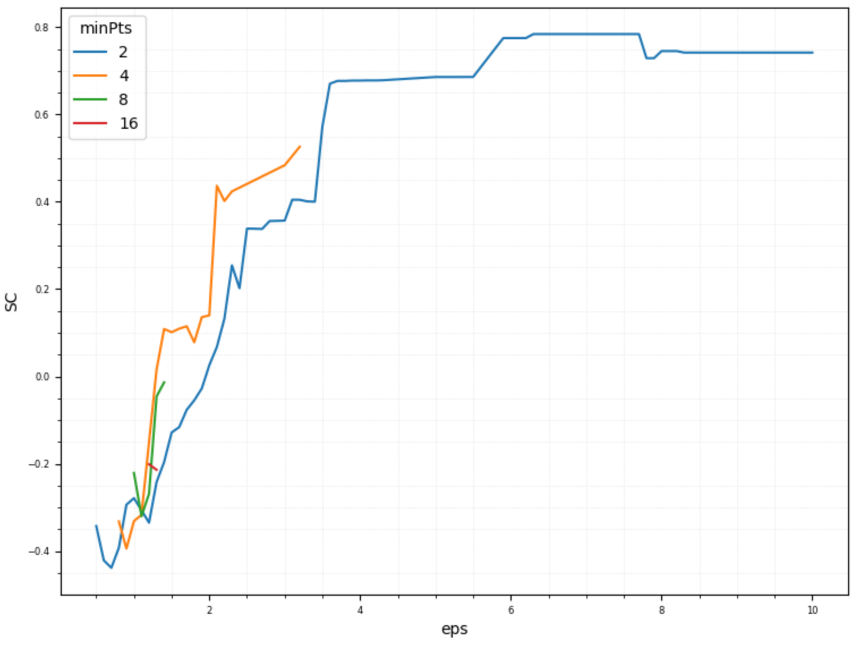


Figure 6.5: Silhouette plot of DBSCAN grid search on Z-Score normalized YouTube data.

for a small subset of values of ϵ due to numerical issues, thus being discarded from the analysis. Furthermore, a noteworthy insight can be provided for this hyperparameters setting, where the SC drops to negative values with higher ϵ and presents a noise points ratio of 80.0%. The setting ($\epsilon = 1.3, minPts = 8$) undergoes a similar inference, where the outlier ratio is considerably lower, but presenting a SC close to 0 nevertheless.

For $minPts = 2$ the SC reaches its plateau at $\epsilon \approx 6.0$, while higher minimum core points settings converge at $\epsilon \approx 2$, despite having a lower SC. The outlier ratio grows accordingly with the value of $minPts$, meaning that requiring denser clusters involves more noise points.

Whereas the highest values of SC are reached for ϵ, the inclusion of many noise points discourages from selecting these settings. Moreover, the only hyperparameters allowing a multi-class partitions are ($\epsilon = 3.4, minPts = 2$), with an underperforming SC. The most promising hyperparameters for the addressed tasks are therefore:

- **Binary clustering ($K = 2$).** $\epsilon = 7.6$, $minPts = 2$ with $SC = 0.7317$ and 0.2% outliers
- **Multi-class clustering ($K = 4$).** $\epsilon = 3.4$, $minPts = 2$ with $SC = 0.4353$ and 2.0% outliers

6.2.3 Unsupervised model selection

The binary clustered dataset for K-Means and DBSCAN algorithms is respectively illustrated in Figure 6.7 and Figure 6.8. While the former is a straightforward partition of the dataset, the latter is, in reality, a partition in three groups, with the third one being the outliers cluster. While reaching a high value of $SC = 0.7317$, DBSCAN is still inferior to K-Means in binary clustering due to its unbalancing in the cardinality of the two partitions.

In Figure 6.9 and Figure 6.10 a visual representation of the clusters is provided for multi-class clustering respectively for K-Means and DBSCAN. Unlike the expectations, DBSCAN has been proved inefficient in providing evenly shaped clusters: the most prominent cluster includes the majority of data points, relegating the elements forming other clusters to a role similar to outliers. Furthermore, the DBSCAN evaluation metric indicates a significantly lower performance than its K-Means counterpart.

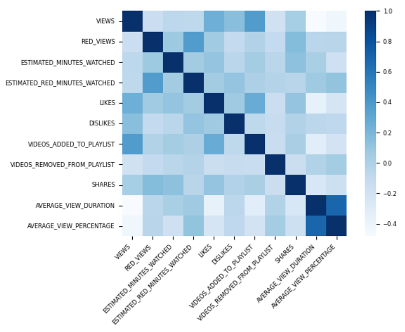
In both cases, the final choice is the partition with K-Means.

6.2.4 Cluster labeling

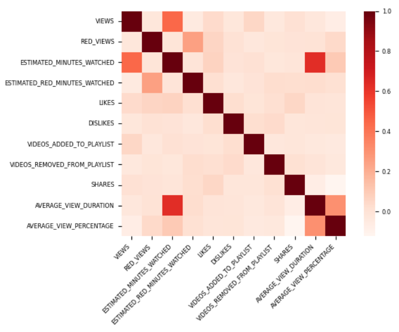
Two kinds of analyses can be performed starting from the results stated in Section 6.2.3, which are here reported with more in depth insights.

Users behavior analysis

In this scenario, a label between $\mathcal{H}_0$ and $\mathcal{H}_1$ is assigned to the data points in order to discriminate them with a binary classification task. The selected model is the clustered version of the Z-Score normalized dataset with $K = 2$.



(a) Label L=$\mathcal{H}_0$



(b) Label L=$\mathcal{H}_1$

Figure 6.6: Per-class Pearson Correlation heatmap on clustered YouTube data.

By a qualitative visual analysis of the feature plots in Figure 6.8 and the Pearson correlation heatmaps in Figure 6.6, we can infer:

- The most discriminant feature is `ESTIMATED_MINUTES_WATCHED`, whose samples have slightly overlapping per-class distributions that can be modeled as Gaussian distributions. Since the analysis is binary, it is safe to assume that the classes refer to the users' engagement: an higher watch time results in a more interested viewer.

- Per-class correlation heatmaps confirms the high correlation between the metrics `AVERAGE_VIEW_DURATION` and `AVERAGE_VIEW_PERCENTAGE`.

- The correlation matrix for the records assigned to cluster $\mathcal{H}_0$ illustrates a significant correlation among the majority of the features, entailing the potential need of further processing. The qualitative interpretation of these relationships is that the number of views for short times is strictly related to the playlist usage. The *watch later* functionality, which allows users to keep the interesting videos in a single location for watching them in later times, appears to be crucial to define "casual users", who have a low watch time and do not interact much with the platform.

- By analyzing the correlation between `VIEWS` and `ESTIMATED_MINUTES_WATCHED` in cluster $\mathcal{H}_1$, we can assume that the cluster represents a small subset of "loyal users", who spend much more time watching videos and tend to complete the vision more than casual users.

In conclusion, we can state that the two clusters properly approximate the behavior of *casual users* and *loyal users*, relatively.

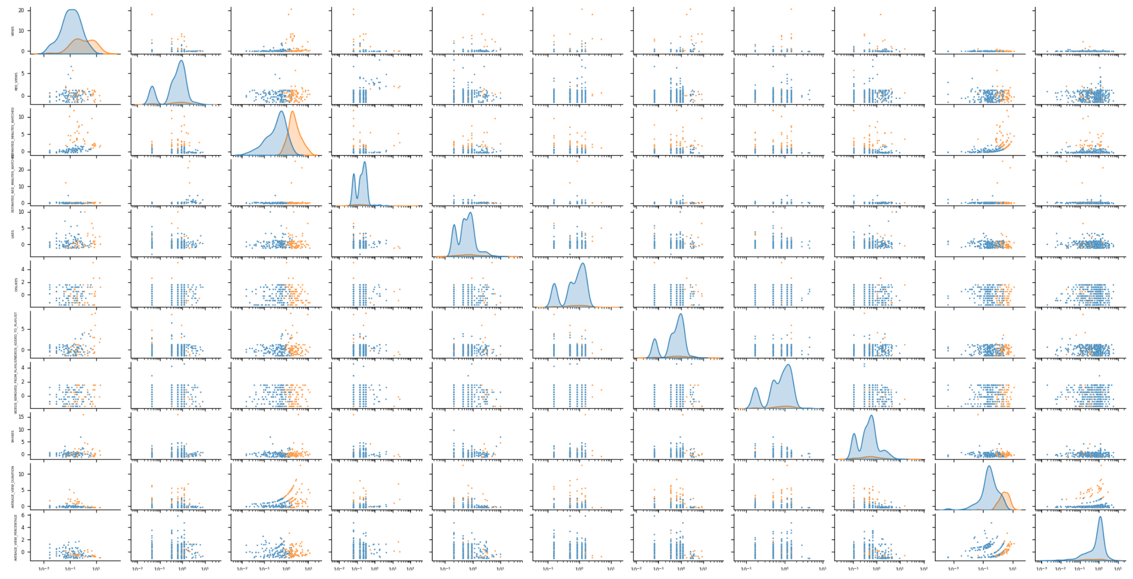


Figure 6.7: Labeled features distribution for K-Means clustering on YouTube data with $K = 2$.

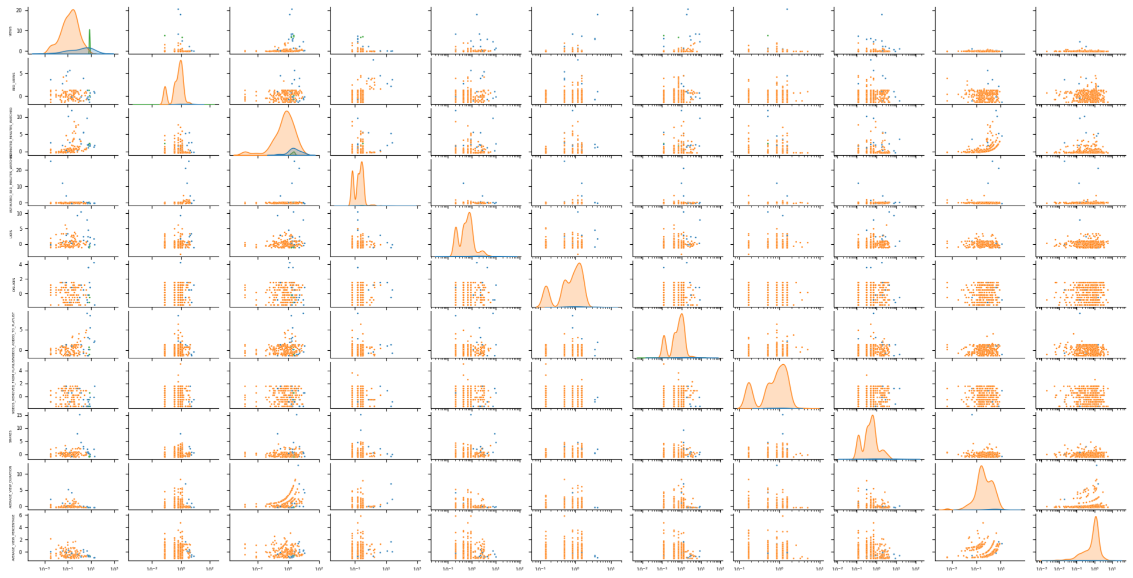


Figure 6.8: Clustered features distribution for DBSCAN on YouTube data with $(\epsilon = 7.6, minPts = 2)$, $K = 2$.

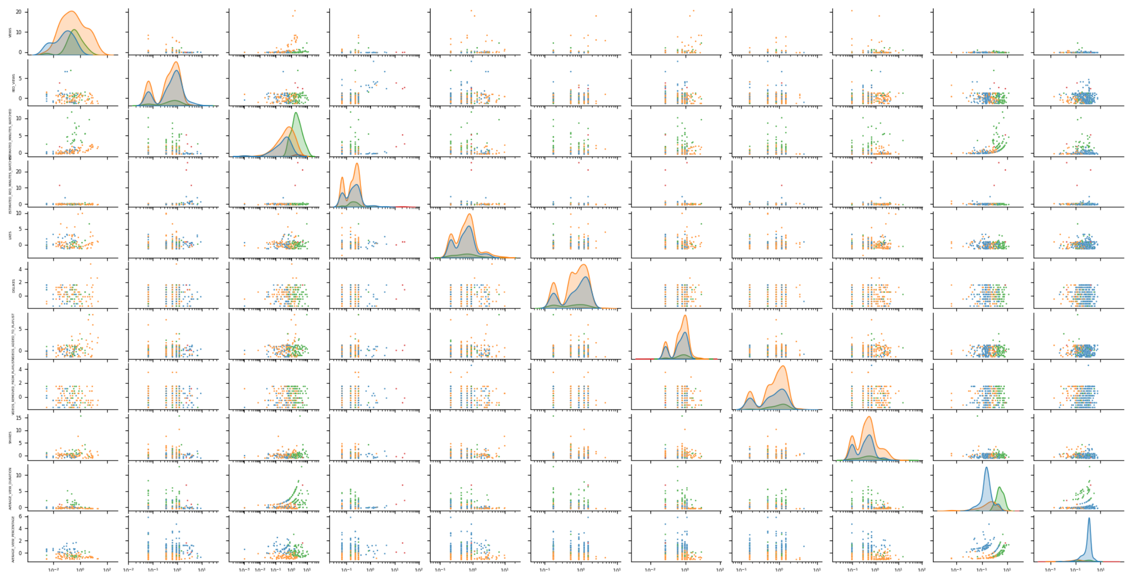


Figure 6.9: Labeled features distribution for K-Means clustering on YouTube data with $K = 4$.

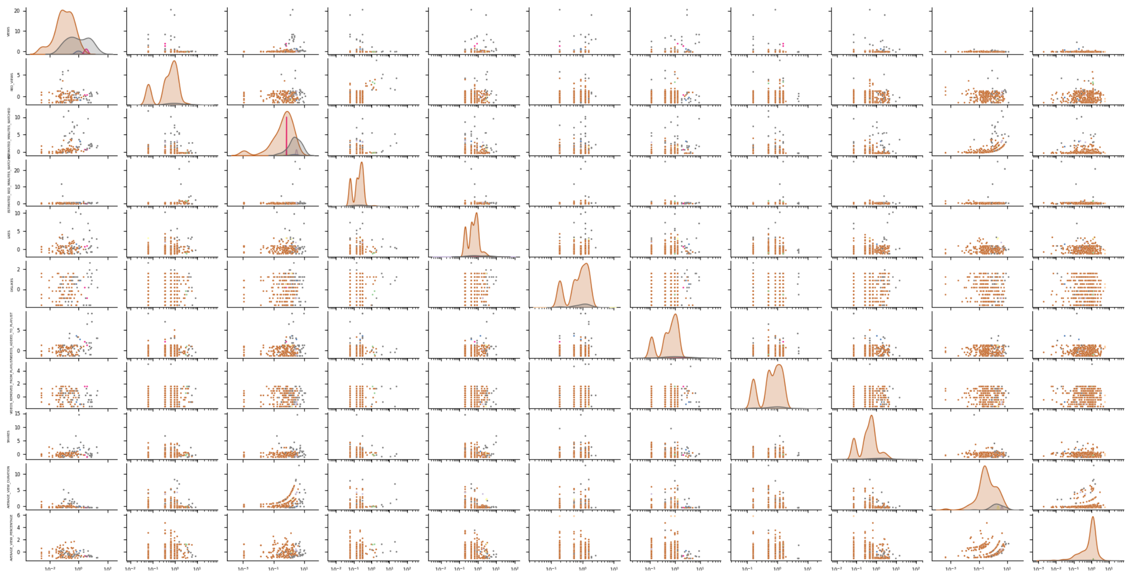


Figure 6.10: Clustered features distribution for DBSCAN on YouTube data with $(\epsilon = 3.4, minPts = 2)$, $K = 4$.

Chapter 7

Conclusion and next steps

The work supplies an automated ETL pipeline for integrating YouTube and LinkedIn engagement metrics for a customer of the company Mediamente Consulting, while performing Data Enrichment tasks with the aid of Unsupervised Learning techniques. The main aim is satisfying the customer requirements to gather the diverse sources into a centralized architecture for Data Visualization and further Machine Learning analysis. This thesis covers in depth the ETL pipeline and the design of the Data Warehouse. Furthermore, preliminary insights for the Data Enrichment step are provided through the implementation of clustering models.

The ingestion step involves the acquisition of the reporting metrics provided by two social media platforms, YouTube and LinkedIn. For what concerns the former, ingestion is carried out by implementing a Python script that leverages YouTube Data API and YouTube Analytics API services in order to perform a daily incremental ingestion. A temporary solution for the automation of LinkedIn metrics retrieval is given by the module Selenium WebDriver: a couple of Python scripts is called along with the first one for downloading the reports from the LinkedIn website and feeding them in a compatible format with the next stages of the pipeline.

The second phase is the proper ETL pipeline, where data is cleansed, integrated and prepared for storage and visualization on a Data Warehouse with a multidimensional model. The ingested records are stored in temporary tables, where they undergo the planned transformations, which derive from a custom version of the company framework. The ETL pipeline is implemented with the open-source tool Pentaho Data Integration. At the end of the stage, data is ready to be stored in a Data Warehouse. Google BigQuery has been appointed by the customer as Data Warehouse solution, thus the Load step is performed at the end of the pipeline with the execution of a dedicated Python script, leveraging Google BigQuery APIs.

The last step is the dataset analysis performed with Unsupervised Learning techniques. Clustering tasks are designed in order to search for meaningful groups of data depending on similarities among the available metrics. Two algorithms are chosen for this operation: K-Means and DBSCAN. Both models are evaluated by the Silhouette Coefficient metric in various settings, in order to select a binary partition model and a multi-class one. Relevant inference is drawn from the clustered data and labels are at last assigned to the records, thus performing the Enrichment step.

The performed work has been carried out by following the principles and best practices of the company workflow, while exploiting the knowledge of Google Cloud Platform resources acquired during the internship that preceded the thesis work.

7.1 Possible future works

The designed ETL pipeline is currently fully operational. Nevertheless, further improvements can be introduced in the ingestion, enrichment, and visualization steps.

- The ingestion of LinkedIn reports requires to be completely replaced by the implementation of LinkedIn APIs when their access is provided by the platform. This enhancement will impact positively on the performance of the application and will be fully compliant to the original requirement.
- Despite the adequate results obtained during the Clustering step, the presence of an high level of noise and the limited size of the dataset available at design time have crucially affected the analysis. Some alternative clustering algorithms are Bisecting K-Means, BIRCH (Hierarchical Clustering), OPTICS (a DBSCAN variant), Gaussian Mixture Models, and Spectral Clustering. Further variations can be applied to the distance metric used in DBSCAN. Moreover, further evaluation metrics can be defined in order to perform a better hyperparameter tuning.
- The actual realization of a Supervised Learning model based on the clustering labels is the next step in the project. Binary or multi-class classifier models can be implemented through Logistic Regression models, Gaussian Mixture Model-based classifiers or Support Vector Machines. Other solutions can be found in Deep Learning models, such as Multilayer Perceptron (MLP) models for classification. Recurrent Neural Networks can also be employed for a time series analysis.
- An actual enrichment step can be implemented by training one of the Supervised Learning models discussed above before loading the enhanced records

into the Data Warehouse and formulating inference on newer records, which lack a label when supplied to the pipeline.

- The company plans to fully migrate the application to Google Cloud Platform. Other cloud Data Integration tools, such as Apache Spark, Google Dataflow, or its fully-managed version Apache Beam can be leveraged to lift the ETL process from the company server and scale up the application.

Appendix A

Data Warehouse tables

The following tables describe the Data Warehouse model in its entirety, including technical fields.

Common tables

Here is a comprehensive list of the tables related to both YouTube and LinkedIn fact tables.

DIM_CALENDAR

The DIM_CALENDAR table has no technical fields since its records have been generated statically. The chosen date range goes from *2000-01-01* to *2030-12-31*.

Field name	**Data type**	**Primary key**	Nullable
`DATE_SK`	int	Yes	No
`ISO_DATE`	int	No	No
`DAY_OF_WEEK`	int	No	No
`DAY_OF_MONTH`	int	No	No
`MONTH`	int	No	No
`YEAR`	int	No	No
`WEEK_OF_YEAR`	int	No	No

Table A.1: Full description for DIM_CALENDAR table.

DIM_DEVICE_TYPE

Field name	**Data type**	**Primary key**	**Nullable**
`DEVICE_TYPE_SK`	int	Yes	No
`DEVICE_DESC`	varchar	No	No
`HIERARCHY_DESC`	varchar	No	No
`INS_TIME`	datetime	No	No
`UPD_TIME`	datetime	No	No
`JOBID_INS`	numeric(14,0)	No	No
`JOBID_UPD`	numeric(14,0)	No	No

Table A.2: Full description for DIM_DEVICE_TYPE table.

DIM_CAMPAIGN_LABEL

Field name	**Data type**	**Primary key**	**Nullable**
`CAMPAIGN_LABEL_SK`	int	Yes	No
`CAMPAIGN_LABEL_DESC`	varchar	No	No
`INS_TIME`	datetime	No	No
`UPD_TIME`	datetime	No	No
`JOBID_INS`	numeric(14,0)	No	No
`JOBID_UPD`	numeric(14,0)	No	No

Table A.3: Full description for DIM_CAMPAIGNtable.

YouTube tables

The following tables are specific to the YouTube portion of the Data Warehouse.

DIM_VIDEO

Field name	**Data type**	**Primary key**	**Nullable**
`VIDEO_SK`	int	Yes	No
`VIDEO_ID`	varchar	No	No
`CHANNEL_ID`	varchar	No	No
`CHANNEL_TITLE`	varchar	No	No
`VIDEO_DESCRPTION`	ntext	No	No
`LIVE_BROADCAST_CONTENT`	varchar	No	No
`PUBLISH_TIME`	datetime	No	No
`PUBLISHED_AT`	datetime	No	No
`VIDEO_TITLE`	nvarchar	No	No
`THUMBNAIL_URL`	varchar	No	No
`INS_TIME`	datetime	No	No
`UPD_TIME`	datetime	No	No
`JOBID_INS`	numeric(14,0)	No	No
`JOBID_UPD`	numeric(14,0)	No	No

Table A.4: Full description for DIM_VIDEO table.

DIM_SUBSCRIBED

Field name	**Data type**	**Primary key**	**Nullable**
`SUBSCRIBED_STATUS_SK`	int	Yes	No
`SUBSCRIBED_STATUS_DESC`	varchar	No	No
`INS_TIME`	datetime	No	No
`UPD_TIME`	datetime	No	No
`JOBID_INS`	numeric(14,0)	No	No
`JOBID_UPD`	numeric(14,0)	No	No

Table A.5: Full description for DIM_SUBSCRIBED table.

DIM_TRAFFIC_SOURCE

Field name	**Data type**	**Primary key**	**Nullable**
`TRAFFIC_SOURCE_SK`	int	Yes	No
`TRAFFIC_SOURCE_DESC`	varchar	No	No
`INS_TIME`	datetime	No	No
`UPD_TIME`	datetime	No	No
`JOBID_INS`	numeric(14,0)	No	No
`JOBID_UPD`	numeric(14,0)	No	No

Table A.6: Full description for DIM_DEVICE_TYPE table.

FACT_YT_METRICS

Field name	Data type	Primary key
DATE_SK	int	Yes
VIDEO_SK	int	Yes
SUBSCRIBED_STATUS_SK	int	Yes
DEVICE_TYPE_SK	int	Yes
TRAFFIC_SOURCE_SK	int	Yes
CAMPAIGN_LABEL_SK	int	Yes
VIEWS	int	No
RED_VIEWS	int	No
ESTIMATED_MINUTES_WATCHED	int	No
ESTIMATED_RED_MINUTES_WATCHED	int	No
LIKES	int	No
DISLIKES	int	No
VIDEOS_ADDED_TO_PLAYLIST	int	No
VIDEOS_REMOVED_FROM_PLAYLIST	int	No
SHARES	int	No
AVERAGE_VIEW_DURATION	int	No
AVERAGE_VIEW_PERCENTAGE	numeric(38,10)	No
ANNOTATION_CTR	numeric(38,10)	No
ANNOTATION_CLOSE_RATE	numeric(38,10)	No
ANNOTATION_IMPRESSIONS	int	No
ANNOTATION_CLICKABLE_IMPRESSIONS	int	No
ANNOTATION_CLOSABLE_IMPRESSIONS	int	No
ANNOTATION_CLICKS	int	No
ANNOTATION_CLOSES	int	No
CARD_CLICK_RATE	numeric(38,10)	No
CARD_TEASER_CLICK_RATE	numeric(38,10)	No
CARD_IMPRESSIONS	int	No
CARD_TEASER_IMPRESSIONS	int	No
CARD_CLICKS	int	No
CARD_TEASER_CLICKS	int	No
INS_TIME	datetime	No
UPD_TIME	datetime	No
JOBID_INS	numeric(14,0)	No
JOBID_UPD	numeric(14,0)	No

Table A.7: Full description for FACT_YT_METRICS model.

LinkedIn tables

Tables that model the LinkedIn metrics and dimensions are listed below.

DIM_LINKEDIN_PAGE

At design time three possible values for `LINKEDIN_PAGE_DESC` are available, to which an undefined value `"ND"` is added:

- `ND`
- `PANORAMICS`
- `COMPANY_LIFE`
- `JOB_OFFERING`

Since the table is defined at the L2 layer, it is used as lookup during the integration step of ETL.

Field name	**Data type**	**Primary key**	**Nullable**
`LINKEDIN_PAGE_SK`	int	Yes	No
`LINKEDIN_PAGE_DESC`	varchar	No	No
`INS_TIME`	datetime	No	No
`UPD_TIME`	datetime	No	No
`JOBID_INS`	numeric(14,0)	No	No
`JOBID_UPD`	numeric(14,0)	No	No

Table A.8: Full description for DIM_LINKEDIN_PAGE table.

FACT_LI_METRICS

The fact table which has been designed for LinkedIn metrics has four dimensions:

- `DATE_SK` from `DIM_CALENDAR`
- `DEVICE_TYPE_SK` from `DIM_DEVICE_TYPE`
- `LINKEDIN_PAGE_SK` from `DIM_LINKEDIN_PAGE`
- `CAMPAIGN_LABEL_SK` from `DIM_CAMPAIGN_LABEL`

Field name	**Data type**	**Primary key**	**Nullable**
`DATE_SK`	int	Yes	No
`DEVICE_TYPE_SK`	int	Yes	No
`LINKEDIN_PAGE_SK`	int	Yes	No
`CAMPAIGN_LABEL_SK`	int	Yes	No
`PAGE_VIEWS`	int	No	No
`PAGE_UNIQUE_VISITORS`	int	No	No
`ORGANIC_FOLLOWERS`	int	No	No
`SPONSORED_FOLLOWERS`	int	No	No
`ORGANIC_IMPRESSIONS`	int	No	No
`ORGANIC_UNIQUE_IMPRESSIONS`	int	No	No
`ORGANIC_CLICKS`	int	No	No
`ORGANIC_REACTIONS`	int	No	No
`ORGANIC_COMMENTS`	int	No	No
`ORGANIC_POST_DIFFUSIONS`	int	No	No
`ORGANIC_INTEREST_PERCENTAGE`	numeric(38,10)	No	No
`SPONSORED_IMPRESSIONS`	int	No	No
`SPONSORED_CLICKS`	int	No	No
`SPONSORED_REACTIONS`	int	No	No
`SPONSORED_COMMENTS`	int	No	No
`SPONSORED_POST_DIFFUSIONS`	int	No	No
`SPONSORED_INTEREST_PERCENTAGE`	numeric(38,10)	No	No
`INS_TIME`	datetime	No	No
`UPD_TIME`	datetime	No	No
`JOBID_INS`	numeric(14,0)	No	No
`JOBID_UPD`	numeric(14,0)	No	No

Table A.9: Full description for FACT_LI_METRICS table.

Bibliography

[1] Douglas Laney. *3D data Management: controlling data volume, velocity, and variety.* 2001. URL: http://blogs.gartner.com/douglaney/files/2012/01/ad949-3D-data-Management-Controlling-data-Volume-Velocity-and-Variety.pdf..

[2] Rob Kitchin and Gavin McArdle. «What makes Big Data, Big Data? Exploring the ontological characteristics of 26 datasets». In: *Big Data & Society* (2016). DOI: 10.1177/2053951716631130. eprint: https://doi.org/10.1177/2053951716631130. URL: https://doi.org/10.1177/2053951716631130.

[3] Fabio Duarte. *Amount of Data Created Daily (2023)*. Through Statista. Apr. 2023. URL: https://explodingtopics.com/blog/data-generated-per-day.

[4] Stacy Jo Dixon. *Media usage in an internet minute as of April 2022*. Through Statista. Sept. 13, 2023. URL: https://www.statista.com/statistics/195140/new-user-generated-content-uploaded-by-users-per-minute/.

[5] Bernard Marr. *Big Data: The 5 Vs Everyone Must Know*. Mar. 6, 2014. URL: https://www.linkedin.com/pulse/20140306073407-64875646-big-data-the-5-vs-everyone-must-know.

[6] James Kobielus. *Measuring the Business Value of Big Data*. May 9, 2013. URL: https://web.archive.org/web/20210128191754/https://www.ibmbigdatahub.com/blog/measuring-business-value-big-data.

[7] Vikash Ranjan. «A Comparative Study between ETL (Extract-Transform-Load) and ELT (Extract-Load-Transform) approach for loading data into a Data Warehouse By». In: 2009. URL: http://www.ecst.csuchico.edu/~juliano/csci693/Presentations/2009w/Materials/Ranjan/Ranjan.pdf.

[8] Antonio Greco. «E-Commerce monitoring solution for product allocation and marketing planning forecasting.» Master's Thesis. Politecnico di Torino, Apr. 2018. URL: http://webthesis.biblio.polito.it/id/eprint/7431.

[9] R. John Davenport. «Etl Vs Elt. A Subjective View». In: 2009. URL: https://api.semanticscholar.org/CorpusID:18694321.

[10] Gui-xian Zhou, Qing-sheng Xie, and Yao Hu. «E-LT Integration to Heterogeneous Data Information for SMEs Networking Based on E-HUB». In: *Fourth International Conference on Natural Computation, ICNC 2008, Jinan, Shandong, China, 18-20 October 2008, Volume 5*. Ed. by Maozu Guo, Liang Zhao, and Lipo Wang. IEEE Computer Society, 2008, pp. 212–216. DOI: 10.1109/ICNC.2008.77. URL: https://doi.org/10.1109/ICNC.2008.77.

[11] Jim Harris. *The growing importance of big data quality*. Nov. 21, 2016. URL: https://blogs.sas.com/content/datamanagement/2016/11/21/growing-import-big-data-quality/.

[12] H. M. N. Dilum Bandara, Xiwei Xu, and Ingo Weber. «Patterns for Blockchain Migration». In: *CoRR* abs/1906.00239 (2019). arXiv: 1906.00239. URL: http://arxiv.org/abs/1906.00239.

[13] Umeshwar Dayal et al. «Data Integration Flows for Business Intelligence». In: *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*. EDBT '09. Saint Petersburg, Russia: Association for Computing Machinery, 2009, pp. 1–11. ISBN: 9781605584225. DOI: 10.1145/1516360.1516362. URL: https://doi.org/10.1145/1516360.1516362.

[14] Tsaone Swaabow Thapelo et al. «SASSCAL WebSAPI: A Web Scraping Application Programming Interface to Support Access to SASSCAL's Weather Data». In: *Data Science Journal* (July 2021). DOI: 10.5334/dsj-2021-024.

[15] Apache. *Apache Beam Programming Guide*. Sept. 25, 2023. URL: https://cloud.google.com/dataflow/docs/concepts/streaming-pipelines.

[16] Pwint Khine and Zhao Wang. «Data lake: a new ideology in big data era». In: *ITM Web of Conferences* 17 (Jan. 2018), p. 03025. DOI: 10.1051/itmconf/20181703025.

[17] Bill Inmon. *Data Lake Architecture: Designing the Data Lake and Avoiding the Garbage Dump*. Ed. by Technics Publications. 2016. Chap. 4. ISBN: 9781634621175.

[18] Pegdwendé N. Sawadogo and Jérôme Darmont. «On data lake architectures and metadata management». In: *CoRR* abs/2107.11152 (2021). arXiv: 2107.11152. URL: https://arxiv.org/abs/2107.11152.

[19] Luca Bregata. *Development of a data mart to support decisions in fashion retail store localization*. Master's Thesis. Oct. 2019. URL: http://webthesis.biblio.polito.it/12636/.

[20] Christopher Adamson. *Star Schema: The complete reference*. Ed. by McGraw-Hill. 2010. Chap. 1, p. 5. ISBN: 9780071744331.

[21] Red Hat. *IaaS vs. PaaS vs. SaaS*. June 16, 2022. URL: https://www.redhat.com/en/topics/cloud-computing/iaas-vs-paas-vs-saas.

[22] Google LLC. *What is Platform as a Service (PaaS)?* Apr. 2023. URL: https://cloud.google.com/learn/what-is-paas.

[23] Gartner Inc. *Gartner Glossary - Software as a Service (SaaS)*. Aug. 4, 2020. URL: https://www.gartner.com/en/information-technology/glossary/software-as-a-service-saas.

[24] Tom M. Mitchell. *Machine Learning*. Ed. by MgGraw-Hill. 1997, p. 2. ISBN: 9780070428072.

[25] FY Osisanwo et al. «Supervised machine learning algorithms: classification and comparison». In: *International Journal of Computer Trends and Technology (IJCTT)* 48.3 (2017), pp. 128–138.

[26] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[27] Sagar Vaze et al. «Open-Set Recognition: A Good Closed-Set Classifier is All You Need». In: *CoRR* abs/2110.06207 (2021). arXiv: 2110.06207. URL: https://arxiv.org/abs/2110.06207.

[28] Abhijit Bendale and Terrance E. Boult. «Towards Open World Recognition». In: *CoRR* abs/1412.5687 (2014). arXiv: 1412.5687. URL: http://arxiv.org/abs/1412.5687.

[29] Lucas Pinheiro Cinelli et al. *Variational methods for machine learning with applications to deep networks*. Springer, 2021.

[30] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[31] Alec Radford et al. «Improving language understanding by generative pretraining». In: (2018).

[32] Christopher JC Burges. «A tutorial on support vector machines for pattern recognition». In: *Data mining and knowledge discovery* 2.2 (1998), pp. 121–167.

[33] S. Katagiri, C.-H. Lee, and B.-H. Juang. «Discriminative multi-layer feedforward networks». In: *Neural Networks for Signal Processing Proceedings of the 1991 IEEE Workshop*. 1991, pp. 11–20. DOI: 10.1109/NNSP.1991.239540.

[34] George EP Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[35] Alex J Smola and Bernhard Schölkopf. «A tutorial on support vector regression». In: *Statistics and computing* 14 (2004), pp. 199–222.

[36] Alex Sherstinsky. «Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network». In: *Physica D: Nonlinear Phenomena* 404 (2020), p. 132306.

[37] Zoubin Ghahramani. «Unsupervised learning». In: *Summer school on machine learning*. Springer, 2003, pp. 72–112.

[38] Jiawei Han, Jian Pei, and Hanghang Tong. *Data mining: concepts and techniques*. Morgan kaufmann, 2022.

[39] P.N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Pearson International Edition. Pearson Addison Wesley, 2006. ISBN: 9780321420527.

[40] Paul E. Black. *Manhattan distance*. Feb. 2019. URL: https://www.nist.gov/dads/HTML/manhattanDistance.html.

[41] Martin Ester et al. «A density-based algorithm for discovering clusters in large spatial databases with noise». In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[42] Donato Chiarello. *Realizzazione di un Datawarehouse e sviluppo di metodologie di Advanced Analytics a supporto delle strategie aziendali = Datawarehouse implementation and development of Advanced Analytics methods to support business strategies*. Master's Thesis. July 2020. URL: http://webthesis.biblio.polito.it/15244/.

[43] Craig Stedman. *What is data management and why is it important? Definition: data quality*. Dec. 2022. URL: https://www.techtarget.com/searchdatamanagement/definition/data-quality.

[44] Google LLC. *YouTube Analytics and Reporting APIs - Metrics*. Dec. 15, 2022. URL: https://developers.google.com/youtube/analytics/metrics.

[45] Google. *Using OAuth 2.0 for Server to Server Applications*. Aug. 10, 2023. URL: https://developers.google.com/identity/protocols/oauth2/service-account?hl=en.

[46] Google LLC. *YouTube Analytics and Reporting API - Overwiew*. Sept. 20, 2023. URL: https://developers.google.com/youtube/reporting.

[47] *Selenium Overview*. Sept. 4, 2022. URL: https://www.selenium.dev/documentation/overview/.

[48] SGOPAL Patro and Kishore Kumar Sahu. «Normalization: A preprocessing stage». In: *arXiv preprint arXiv:1503.06462* (2015).

[49] Peter J Rousseeuw. «Silhouettes: a graphical aid to the interpretation and validation of cluster analysis». In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[50] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 1990, p. 87. ISBN: 9780471878766. DOI: 10.1002/9780470316801.