Treball de Fi de Grau

Grau en Enginyeria en Tecnologies Industrials

Music feature extraction and analysis through Python

REPORT

Author: Inés Sabaté Ferrer
Supervisor: Manuel Moreno Eguílaz

Call: September 2023

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resum

En l'era digital, plataformes com Spotify s'han convertit en els principals canals de consum de música, ampliant les possibilitats per analitzar i entendre la música a través de les dades. Aquest projecte es centra en un examen exhaustiu d'un conjunt de dades obtingut de Spotify, utilitzant Python com a eina per a l'extracció i anàlisi de dades.

L'objectiu principal es centra en la creació d'aquest conjunt de dades, emfatitzant una àmplia varietat de cançons de diversos subgèneres. La intenció és representar tant el panorama musical més tendenciós i popular com els nínxols, alineant-se amb el concepte de distribució de *Cua Llarga*, terme popularitzat com a "Long Tail" en anglès, que destaca el potencial de mercat de productes de nínxols amb menor popularitat.

A través de l'anàlisi, es posen de manifest patrons en l'evolució de les característiques musicals al llarg de les dècades passades. Canvis en característiques com l'energia, el volum, la capacitat de ball, el positivisme que desprèn una cançó i la seva correlació amb la popularitat sorgeixen del conjunt de dades.

Paral·lelament a aquesta anàlisi, es concep un sistema de recomanació musical basat en el contingut del conjunt de dades creat. L'objectiu és connectar cançons, especialment les menys conegudes, amb possibles oients.

Aquest projecte ofereix perspectives beneficioses per a entusiastes de la música, científics de dades i professionals de la indústria. Les metodologies implementades i l'anàlisi realitzat presenten un punt de convergència de la ciència de dades i la indústria de la música en el context digital actual.

Resumen

En la era digital, plataformas como Spotify se han convertido en los principales canales de consumo de música, ampliando las posibilidades para analizar y entender la música a través de los datos. Este proyecto se centra en un examen exhaustivo de un conjunto de datos obtenido de Spotify, utilizando Python como herramienta para la extracción y análisis de datos.

El objetivo principal se centra en la creación de este conjunto de datos, enfatizando una amplia variedad de canciones de diversos subgéneros. La intención es representar tanto el panorama musical más tendencioso y popular como los nichos, alineándose con el concepto de distribución de *Cola Larga*, término popularizado como *Long Tail* en inglés, que destaca el potencial de mercado de productos de nichos con menor popularidad.

A través del análisis, se evidencian patrones en la evolución de las características musicales a lo largo de las décadas pasadas. Cambios en características como la energía, el volumen, la capacidad de baile, el positivismo que desprende una canción y su correlación con la popularidad surgen del conjunto de datos.

Paralelamente a este análisis, se concibe un sistema de recomendación musical basado en el contenido del conjunto de datos creado. El objetivo es conectar canciones, especialmente las menos conocidas, con posibles oyentes.

Este proyecto ofrece perspectivas beneficiosas para entusiastas de la música, científicos de datos y profesionales de la industria. Las metodologías implementadas y el análisis realizado presentan un punto de convergencia de la ciencia de datos y la industria de la música en el contexto digital actual.

Abstract

In the digital era, platforms like Spotify have become the primary channels of music consumption, broadening the possibilities for analyzing and understanding music through data. This project focuses on a comprehensive examination of a dataset sourced from Spotify, with Python as the tool for data extraction and analysis.

The primary objective centers around the creation of this dataset, emphasizing a diverse range of songs from various subgenres. The intention is to represent both mainstream and niche musical landscapes, aligning with the *Long Tail* distribution concept, which highlights the market potential of less popular niche products.

Through analysis, patterns in the evolution of musical features over past decades become evident. Shifts in features such as *energy*, *loudness*, *danceability*, and *valence* and their correlation with popularity emerge from the dataset.

Parallel to this analysis is the conceptualization of a music recommendation system based on the content of the data set. The aim is to connect tracks, especially lesser-known ones, with potential listeners.

This project provides insights beneficial for music enthusiasts, data scientists, and industry professionals. The methodologies and analyses present a convergence of data science and the music industry in today's digital context.

Contents

RESUM 3

RESUMEN 4

ABSTRACT 5

CONTENTS 7

ABBREVIATIONS AND SYMBOLS 10

LIST OF FIGURES 11

LIST OF TABLES 13

1. INTRODUCTION 16

1.1. Motivation 16

1.1.1. The Long Tail Economy 16

1.1.2. From hits to niches: why MRS matter 18

1.2. Scope 18

1.3. Prerequisites 19

1.4. Objectives 20

2. THEORETICAL BACKGROUND 21

2.1. CRISP-DM Methodology 21

2.2. Recommendation Systems Fundamentals 27

2.2.1. Knowledge Source and objects 27

2.2.2. Recommendation Techniques 27

2.2.2.1. Collaborative filtering 28

2.2.2.2. Content-based systems 29

2.2.2.3. Hybrid Systems 31

2.2.2.4. Example of the hybrid approach 31

2.3. Data Quality 31

2.3.1. Rules and dimensions 32

2.3.2. Data Profiles 33

2.3.3. Data Quality in Machine learning 33

2.4. State of the Art 34

2.4.1. Recommendations in the music domain 34

2.4.2. Artist Recommendation 35

2.4.3. Playlist Generation 36

2.4.4. The role of music feature extraction today 37

2.4.5. Popular algorithms used in music recommendation engines. 38

2.4.6. Clustering: visualization and evaluation methods 39
2.4.6.1. Data projection 39
2.4.7. Spotify Web API 39
2.4.7.1. Concepts 39
2.4.7.2. Access Methods 41
3. METHODOLOGY AND EQUIPMENT (O EXPERIMENTAL) 44
3.1. Tools, Libraries, and packages 44
3.1.1. Code Editor 44
3.1.2. Libraries and packages 44
3.2. Data Collection 45
3.2.1. Source of Data 46
3.2.2. Feature Extraction 48
3.2.2.1. Functions 48
3.2.2.2. Methodology 49
3.2.2.3. Features 51
3.3. Data Understanding 53
3.3.1. Description of the data 53
3.3.1.1. Dimensions and data types 53
3.3.1.2. Descriptive Statistics 55
3.3.1.3. Observations 60
3.3.2. Data Quality verification 61
3.3.3. Exploratory Data Analysis (EDA) 62
3.3.3.1. Correlation of features 62
3.3.3.2. Evolution over time 67
3.4. Data preparation 69
3.5. Modeling 69
3.5.1. Data Scaling 70
3.5.2. Data Projection 71
3.5.3. Clustering 71
3.5.3.1. DBSCAN 71
3.5.3.2. K-Means 73
3.5.3.3. Model Selection 76
3.5.4. Music Recommender 76
4. RESULTS AND DISCUSSION 79

5. PLANNING 82

6. ECONOMIC ASSESSMENT 83

7. ENVIRONMENTAL ASSESSMENT 85

8. SOCIAL AND GENDER EQUALITY ASSESSMENT 87

9. CONCLUSIONS 89

10. BIBLIOGRAPHY 90

References 90

Abbreviations and Symbols

API: Application Program Interface

CB: Content-Based

CF: Collaborative filtering

ML: Machine learning

MRS: Music Recommendation System

RS: Recommendation System

WCSS: Within-Cluster Sum of Squares

List of Figures

Figure 1. The Long Tail Economy. [Source: own elaboration] 17

Figure 2. CRISP-DM model diagram [4]. 21

Figure 3. Illustrates the Matrix factorization technique [Source: own elaboration]. 28

Figure 4 Client Credentials Workflow [31]. 42

Figure 5. Authorization Code Workflow [31]. 42

Figure 6. Application created to request authorization to access data. 46

Figure 7. Credentials provided by the application. 47

Figure 8. Architecture of requests from the web app to the APIs. [Source: own elaboration] 47

Figure 9. Connection to my Spotify account. 48

Figure 10. Boxplot of danceability, energy, acousticness, Instrumentalness, liveness, and valence. 58

Figure 11. Boxplot of tempo. 59

Figure 12. Boxplot of loudness. 59

Figure 13. Boxplot of popularity. 60

Figure 14. Scatter plot and histograms of the features. 63

Figure 15. Scatter plot and histograms of the most popular songs. 64

Figure 16. Scatter plot and histograms of the least popular songs. 65

Figure 17. Evolution of audio features over time. 67

Figure 18. Average music duration over the years. 68

Figure 19. 2D Projection of the data. 71

Figure 20. Nearest Neighbors distances. 72

Figure 21. Clustering with DBSCAN. 73

Figure 22. K-Means scores. 74

Figure 23. Screenshot of the scores for the computed values. 75

Figure 24. Clustering with K-Means. 75

Figure 25. Window to introduce a seed song in Visual Studio Code. 77

Figure 26. Recommended songs given two seed songs. 78

Figure 27. Gantt Chart. 82

List of Tables

Table 1. Comparison of CF and CB systems . 30

Table 2. Data Quality dimensions. 32

Table 3. Key concepts of the Spotify Web API [30]. 40

Table 4. Subgenres searched to create genre-specific datasets. 50

Table 5. Description of features [31]. 51

Table 6. Number of samples in each dataset. 53

Table 7. Data types in the raw datasets. 54

Table 8. Number of null values in each raw dataset. 55

Table 9. Descriptive statistics of the dataset. 56

Table 10. Top ten artists contributing to the dataset. 61

Table 11. Valid values for each feature. 61

Table 12. Clustering scores comparison. 76

Table 14. Overall costs of the project. 84

1. Introduction

1.1. Motivation

In the recent past, the tracking of the best-selling products was a national obsession [1]. For decades, our culture has pursued – and competed for- achieving massive popularity. In this context, the focus of our culture and economy was on a relatively small number of mainstream products at the high end of the demand curve.

The music industry is a great example of thin scene. In the past decades, the vast majority of music consumers were exposed to a few well-known artists, while smaller bands struggled to gain visibility and reach audiences. The number of records promoted through radio stations was limited, and music production executives strived to identify and promote the next big hit. The market was shaped around a "hit-driven" business model.

The rise of digital technologies and the internet has led to a significant shift in the way we consume and distribute goods and services, leading to the emergence of what is known as the “Long Tail of Retail”. This term was introduced by Chris Anderson, the editor-in-chief of Wired Magazine [2].

In his book [1], Chris Anderson argues that the *Long Tail* is the new paradigm of the digital economy. This chapter aims to provide an overview of the Long Tail Economy and how it has affected the music industry, specifically in the context of music streaming services such as Spotify and Apple Music.

1.1.1. The Long Tail Economy

The *Long Tail* is a term used to describe the shift from a traditional model of selling a few blockbuster products to selling many niche products. The *Long Tail* is made up of the many small and unique niches that make up the bulk of the market, as opposed to the popular and well-known products that make up the head of the market.

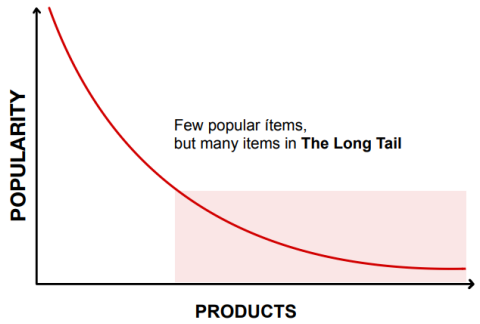


Figure 1. The Long Tail Economy. [Source: own elaboration]

One of the key drivers of the *Long Tail* is the low cost of distribution made possible by digital technologies. With the rise of the internet, it has become much easier and cheaper to distribute digital goods such as music, movies, and books. This has led to an explosion in the number of available products, and it has allowed consumers to discover and access a much wider range of products than ever before.

How are niche Markets different from conventional Markets?

One of the most frequent mistakes when addressing the lower end of the *Long Tail*, is assuming– it is full of low-quality products; bad songs in the context of music. The truth is that not all masterpieces sell well, numerous sophisticated and finely crafted songs can be found in the *Long Tail* too.

First all, let us acknowledge reality: the lower end of the Long Tail is indeed full of substandard products. Nevertheless, it also contains multiple undistinguished, sophisticated, and well-crafted works.

According to The Sturgeon Law[1], "at least 90% of almost anything you can think of, is crud. This can be said about books, movies, and songs. The reason why we do not realize this is that most products we see are likely to have been filtered or selected by retail distributors […]" [1].

Traditionally, the products we consumed were distributed by retail stores, where the proportion of good and bad is important. In traditional stores, the space is limited and the space occupied by an article eliminates the space of another. In other words, the visibility of a particular CD hides the others.

On the internet however, with an unlimited exposure space, the situation is different. There are billions of bad pages, but they do not represent the same kind of problem that billions

[1] "A humorous aphorism that holds that ninety percent of any realm of endeavor-or more generally-of everything is worthless […]."

of bad CD's could cause on the selves of a local CDs and records Store. On the web, inventory is "non-competitive", actually this is not a problem, just better filters are needed.

The *Long Tail* has significant implications for businesses and consumers.

For businesses, the *Long Tail* means that they can tap into new and previously untapped markets, and they can generate revenue from multiple niche products that would have been too expensive to sell using traditional methods. For consumers, the *Long Tail* means that they have access to a much wider range of products and they can find products that cater to their specific tastes and interests.

1.1.2. From hits to niches: why MRS matter.

The music industry has been significantly impacted by the *Long Tail*, specifically in the context of music streaming services such as Spotify and Apple Music. These services have disrupted the traditional model of music distribution by providing consumers with access to a vast library of music at a low cost. This has allowed consumers to discover and access a much wider range of music than ever before, including niche genres and artists that would have been difficult to find using traditional methods.

Music streaming services have also had a significant impact on the business side of the music industry. In the past, the traditional model of music distribution was dominated by a few large record labels who controlled most of the market. However, the rise of music streaming services has given independent artists and labels the opportunity to reach a wider audience and generate revenue from multiple niche products.

All things considered, simply offering more variety does not shift consumer's behavior by itself. Consumers must be given ways to find the niches – music genres for example- that suit their particular interest. Recommender systems are effective at doing this, driving demand down the Tail.

1.2. Scope

Stemming the motivation of this project (Read chapter 1.1) to cater music niches and the value of music features in MRS (hence, in the music industry), this project is primarily focused around the extraction, analysis, and utilization of musical features from data retrieved from Spotify using Python. This entails the construction of a dataset, an exhaustive examination of musical evolution across different time frames, and the development of a

content-based recommendation system. The implementation of the latter activities is restricted to certain aspects, which are as follows:

- Data is exclusively obtained from the Spotify API, performing searches based on music genres and subgenres. For this reason, the data set may not encompass entirely music trends or the global music scene.
- The data set is solely based on the content of the songs.
- The usage of Python programming language and the utilized libraries is based on existing methodologies, techniques, and tools.
- The sociological aspects behind the evolution of musical tastes and popularity are beyond this study.
- The creation of new algorithms or techniques is beyond the scope of this project.
- The recommendation engine is based on the content of seed songs, without extending to other forms of music recommendation such as the activity of users.

It is important to note that the data utilized in the project is limited to the veracity of the information retrieved from Spotify. This research is performed under the assumption that the metrics and features assigned by Spotify to each song are reliable.

1.3. Prerequisites

To conduct this project, certain prerequisites should be met. These requirements encompass software, hardware, and knowledge-based skills. This section provides a comprehensive breakdown of the essential prerequisites.

Hardware and Software requirements:

- Internet connection: a stable internet connection is mandatory for data retrieval from Spotify's API. Data retrieval can take hours, and an interruption of the internet connection interrupts the retrieval.
- Computer system: a reliable computer system with sufficient memory and processing capacity to process and store large csv files is crucial.
- Python Environment: having Python 3 installed is the primary software prerequisite, given the project's dependency on Python for data extraction, analysis, and model development.
- Spotify Developer Account: a developer account that provides access to Spotify's Web API is essential to enable data retrieval.
- Python libraries: installing and utilizing specific libraries within the Python environment is crucial throughout the project.

Knowledge and skills:

- Python Programming: the entire data pipeline, from extraction to analysis and recommendation engine development, is grounded in Python.
- Data Analysis: an understanding of data analysis techniques and statistical methods is key for drawing insights from the dataset.
- Music Understanding: a basic knowledge of musical features (for instance, acousticness, key, mode) will help in contextualizing and interpreting the data.
- Machine Learning: familiarity with basic machine learning concepts and algorithms, especially those related to recommendation systems.

1.4. Objectives

Essentially, the objective of this project is the development and analysis of a data set obtained from Spotify utilizing Python. The purpose of retrieving data and building a dataset is to examine its features and use the gathered data to construct a music recommendation system.

Once the main objective is stablished, it can be subdivided into different secondary, and more specific, objectives thar focus more on the procedure and methodology of the project. The detailed objectives are as follows:

- Creation of a dataset from scratch: Develop a dataset through information extraction from Spotify. This dataset will serve as the foundation for subsequent analyses, and is intended to encompass songs from as many subgenres as possible, acknowledging niches and the “Long Tail” distribution of the market. (Read chapter 1.1.1)

- Analysis of music features: Utilize Python libraries for the examination of musical evolution over past decades. Investigate trends and significant shifts over periods.

- Correlation between features and popularity: Investigate the correlation between specific features and popularity to discern if this relationship has shifted over periods. Identify characteristics, if present, that have influenced popularity through time.

- Construction of a music recommendation system: Utilize the data from the established dataset to develop a recommendation engine based on content. Upon input of certain seed songs, provide song suggestions of similar nature.

2. Theoretical background

2.1. CRISP-DM Methodology

CRISP-DM stands for "Cross Industry Standard Process for Data Mining". The CRISP-DM methodology can be viewed as “the canonical approach from which most of the subsequent proposals have evolved (both for data mining and data science process models).

The goal methodology is to transform specific situations (case-specific scenarios) and broad patterns or tendencies (general behaviors) into a more universal and applicable approach that can be used across different domains or industries.

As a process model, CRISP-DM provides an overview of the data mining life cycle [3]. The life cycle model consists of six phases with arrows indicating the most important and frequent dependencies between phases. The sequence of the phases is not strict. In fact, most projects move back and forth between phases as necessary. The six steps that comprise the model are shown in Figure 2..

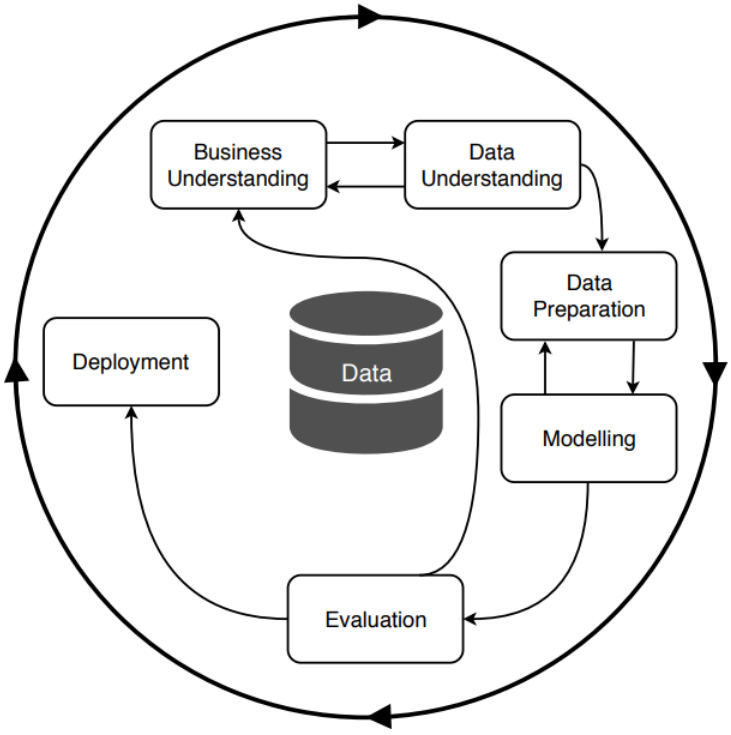


Figure 2. CRISP-DM model diagram [4].

Step 1. Business Understanding: the first step involves fully understanding the business context and data mining goals. Taking time to explore what is expected to be gained from data mining is crucial. The objective of this step is to get to know the reasons for investing effort in data mining. To do so, the *IBM Documentation [3]* suggests the following considerations:

I. Determine objectives: gain as much insight as possible into de business goal for data mining.
    a. Gather background information.
    b. Document specific objectives.
II. Success criteria: each objective listed in *Step I* should have a correlative criterion for success.
III. Assessing the situation: evaluate the current situation and resources.
    a. Determine the data sources.
    b. Identify the biggest risk factors.
    c. Make a contingency plan for each risk.

Step 2. Data Understanding: the second step entails exploring the available data to better understand its characteristics and composition. Looking closely at the available data is critical to avoid unexpected problems during data preparation. This step is comprised of the following sections:

I. Collecting data: this section not only includes gathering information from different resources, but also considering which attributes of the database seem more promising, or which appear to be irrelevant and could be excluded.
    a. Existing Data: existing data can include transactional data, survey data, web logs, and more. One should always assess whether this data is enough to meet the project's requirements, and if there is enough data to draw generalized conclusions.
    b. If the existing data is not enough, consider supplementary data such as purchasing demographics from an organization.
    c. If neither of the latter sources provide sufficient information, conduct surveys, or track additional data.

II. Describing data: there are various methods to depict data. Most of them focus on what is the amount of data available, and the condition of the data [2]. Some key characteristics to address when describing data are:

[2] "Data quality assesses your data against standard technical performance measures such as completeness, accuracy, timeliness, validity, and uniqueness. These measures can be applied to any dataset at any company regardless of industry. Data condition includes an assessment of how

a. Amount of data: it is important to bear in mind that large datasets can improve the accuracy, but can also lengthen processing times. If the dataset is too large, it might be more effective to work with smaller subsets of data. When addressing the amount of data, one should consider both the number of records as well as fields (attributes).
b. Data types: Data types: Paying attention to value types can cut off problems during later phases of the project. Data can take multiple formats; in Python, the main data types are numerical (int or float), categorical (strings), Boolean (True/False), and date types.
c. Coding Schemes[3]: note if the data contains any conflicting schemes.

III. Exploring the data: exploring the data involves using charts, tables, and plots to gain a better understanding of the data [3]. This analysis can help address the goal constructed in *Step I*, and formulate new hypothesis. In addition, exploration can suggest data transformation tasks that need to be done during data preparation.

IV. Verifying Data Quality: verifying data quality is critical, especially when the data is meant to be merged, or obtained from external data sources. Quality assurance is often discipline-specific, and expectations and standards may vary [4], "there are two main axes for evaluating data quality: (1) the integrity of data itself, and (2) the fit or appropriateness of the data with respect to a particular question or problem." [5] (see Section 2.3).

Step 3. Data Preparation: this is one of the most time-consuming phases of a data mining project. It is estimated that data preparation can take up to 70% of a project's time and effort. Data preparation typically includes choosing subsets of data, sorting the data for modeling, deriving new attributes, and dealing with missing data [4].

I. Selecting data: It involves selecting items (rows) to include or exclude from the dataset, as well as selecting attributes or characteristics (columns). This includes making decisions about which features to include in the subsequent analysis. It is important to document the rationale for including or excluding data.

---

the data is used, the value it brings to the business, and the systems that support its collection, maintenance, and usage." [3].

[3] In the context of data analysis, a coding scheme refers to a structured set of categories or codes that are used systematically to classify and organize qualitative data. It is employed as a means to analyze categorical data, facilitating the process of identifying patterns, or relevant information within the data. Each code within the scheme represents a specific concept or range of values, and data analysts assign these codes to relevant segments of the data during the coding process. This approach is ideal for working with large datasets.

II. Cleaning data: data at an entry point is not likely to be clean [5]. Raw data usually contains omissions, error messages, extreme values, and incompatible formats. With the right approach to data cleaning, the latter issues can be mitigated. Some of the common ways data needs to be cleaned are the following:
    a. Identifying and removing outliers (see Section 2.3).
    b. Missing data: In many cases missing fields do not pose a problem. However, there are instances when all fields must have a corresponding value. In such cases, missing values can be approximated using techniques such as interpolation, extrapolation, or labeling similar data with categories. Alternatively, the incomplete records or features can be removed from the dataset.
    c. Time zone conversion: "Comparing timestamps to one another is possible only with some standard of truth. Often, this is Coordinated Universal Time (UTC). UTC is not a time zone, but a time standard (countries using Greenwich Mean Time [GMT] happen to always agree with UTC, but they do not use UTC)" [7].
    d. Date format consistency: "any date and time format string that contains more than one character, including white space, is interpreted as a custom date and time format string" [8]. In order to convert dates into datetime objects[4], dates should be formatted to standard formats.
    e. Coding inconsistencies: in case there were coding schemes detected in the point *II.c*, it is crucial to verify their consistency across the entire dataset and fix any inconsistencies [3].

III. Integrating Data: frequently, data is segregated into multiple datasets, or data sources. To integrate all the information using Python, the *Pandas* library offers multiple tools for combining datasets using set[5] logic for indexes and relational algebra for join/merge-type operations.

IV. Formatting data: before building a model, it is important to check if certain techniques require specific data formatting or ordering [3].

[4] "The datetime module in Python provides classes for manipulating dates and times" [9]. The name datetime refers to both the datetime library and to one of the Python datatypes.

[5] In Python, a set is a collection of unique data. That is, elements of a set cannot be duplicate nor mutable [10].

Step 4. Modeling [6]*:* At this stage, the effort dedicated in the latter phases begins to pay off. Modeling usually involves conducting multiple iterations. It is rare to find a solution to the problem posed in step 1 with a single model and running a single execution. Typically, the first executions will reveal new parameter adjustments, or requirements that will imply reverting back to the data preparation phase.

I. Selecting modeling techniques: there are multiple algorithms existing to solve different problem tasks, but one cannot expect to achieve good performance without pondering on the different existing machine learning algorithms. “Each classification algorithm has its own biases, and no single algorithm enjoys superiority among others if we do not make any assumptions about the task” 0. In practice, the selection is typically made based on the following considerations:
    a. The number of features or samples in the dataset.
    b. The amount of noise in the dataset.
    c. Whether the classes are linearly separable[7] or not.
    d. Model requirements such as specific data types.
    e. Whether the model requires splitting the data into training and testing subsets or not.
    f. The level of data quality required by the model, the granularity of the dataset, and expected accuracy of the results [8].

II. Testing design: Testing design involves exploring the available techniques for evaluating the model’s performance, and defining a success criterion.

Step 5. Evaluation: evaluation involves assessing the outcomes to confirm and justify the appropriateness of the model, as well as identifying potential issues that may hinder the functionality of the model. This phase can be broken into three main tasks [11]:

[6] Building mathematical models to help understand data. These models are given parameters that can be fitted to observed data. Once the model is fit to the observed data, it can be used to predict aspects of newly observed data.

[7] A class is separable if there is a classifier whose decision boundary separates the positive objects from the negative ones. If such a decision boundary is a linear function (straight line through the plane containing two variables) of the features, the classes are linearly separable. In this scenario, the points on the same side of the boundary line fall in the same group.

[8] Selecting the model based on the granularity of the data is crucial. The level of detail of a model must match the granularity of the data. The same applies to the expected accuracy of the outcome. If the quality of the data is not the best, using a les granular model will provide more reliable results.

I. Evaluating results: assessing the degree to which the results meet the objectives of the project.

II. Reviewing the process: reviewing each of the previous phases in order to disclose important factors that may have been overlooked, or identify data quality issues. This task includes summarizing the process ad highlight activities that have been missed or could be improved.

III. Determining next steps: this task involves creating a list of possible next steps to move forward with the project. Each step should be accompanied by a description an explanation of the pros and cons.

Step 6. Deployment: deployment refers to the process of taking the solutions generated in the modeling phase and putting them into action. Generally, the deployment phase involves three types of activities: (1) planning and monitoring the deployment of results, (2) planning and monitoring maintenance, and (3) completing wrap-up tasks such as producing a final report and conducting a project review.

I. Planning and monitoring the deployment of results: to plan a comprehensive deploy of results, one should summarize the results of the project, and determine which findings should be presented. Additionally, create a step-by-step guide for deployment and integration of each of the deployable models.

II. Planning monitoring and maintenance: this involves ensuring that the solutions are functioning correctly, fixing any issues that arise, and making any necessary updates to the solutions.

III. Producing a final report: the report should include most of the following points [3]:
    a. Description of the original problem.
    b. Description of the process and conducted tasks.
    c. Costs of the project.
    d. Deviations from the original plan.
    e. A summary of the results.
    f. An overview of the proposed plan for development.
    g. Recommendations for further tasks, including interesting leads discovered during exploration and modelling.

2.2. Recommendation Systems Fundamentals

2.2.1. Knowledge Source and objects

There are different approaches to build a recommendation system (RS). Therefore, the data used by these systems can be very diverse. Some of these techniques use very simple data such as user ratings for items, known as knowledge poor techniques. Other techniques rely on ontological descriptions of the data, or relations and interactions between users, these are known as knowledge-dependent techniques [13].

Regardless of the type of involved knowledge source, there are three kinds of objects [14]:

- *Items:* items are the objects that the system recommends. Generally, a recommender system focuses on only one item (e. g. songs).
- *Users:* users interact with the system, providing information about their preferences and needs. This information is used to generate personalized recommendations that match the user's interests and improve their overall experience with the system. Music recommendation systems (MRS) exploit a range of information about the users, such as ratings, demographics, and past behavior, may be used depending on the recommendation technique employed by the system.
- *Transactions:* in recommender systems, transactions are recorded interactions between a user and the system that store important information used by the MRS algorithm. These interactions can include the item selected by the user, the context of the recommendation, and any explicit feedback provided by the user, such as ratings. The most common form of transaction data is ratings, which can be collected either explicitly or implicitly.

2.2.2. Recommendation Techniques

Among the approaches commonly used by recommender systems, six main ones can be mentioned [15]:

I. *Content-Based Systems,* which recommends items that are similar to the ones that one user liked in the past, based on the features associated with the compared item.
II. *Collaborative Filtering,* based on ratings made by a population and making a consensus of them.

III. *Demographic Filtering, based* on user characteristics such as gender, country, language, or age.
IV. *Community-Based, used* in social networks, relying on user's friends' ratings to make recommendations.
V. *Knowledge-Based, systems that* use information provided by the user to give a ranking.
VI. *Hybrid Filtering,* which combines two or more of the above-mentioned types of recommender systems.

In the context of music RSs, there are three primary methods for generating personalized recommendations for users: collaborative filtering, content-based, and the hybrid approach.

2.2.2.1. Collaborative filtering

Collaborative filtering (CF) systems are built on the basis of user's overlapping preferences and ratings of songs. In this case, the system provides reasonable predictions of a user's preferences for an item that the user has not yet rated. The fundamental assumption is that users that act similarly have the same preferences. Netflix's "Movies You Might Like" list, and the "Weekly Discovery" section in Spotify, are successful examples of CF systems [16]. There are two important approaches when implementing CF systems:

The first family of methods is the *matrix factorization* technique, which involves storing all data in a Users-Songs matrix. Through implicit or explicit feedback, values are assigned in the Users/Songs matrix. For example, if a user listens to a song, the value is 1, otherwise, it is 0. If the user rates a song, the value is the given number of stars. This results in a large matrix that can be reduced by Spotify using matrix factorization to approximate the matrix by an inner product of two smaller matrices. This approach results in two types of vectors for each listener - a user vector X and a song-vector Y.

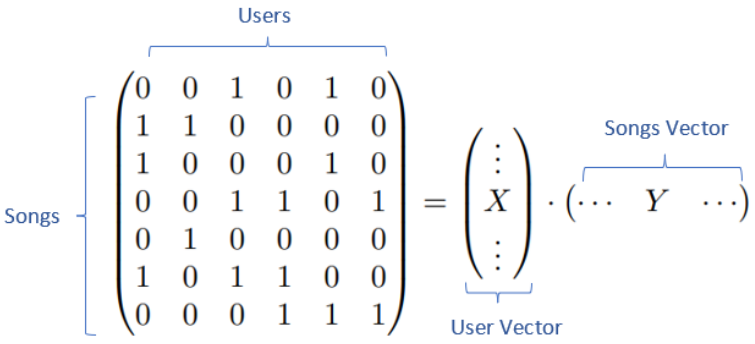


Figure 3. Illustrates the Matrix factorization technique equation [Source: own elaboration].

Once vectors X, and Y are calculated, the system will find similarities by comparing either:

- User-User similarity: Finding users with similar tastes by comparing their vectors.
- Item-item similarity: comparing the songs vectors with others and finding similar Y compared to the actual record of songs of a particular user.

There is a second approach called *model-based*: the goal is to predict the user's rating for missing items using machine learning models [16].

The key advantages of collaborative filtering are that it *does not require extracting features* from the items, so there is no need for studying the metadata of the files, nor for having in-depth knowledge of music and audio signaling. Moreover, this model brings *serendipity*[9] to the system, which is a desirable property in RSs as it can increase user engagement and satisfaction.

Collaborative filtering has three major drawbacks. A common problem in CF is *cold start.* Cold start refers to the situation where the recommendation system faces a new user or a new item for which there is not enough data available to make accurate recommendations. There are two types of cold start problems: new user and new item.

The new user problem occurs when a new user joins the system. Until the system collects enough data about the user's behavior, it cannot make personalized recommendations. The same happens when a new item is introduced in the system. If there is not enough data about the item, the system cannot rely on user data to make recommendations, and it may use content-based features or other external data sources to make recommendations.

Another issue is *scalability,* as a vast number of users and items require significant computing resources. The last limitation of CF is data sparsity and high dimensionality. When the number of users is relatively large, the main problem is the low coverage of the user's ratings among the items. It is common to have a sparse user-item matrix of 1% (or even less) coverage.

2.2.2.2. Content-based systems

Content-based (CB) systems rely on the similarity between particular items, such as songs. A user of a streaming music service might provide feedback on songs they like or dislike and create playlists. The system can analyze features extracted from the song and compares them with other songs to suggest similar items. These systems assume that if a user likes a particular song, they are likely to enjoy other songs that share similar

9 The property of a recommendation that is novel, surprising, and valuable [7].

characteristics. The CB method does not require any feedback from the listener, as predictions are only based on sound similarity, deduced from the features extracted from previously listened songs.

To determine similarities, features that best describe the music are extracted, and Machine Learning algorithms recommend items that are closest to what the user already likes. To create item profiles, it is necessary to extract features from each item. Additionally, user profiles that include both their preferences and history on the platform are required. These profiles consist of a list of weights, where each weight corresponds to a selected feature's importance.

CB methods have its roots in the information retrieval field, where feature extraction is performed.

A positive aspect of content-based recommendations is that new and unknown music can be recommended just as easily as popular or timeless music. This *helps new artists* with a limited number of views to be discovered, and users find songs at the bottom of the *Long Tail* (see chapter 2.4.2).

The issue of the cold start regarding new items can be avoided because new items can be recommended immediately, unlike collaborative filtering methods that require integration time.

A negative aspect of this method is that it *tends to over-specialize*, limiting diversity of the recommendations. Additionally, new users cannot receive immediate recommendations, and they must listen and evaluate a certain number of songs before receiving recommendations, this issue is called *new user cold start*.

Table 1. Comparison of CF and CB systems .

	Collaborative filtering	Content-based
Advantages	• No need for extraction and analysis of audio features. • Serendipity	• More likely to recommend unknown tracks. • New Item cold start is avoided.
Negative points	• New user cold start • New item cold start • Scalability • Sparsity	• New user cold start: new users must evaluate a certain number of songs before receiving relevant recommendations.

		• Limits diversity: CB models tend to over-specialize.

2.2.2.3. Hybrid Systems

A *hybrid recommendation system* can be created by combining the previous stand-alone methods. This approach aims to mitigate the issues of cold start and sparsity.

There are several ways to implement a hybrid system such as combining all recommendation systems into one, keeping multiple systems separate and assigning weights to them, or allowing the ability to switch between systems. Additionally, results from one system can be extracted and used as input for the next.

2.2.2.4. Example of the hybrid approach

Imagine a team is designing a music recommendation system for a streaming service, using both collaborative filtering and content-based filtering to make recommendations to users.

To start, they collect data on the music listened by each user, as well as data on the characteristics of each song (genre, tempo, instrumentation, ...). By using collaborative filtering, the system can recommend songs to users based on the listening history of other users who have similar preferences.

However, since collaborative filtering can suffer from cold start problems for new users, they also incorporate content-based filtering. Adding this method allows the system to recommend songs based on the characteristics of the songs that the user has previously listened to and liked.

This is the scenario of a hybrid approach, where they assign weights to each of the two methods based on how well they perform for each individual user. For example, if a user has a lot of data available in their listening history, the system will give more weight to collaborative filtering. On the other hand, if a user has a smaller listening history, the system may give more weight to content-based filtering.

This hybrid approach allows the system to take advantage of the strengths of both collaborative and content-based filtering, while also addressing their respective weaknesses. By combining the two methods, the system can provide more personalized and accurate recommendations to users.

2.3. Data Quality

Data quality is often defined as a measurement of the degree to which data is fit for purpose. However, it is a multifaceted concept that encompasses various aspects [17]. There are

over 60 dimensions identified in the data management literature [18]. According to [19], there are six essential dimensions that determine the quality of a dataset: accuracy, completeness, consistency, reliability, and relevance.

2.3.1. Rules and dimensions

A data quality rule serves as a standardized criterion that assesses the accuracy, integrity, and compliance of data with defined business requirements. These rules act as quality checkpoints, evaluating data to ensure it aligns with predetermined expectations. Data quality rules can operate at two levels: dataset and feature.

Feature level rules focus on validating individual features and characteristics within a dataset. These rules are generally applied to validate the data type, format, range, or distribution of a particular feature. For example, a rule may specify that a certain field should contain only integer values between a given range of numbers, or require dates to appear in a day/month/year format.

Dataset level rules address the overall quality of the data set. They are used to check for accuracy, completeness, and consistency among features in the dataset. These rules can also be used to identify missing values or duplication of records that can lead to incorrect analysis.

Table 2 provides a brief description of the six dimensions of data quality and how rules address each of them at dataset, and feature levels.

Table 2. Data Quality dimensions.

Dimension	Dataset Level	Feature Level
Completeness	Refers to the extent to which all expected data records are present in the dataset.	Focuses on whether specific attributes or fields within a record have non-null, valid values [20].
Timeliness	Represents the degree to which data is made available within a specified timeframe.	Examines whether individual attributes are populated or updated within defined timeframes.
Validity	Measures the extent to which data conforms to predefined standards or rules.	Focuses on ensuring specific attribute values adhere to predefined business rules or constraints.

Consistency	Addresses the agreement between different sources of data or within datasets over time.	Ensures consistency of data values across related attributes or tables to avoid contradictions or conflicts.
Accuracy	Reflects how closely data values align with the actual attributes or events they represent.	Focuses on verifying attribute values against external, trusted sources to validate their correctness.
Uniqueness	Addresses the presence of duplicate records or values within a dataset, ensuring distinctiveness.	Ensures unique values for specific attributes to avoid redundancies and data ambiguities.

2.3.2. Data Profiles

"Data profiling is the activity of running statistics on a data set to better understand the data and field dependencies" [21]. Examining the profile of a dataset is often performed at the earliest stage of a project. It involves examining the data to gain insights into its properties, patterns, and potential issues. Understanding the profile of a dataset is essential to find anomalies, identify data quality issues, and write useful quality rules.

Profiling a dataset or a column includes measuring the percentage of populated records, the number of null values, identifying valid values, data types, and formats [22].

2.3.3. Data Quality in Machine learning

"The quality of the data and the amount of useful information that it contains are key factors that determine how well a machine learning algorithm can learn" [23]. Therefore, it is critical to examine and preprocess a dataset before feeding the learning algorithm.

The main aspects to focus when addressing the quality of a dataset for a machine learning algorithm are (1) removing or imputing[10] missing values, (2) getting categorical data into shape for machine learning algorithms, and (3) selecting relevant features for the model construction [23].

[10] Interpolation technique to estimate the missing value by the mean value of the entire feature column.

2.4. State of the Art

2.4.1. Recommendations in the music domain

This section approaches recommendation systems in the music domain exclusively.

The main task of music recommendation systems (MRS) is to suggest interesting music, which usually consists of a mix of known and unknown artists or tracks. Given a user profile, most of the work related MRS focuses on:

I. *Artist Recommendation: Presenting a curated list of recommended artists or bands based on a user's listening history or preferences.*
This type of recommendation is based on the similarity between the user's listening history and the characteristics of the recommended artists or bands.
Examples of Artist Recommendation tools:
- Spotify's "Discover Weekly" feature, which uses collaborative filtering to recommend new artists and songs based on a user listening history and preferences.
- Tidal's "My Mix" feature, which uses a combination of collaborative filtering and content-based filtering to recommend new artists and songs based on a user's listening history, preferred genres, and favorite artists.

I. *Generating personalized playlists, either for individual users or for a group of users with similar tastes.*
This type of recommendation is more specific than the first scenario, as it involves selecting specific songs that the user may like and arranging them in a playlist format. The system may use information about the user's listening history, preferences, and contextual factors to generate the playlist, such as time of day or activity being performed while listening to the playlist. Technologies used in this scenario:
- Apple Music's "New Music Mix" feature, which generates a personalized playlist for individual users based on their listening history and preferences, as well as the latest music releases.

- JQBX.fm, which allows users to create virtual rooms where they can listen to music together and generate personalized playlists based on the collective listening history and preferences of the group.

There are other interesting scenarios. The following list includes a brief description and the example of a technology used in each use case.

I. *Providing recommendations for new songs or albums to users based on their listening history or preferences.* Spotify's "Discover Weekly" feature uses collaborative filtering to recommend new songs or albums to users based on their listening history and preferences.

II. *Facilitating music discovery by suggesting songs or artists that are similar to those that a user has already enjoyed.* Pandora's "Music Genome Project" uses music analysis to suggest songs or artists that are similar to those that a user has already enjoyed.

III. *Creating custom radio stations based on a user's favorite songs, artists, or genres.* Pandora's "Thumbprint Radio" creates a custom radio station based on a user's "thumbs up" on various songs, artists, or genres.

IV. *Powering intelligent group music recommendation, where a system recommends music to a group of users based on their collective listening history or other contextual factors.* Flytrap is an example of an intelligent group music recommender that recommends music to a group of users based on their collective listening history and contextual factors such as location, time, and weather.

V. *Supporting music search and discovery by enabling users to browse for new music using a variety of search criteria, such as genre, era, mood, or tempo.* MusicMap is a music discovery platform that allows users to discover new music by browsing through a map of genres and artists. The platform also allows users to search for music based on mood, era, tempo, and other search criteria.

2.4.2. Artist Recommendation

Artist recommendation follows the user-item matching[11]. By matching users and items, these engines only provide the user with a list of interesting items. However, some systems are enriching their recommendations with the use of Really Simple Syndication[12] (RRS).

11 In user-item marching, items are recommended to users based on their profiles.

12 According to [8], an RSS feed is a set of instructions residing on the computer server of a website, which is given upon request to a subscriber's RSS reader, or aggregator. The feed tells the reader

Really Simple Syndication (RSS) is a web feed format used to publish frequently updated content, such as blog entries, news headlines, and podcasts. It allows users to subscribe to a feed and receive new content automatically in their feed Reader [23].

RSS could be used in artist recommendation to deliver updates and new content related to a user's favorite artists. For example, a user could subscribe to an RSS feed for an artist they like and receive updates on new album releases, tour dates, and interviews. This could help the user stay up-to-date on their favorite artists and discover new music recommendations based on the related artists that are often featured in these updates.

For instance, *iTunes Music Store* provides an RSS feed generator, updated weekly, so users can stay up to date with new music releases, podcasts, concert listings, and unknown - long tail - artists.

2.4.3. Playlist Generation

In music, the terms mix and playlist refer to a collection of songs that are grouped together. However, it is important to acknowledge their differences.

A *mix* is a type of playlist that is created for a specific occasion, mood, or atmosphere, and is usually composed of songs that are mixed together seamlessly to create a continuous flow of music. In contrast, a *playlist* is a collection of songs that are grouped together based on a theme, genre, or personal preference, but the songs are played in their entirety without being mixed.

This distinction is also acknowledgeable in the Oxford English Dictionary [24], which defines a mix as "a sequence of recordings, typically of popular music, selected and combined by a disc jockey or music enthusiast to create a particular mood or atmosphere," while a playlist is defined as "a list of recordings to be played, especially as background music, in a particular order or for a particular purpose."

With the advent of more sophisticated algorithms and the availability of vast amounts of data, playlist generation has evolved significantly in the recent years. There are several ways to automatically generate playlists:

Shuffle/random playlists: Shuffling is still a common way to generate playlists, but studies have shown that it has limitations in terms of providing personalized recommendations. While it can create serendipitous moments and unexpected rediscoveries, it does not take into account the user preferences or listening habits. The shuffling method does not involve content-based or collaborative filtering approaches as

when new material—such as a news article, a blog posting, or an audio or a video clip—has been published on the website.

Seed-based generation: This method involves creating a playlist based on a given seed song or artist, where the system analyzes the characteristics of that song/artist and generates a playlist with similar songs. This method provides a more personalized approach than shuffling, as it considers the user musical preferences and can result in more focused playlists. Seed-based generation recommendations are content-based, as they analyze the features of the seed song or user profile, respectively.

Neighborhood recommendation: This method aims to find like-minded users and recommend music based on their listening habits. The system matches users based on their profiles and creates communities or clusters of users with similar interests. This approach can result in more diverse and varied playlists, as users can discover new music through their neighbors. Neighborhood recommendation is a collaborative filtering approach, as it recommends music based on the preferences of similar users in the neighborhood.

Rather than shuffling, music recommendation engines today rely on more advanced algorithms that use music feature extraction, content-based and collaborative filtering approaches, and seed-based generation. However, shuffling can still be used as a complementary feature to these methods, providing variety and randomness to the playlists.

2.4.4. The role of music feature extraction today

The role of music feature extraction in MRS is crucial, as it helps to identify and analyze the underlying patterns and characteristics of a song. Due to the rise of machine learning (ML) and deep learning algorithms, music recommendation engines have seen significant advancements, making it possible to extract more complex features and generate more accurate recommendations. Nowadays, Python is widely known as the go-to language for developing music recommendation engines due to its versatility and powerful libraries.

Music feature extraction involves analyzing different aspects of a song, including its tempo, melody, rhythm, pitch, and timbre. Traditionally, feature extraction was carried out using hand-crafted features, where experts manually identify and extract relevant features from the audio signal. This approach is time-consuming and may not capture all relevant features.

ML and deep learning algorithms have changed the paradigm of feature extraction, allowing to automatically extract a wide range of features using neural networks. The use of ML is not essential in MRS, but they it can analyze and identify complex patterns in audio signals that would be difficult or impossible for a human expert to detect.

2.4.5. Popular algorithms used in music recommendation engines.

This section provides an overview of machine learning (ML) basics in music recommendation engines and how they are applied to feature extraction.

Clustering algorithms: "Clustering is an exploratory data analysis technique that allows us to organize information into meaningful subgroups (clusters) without having any prior knowledge of their group memberships. Each cluster that may arise during analysis defines a group of objects that share a certain degree of similarity" [22]. In the context of music, clustering algorithms can be used to group songs based on their similarity in terms of features such as tempo, rhythm, and timbre.

- *K-means: "*clustering algorithm that aims to partition data into K clusters" [22]. The k-means algorithm belongs to the category of prototype-based clustering. Prototype-based clustering means that each cluster is represented by a prototype. In the context of continuous data, the prototype is termed *centroid*, which is the average of all data points in the cluster. In the context of categorical features, the prototype is known as *medoid*, and is the most representative or most frequently occurring point in the cluster. The algorithm works by iteratively assigning data points to the nearest centroid and updating the centroid of each cluster based on the mean of the data points assigned to it. K-means is a simple and efficient algorithm that can be used for clustering large datasets. In music feature extraction, k-means can be used to cluster songs based on their features, allowing for the creation of playlists or personalized recommendations based on a user's preferred cluster [25].

- *DBSCAN* (Density-Based Spatial Clustering of Applications with Noise): commonly used clustering algorithm in music feature extraction. DBSCAN works by grouping together data points that are closely packed together, while also identifying points that are outliers and do not belong to any cluster. DBSCAN is particularly useful in music feature extraction because it can identify clusters of varying shapes and sizes, and it does not require the user to specify the number of clusters beforehand. This makes it a useful tool for discovering new and previously unknown patterns in music data.

Classification algorithms: such as support vector machines (SVMs) and decision trees, they are also used in music feature extraction to identify patterns in the audio signal and classify songs into different genres or moods based on their feature vectors.

k-Nearest Neighbors (k-NN): compares a new data point to its k closest neighbors in a training set and assigns it to the most common class among those neighbors. In MRS, *k-NN* can be used to predict the genre of a new song based on its features.

- *Support Vector Machine (SVM):* finds a hyperplane that separates data points into different classes based on their features. *SVMs* are able to identify the instrument used in a given audio signal based on its frequency spectrum or timbral features.

Advanced Classification Algorithms:

- *Deep Neural Networks (DNNs):* are capable of learning hierarchical representations of audio signals to capture more complex relationships between features and classes. DNNs can be trained to classify music based on its emotional content using a dataset of labeled audio files

2.4.6. Clustering: visualization and evaluation methods

2.4.6.1. Data projection

T-distributed Stochastic Neighbor Embedding: T-distributed Stochastic Neighbor Embedding (t-SNE) is a technique for visualizing high-dimensional datasets in lower dimensional graphs through dimensionality reduction [26]. The main purpose of projecting data into a 2D is to facilitate human comprehension, given our innate capacity to interpret two or three-dimensional data [27].

T-SNE operates by converting high-dimensional Euclidean distances[13] between data points into conditional probabilities, capturing intricate polynomial relationships in the data. The method functions by representing similarities in the original space using Gaussian joint probabilities and mapping them to a lower-dimensional space using Student's t-distributions [28]. A noteworthy strength of t-SNE is its "adeptness at preserving local structures, making it particularly effective in maintaining natural clusters or groups in the 2D representation" [29].

2.4.7. Spotify Web API

The Spotify Web API is a set of web-based endpoints and functionalities provided by Spotify for developers. It enables developers to interact with Spotify's streaming service programmatically, accessing features such as retrieving track information, searching for music, managing playlists, and controlling playback. It provides a way for developers to integrate Spotify's music streaming capabilities into their own applications or services [30].

2.4.7.1. Concepts

This chapter aims to provide a clear understanding of the key elements involved in utilizing the Spotify API effectively. The fundamental concepts associated with the Spotify Web API are presented in *Table 3* .

[13] The Euclidean distance, in Euclidean space, is "the length of a straight line segment that would connect two points. In such a space, the distance formulas for points in rectangular coordinates are based on the Pythagorean theorem." [31]. In the context of projecting large datasets, this distance is calculated between vectors, where each vector represents a row.

Table 3. Key concepts of the Spotify Web API [30].[14]

Concept	Description
Access Token	The Access Token is a security credential that allows your application to access the Spotify Web API. It acts as a digital key that grants permission to make requests and retrieve data from the API.
API Calls	API Calls refer to the requests made to the Spotify Web API to perform various operations, such as retrieving metadata, creating playlists, or getting recommendations. These calls are made using specific endpoints provided by the API.
Apps	Apps refer to the applications or services that developers create to interact with the Spotify Web API. These apps can range from music recommendation systems to playlist management tools, offering users new and personalized experiences.
Authorization	Authorization is the process of obtaining permission from users to access their Spotify accounts and retrieve their data. This process ensures that user privacy and data security are maintained. The Spotify Web API supports different authorization flows, such as the Authorization Code Flow and the Client Credentials Flow, depending on the level of access required (see Section 2.4.7.2).
Playlists	Playlists are a fundamental aspect of Spotify's music streaming service, and they are also an integral part of the Spotify Web API. With the API, developers can create, retrieve, modify, and manage playlists programmatically. This allows for dynamic playlist generation, collaborative playlists, and playlist-based recommendations.
Quota Modes	The quota mode refers to the mode in which an app can be: development mode or extended quota mode. These modes specify the maximum number of API calls allowed within a given time period, preventing abuse or excessive usage.

[14] For detailed information and examples on each concept, please refer to the official Spotify Web API documentation [30].

	Quota Modes define the rate limits and access restrictions imposed on API calls.
Rate Limits	Rate Limits are the restrictions imposed on the number of API calls that can be made within a specific timeframe. They are in place to manage server resources and maintain system stability. Developers must be mindful of rate limits to avoid hitting usage restrictions. Spotify's API rate limit is calculated based on the number of calls that your app makes to Spotify in a rolling 30 second window.
Scopes	Scopes determine the level of access and permissions granted to an application when a user authorizes it. By defining scopes, developers can request specific permissions to perform actions such as reading user data, modifying playlists, or playing music. Scopes provide granular control [15]over the data and functionality accessible through the API.
Spotify URIs and IDs	Spotify URIs (Uniform Resource Identifiers) and IDs are unique identifiers used to reference specific entities within Spotify's ecosystem. Spotify URI is a string that identifies a track, album, artist, or playlist, while an ID is a shorter alphanumeric code representing the same entity. These identifiers are essential for performing targeted operations and retrieving specific content.
Track Relinking	Track Relinking is a feature of the Spotify Web API that handles the availability of music content across different regions. Due to licensing restrictions, certain tracks may not be available in all countries. The API provides mechanisms to automatically substitute unavailable tracks with equivalent alternatives, ensuring a consistent listening experience for users worldwide.

2.4.7.2. Access Methods

The Spotify API provides developers with two primary methods of accessing its services: the Client Credentials Flow and the Authorization Code Flow.

[15] "Granular access control is a concept in computer science that refers to the practice of granting differing levels of access to a particular resource to particular users. Access determines what a user is authorized to do in a system." [31].

Client Credentials Flow: aims to access general resources provided by Spotify. This method is suitable when an application does not require access to user-specific content. the application itself is authenticated rather than an individual user.

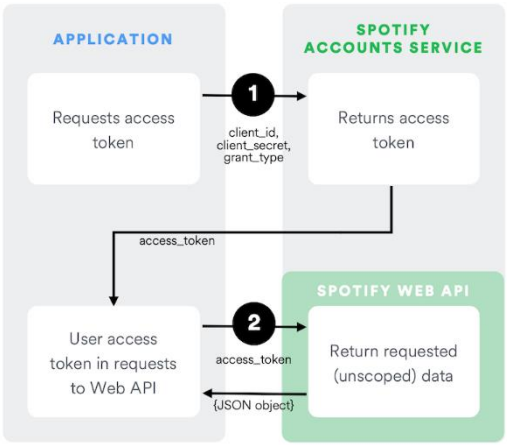


Figure 4 Client Credentials Workflow [31].

The process, illustrated in Figure 4 Client Credentials Workflow involves registering the application, obtaining client credentials (client ID and client secret) from the Spotify Developer Dashboard, requesting an access token using the client credentials, and utilizing the access token to make authenticated requests to the Spotify API.

Authorization Code Flow: this method is employed when an application needs to access user-specific data, such as a user's playlists or recently played tracks.

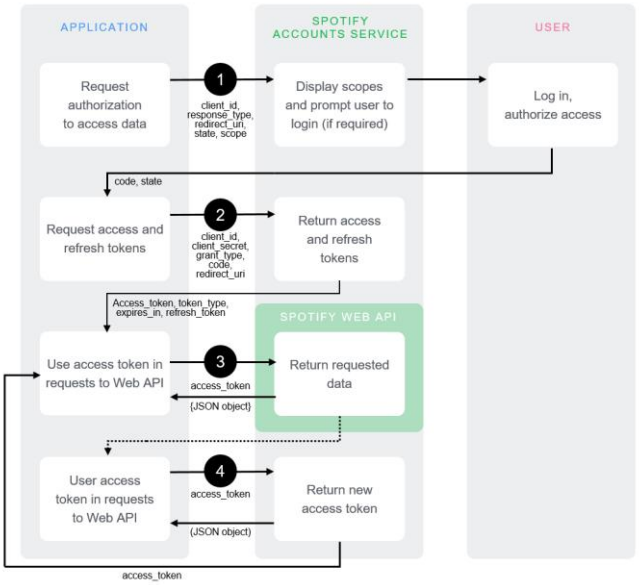


*Figure 5. A*uthorization Code Workflow [31].

As shown in Figure 5, it involves a multi-step process:

1. Application registration: The developer registers the application on the Spotify Developer Dashboard to obtain the necessary credentials, including a client ID and client secret.

2. User authorization: When a user wants to grant the application access to their Spotify data, they are redirected to the Spotify Accounts service authorization endpoint. At this endpoint, the user logs in and authorizes the application to access their data by specifying the required scopes (permissions). Once the user grants consent, they are redirected back to the application with an authorization code.

3. Token exchange: The application exchanges the authorization code received from the user for an access token. This is done by making a POST request to the Spotify Accounts service token endpoint, providing the client credentials, authorization code, and redirect URI.

4. Access token and refresh token: Upon a successful token exchange, the application receives an access token and a refresh token. The access token is used to authenticate requests to the Spotify API on behalf of the user. The refresh token allows the application to obtain new access tokens when the current one expires.

5. Accessing the Spotify API: With the access token, the application can make authenticated requests to the Spotify API, accessing the user's specific data based on the granted scopes. The access token is included in the authorization header of each request.

6. Refreshing access tokens: To ensure uninterrupted access to the user's data, the application can use the refresh token to obtain new access tokens without requiring the user's authorization again. This enables seamless and continuous interaction with the Spotify API.

This flow provides a comprehensive approach for applications to interact with the Spotify API on behalf of users, allowing access to playlists, tracks, and personalized information. In contrast, the Client Credentials Flow offers a simpler method for accessing general resources provided by Spotify without user-specific authorization.

Implementation of the Authorization Code Flow [32]*:* the *SpotifyOAuth* class *provides an implementation of the Authorization Code Flow*. The class *SpotifyOAuth* is a part of the Spotipy library, which is a Python client for the Spotify Web API. It simplifies the process of obtaining authorization codes, exchanging them for access tokens, and refreshing tokens when needed.

3. Methodology and equipment (o experimental)

3.1. Tools, Libraries, and packages

3.1.1. Code Editor

The process of accessing Spotify's Web API, retrieving songs data, and creating a comprehensive output was done entirely through Visual Studio Code. This package provides an environment for writing and executing Python code. Most of the work of this project has been developed in Notebooks, an integrated tool of Visual Studio Code that allows executing code in blocks. Notebooks also support interactive visualization of data; this helps to explore the data quickly.

3.1.2. Libraries and packages

Libraries are collections of reusable code written by developers, which can be used in multiple projects.

This section lists the libraries and packages imported in each of the notebooks that make up the project.

Python Libraries:

- Spotipy: The Spotipy library is imported to access the Spotify API and perform operations such as retrieving playlists, tracks, and audio features.
- Pandas: The Pandas library is imported as pd to work with data in tabular form, allowing data manipulation and analysis.
- Requests: The Requests library is a Python library used for making HTTP requests to web servers, allowing easy interaction with web APIs and retrieve data from remote servers. In this project, it is used for sending GET, POST, and other types of requests, as well as handling responses.
- Base64: This library is useful for encoding and decoding data in various applications, such as encoding images for use in data URLs or handling authentication credentials. In this project, the encoding module is used to encode the credentials used when authenticating to the Spotify API.
- Numpy: The Numpy library, imported as np, is fundamental for numerical operations in Python. It offers support for arrays (including mathematical operations on them) and is particularly useful for linear algebra, statistical operations, and mathematical simulations.

- Seaborn: Seaborn, imported as sns, is a data visualization library that works on top of Matplotlib. It provides a higher-level interface and aesthetically pleasing graphics for statistical plotting.
- Scikit-learn: Various modules from Scikit-learn are used in this project. They include:
    - *TSNE*: For visualizing high-dimensional data by reducing it to two or three dimensions.
    - *DBSCAN & KMeans*: For clustering the data into groups based on similarity.
    - *NearestNeighbors*: To find the nearest data points in a dataset.
    - *StandardScaler*: To normalize and scale the dataset.
    - *Silhouette_score & davies_bouldin_score*: For evaluating the clustering quality.
- Matplotlib: The Matplotlib library is used for creating visualizations and plots. It is a comprehensive library that offers various plotting styles and utilities.
- Scipy: Specifically, the *cdist* function from the *scipy.spatial.distance* module is imported to compute distance between two sets of points.

Python modules:

- Os: The Os module is imported to interact with the operating system and retrieve environment variables.
- Dotenv: The *load_dotenv* function from the *dotenv* module is imported to load environment variables from a .env file.
- Spotipy.oauth2: The SpotifyOAuth class from the *spotipy.oauth2* module is imported to authenticate the script with Spotify using OAuth 2.0.
- Random: The Random module is imported to generate random numbers, which are used to introduce a random delay before making API requests.
- Time: The Time module is imported to include time-related functionality in the script. In this case, it is used with *time.sleep()* to introduce delays and avoid the API rate limit.

3.2. Data Collection

This project has three main objectives: (1) analyzing the evolution of music features throughout the past decades (2) determine if the relation between features and popularity over the years has changed, and (3) building a content-based recommendation engine (see Section 1.4). All the objectives can be achieved with the same dataset, since the evolution of music throughout the last decades requires obtaining the necessary features for building a content-based recommendation engine.

To meet these objectives, the project requires the following:

- Large amount of data.
- Wide variety of genres, artists, popularity, and years.
- Music from the past 100 years, evenly distributed.
- In addition to audio features, the release date of each track is required.

There are various datasets existing in the internet. However, none of them meets the characteristics needed in this project. For this reason, by making requests to the Spotify API, a dataset was created exclusively for this project.

3.2.1. Source of Data

Spotify has APIs that provide access to user, artist, and playlist data for external developers.

For authentication and authorization, the Spotify Authorization Code Flow (see Section 2.4.7.2) was implemented.

The process started by registering a "Spotify Developer" account, where I created a web application[16] in the developer dashboard, shown in below in Figure 6.

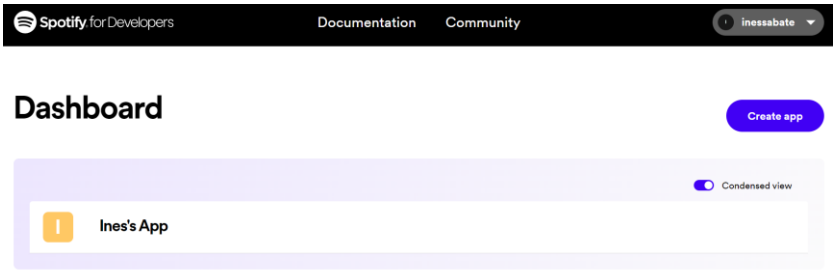


Figure 6. Application created to request authorization to access data.

The app provided the credentials needed to implement the Authorization Code Flow:

- Client ID: the unique identifier of my app.
- Client Secret: the key used to authorize Web API calls.

For security purposes, these credentials were stored in a .txt file and defined as environmental variable, accessed within the code as shown in Figure 9.

[16] "A web application is a computer program that utilizes web browsers and web technology to perform tasks over the Internet."

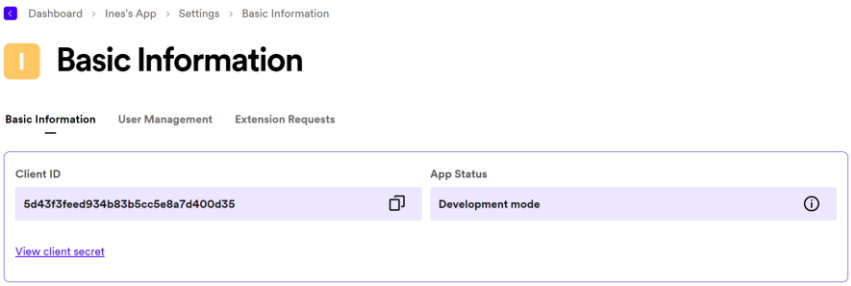


Figure 7. Credentials provided by the application.

Once authenticated, I had access to the Spotify API and began retrieving songs data through individual endpoints[17] of the Spotify API. As *Figure 8* suggests, accessing the Spotify's Web API and retrieving song's data was done through Visual Studio Code.

Figure 8. Architecture of requests from the web app to the APIs. [Source: own elaboration]

To interact with the Spotify Developer API, I used a Python Script that imported the Spotipy library. This library made it easy to search through my Spotify account's browser and retrieve playlists based on my specific search criteria (the search criteria is described in chapter 3.2.2.2).

The code snippet shown in Figure 9Figure 9 is used for authentication and authorization with the Spotify Developer API.

[17] Endpoints are specific URLs that provide access to several types of data. As mentioned in [33], "endpoints specify where APIs can access resources and help guarantee the proper functioning of the incorporated software."

```
# Calling the `load_dotenv()` function to load environment variables from a .env file.
load_dotenv()

# Calling `os.getenv()` function to retrieve values from environment variables.
client_id = os.getenv("client_ID")
client_secret = os.getenv('client_secret')
#redirect_uri = os.getenv('my_redirect')
username=os.getenv('myUser')
redirect_uri = 'https://ines_dashboard.com/callback'


# Creating an instance of the `SpotifyOAuth` class from the `spotipy` library.
sp = spotipy.Spotify(auth_manager=SpotifyOAuth(client_id=client_id,
                                               username=username,
                                               client_secret=client_secret,
                                               redirect_uri=redirect_uri, scope=None))
```

Figure 9. Connection to my Spotify account.

It begins by loading the credentials provided by the application (defined as environment variables for safety) using the *load_dotenv()* function. Then, it retrieves the values of specific environment variables using the *os.getenv()* function and assigns them to variables *client_id, client_secret, username, and redirect_uri.* Finally, it creates an instance of the SpotifyOAuth class (see Section 2.4.7.2) from the *spotipy* library, passing in the retrieved environment variable values as arguments.

3.2.2. Feature Extraction

The purpose of this section was to create a dataset containing music from a variety of genres released between 1900 and 2023.

3.2.2.1. Functions

The Python functions described below were developed to search playlists, obtain track features, and store data in comma-separated files.

Get_playlists_by_genre(genre, limit):

Retrieves playlists from the Spotify API based on a given genre and limit.

I. Parameters:
   a. genre (str): The genre of the playlists to search for.
   b. limit (int): The maximum number of playlists to retrieve.

II. Returns:
   a. A tuple containing two lists - id_list and username.
   b. id_list (list): A list of playlist IDs.
   c. username (list): A list of usernames associated with the playlists.

Get_tracks(id, user):

Retrieves tracks from playlists associated with a given playlist ID and user.

I. Parameters:
   a. id (str): The playlist ID.
   b. user (list): A list of usernames associated with the playlists.
II. Returns:
   a. pandas.DataFrame: A DataFrame containing the combined tracks from all playlists.

New_funct(df_tot, df):

Concatenates two DataFrames using the pd.concat() function.

I. Parameters:
   a. df_tot (pandas.DataFrame): The DataFrame to which the other DataFrame will be concatenated.
   b. df (pandas.DataFrame): The DataFrame to be concatenated.
II. Returns:
   a. pandas.DataFrame: The concatenated DataFrame containing features from all the songs in the given playlists.

Get_playlist_tracks(username, playlist_id_list):

Retrieves tracks and their audio features from playlists associated with a given username and playlist ID list.

I. Parameters:
   a. username (str): The username associated with the playlists.
   b. playlist_id_list (list): A list of playlist IDs.
II. Returns:
   a. pandas.DataFrame: A DataFrame containing tracks and their audio features from the playlists.

3.2.2.2. Methodology

The functions described in *Section 3.2.2.1* were used to search for a specific word (in my case, music genres and subgenres) in the Spotify browser and get data from the first n playlists that match the search. The function *get_playlists_by_genre* returns a list with playlist ID and username for each search word. Then, function *get_tracks* takes those values and retrieves the features for all the songs in the playlists specified in the list of IDs and saves them into comma-separated files.

Searching different subgenres of genres, allowed me to create twelve genre-specific datasets. The subgenres and words parsed into the function *get_playlists_by_genre* to build each dataset are listed in Table 4.

Table 4. Subgenres searched to create genre-specific datasets.

Genre	Subgenres		
Rock	Alternative Rock	College Rock	Indie
	Grunge	Experimental Rock	Progressive Rock
	New Wave	Hardcore Punk	
Anime	Anime		
Blues	Acoustic Blues	Contemporary Blues	Blues Origins
	Chicago Blues	Delta Blues	Electric Blues
	Classic Blues		
Classical	Classical Crossover	Classical Essentials	Impressionist
	Early Music	Modern Composition	Medieval
	Opera		
Dance Music	Electronic	Dubstep	Techno
	Trance	Minimal Techno	Breakbit Classics
	House	Dance music 2023	
Rap/Hip Hop	Alternative Rap	Hip Hop Classics	East Coast Rap
	Bounce	Hardcore Rap	West Coast Rap
	Dirty South	Hip Hop	
Electronic	Ambient	Electronica	Downtempo
	Crunk	Industrial	Electro Disco
Country	Alternative Country	Urban Country	Traditional Bluegrass
	Americana	Country Gospel	Honky Tonk
	Bluegrass	Outlaw Country	Urban cowboy
	Western swing	Nashville sound	
Pop	Art pop	Art pop	Sad pop
	Brill Building	Brill Building	Schlager
	Bubblegum	Bubblegum	Sophisti-pop
	Cringe pop	Cringe pop	Teen pop
	K pop	Wonky pop	
Jazz	Smooth Jazz	Bebop Jazz	Traditional Jazz
	Fusion Jazz	Cool Jazz	Swing Jazz
	Latin Jazz	Free Jazz	Vocal Jazz
	Gypsy Jazz		
Latin	Reggaeton	Latin Pop	Latin Ballads
	Bachata	Ranchera	Latin Urban
	Salsa	Flamenco	Latin Rock
	Cumbia	Tango	Reggae en Español
	Merengue	Banda	Latin Jazz

R&B / Soul	Contemporary R&B	Motown	Doo Wop
	Neo-Soul	Quiet Storm	Hip-Hop Soul
	Funk	Funk/Soul Revival	Disco
	Soul	New Jack Swing	Trap&B
	Alternative R&B	Gospel R&B	

To build a dataset containing music from as many genres as possible, I used the function *get_playlists_by_genre(genre, limit)*. I obtained relatively small datasets changing the *genre* argument to the values in the column *Subgenres* of Table 4. This method allowed me to create a data frame for each genre.

Genre: the Spotify API does not provide any genre with the songs, since music genres are associated to albums and artists, not specific songs. In order to associate each track to a musical genre, the artist id or album id of the track must be retrieved. In this project, the data regarding artists and albums was not addressed.

Additional data: during the data understanding phase (see section 3.3.1), I noticed the size of the datasets after removing duplicate samples was significantly reduced. This highlighted the necessity to gather more data, which was accomplished by directly using the 'get_playlist_tracks(username, playlist_id_list)' function. For this purpose, five playlists for each genre, all created by the user 'spotify', were retrieved.

The reason why the data was not gathered using the *function 'get_playlist_tracks'* in the first place was that it requires providing a username. As I intended to gather data in the most random way possible, I preferred searching for random users through the function *'get_playlists_by_genre('genre', limit)'.*

3.2.2.3. Features

The features extracted by the function get_playlist_tracks(username, playlist_id_list) are described in Table 5.

Table 5. Description of features [31].

Feature	Description
Id	The Spotify ID of the track.
Title	The title of the track.
All artists	A list of all the artists who performed the track.

Popularity	The popularity of the track on Spotify, ranging from 0 to 100.
Release date	The release date of the track.
Danceability	Represents the suitability of the track for dancing based on a combination of musical elements, including tempo, rhythm stability, beat strength, and overall regularity. Ranges from 0.0 (not danceable) to 1.0 (highly danceable).
Energy	Represents a perceptual measure of intensity and activity in the track. Higher energy values typically indicate fast, loud, and noisy tracks. Ranges from 0.0 to 1.0.
Key	Represents the key of the track. The values correspond to the pitch classes (0 = C, 1 = C♯/D♭, 2 = D, and so on).
Loudness	Represents the overall loudness of the track in decibels (dB). Negative values indicate quieter tracks, and positive values indicate louder tracks.
Mode	Represents the modality of the track (major = 1, minor = 0).
Acousticness	Represents the acoustic quality of the track. A value of 1.0 indicates a high probability that the track is acoustic, while a value of 0.0 indicates a low probability.
Instrumentalness	Represents the likelihood that a track is instrumental. A value close to 1.0 indicates a high probability of the track being instrumental, while a value close to 0.0 suggests vocals are present.
Liveness	Represents the probability of the track being performed live. A value close to 1.0 indicates a high probability of the track being performed live, while a value close to 0.0 suggests a studio recording.
Valence	Represents the musical positiveness conveyed by a track. Higher valence values represent more positive and cheerful tracks, while lower values represent more negative and sad tracks. Ranges from 0.0 to 1.0.
Tempo	Represents the overall estimated tempo of the track in beats per minute (BPM).
Duration [ms]	The duration of the track in milliseconds.

Time signature	Represents the estimated number of beats per bar (measure) in a track. “The time signature ranges from 3 to 7 indicating time signatures of "3/4", to "7/4" ” [33].

The latter actions were performed in a single notebook. The notebook is structured into four sections: import of libraries and modules (see section 3.1.2), connection to Spotify (Read Section 2.4.7.2), definition of functions (see section 3.2.2.1), and creation of datasets for each genre (see section 3.2.2.2).

3.3. Data Understanding

This section aims to provide a comprehensive description of the available data.

3.3.1. Description of the data

3.3.1.1. Dimensions and data types

Quantity of data: the dataset is comprised of twelve genre-specific datasets. Each of them contains 17 columns[18], while the number of samples (tracks) in each dataset varies. The latter – before and after removing duplicate rows – is shown in Table 6.

Table 6. Number of samples in each dataset.

Dataset	Before removing duplicates	After removing duplicates
df_anime.csv	139	139
df_blues.csv	399	374
df_classic.csv	9715	4897
df_dance_music.csv	7885	2686
df_electronic.csv	12203	3615
df_jazz.csv	8573	4014
df_latino.csv	9339	4384
df_pop.csv	4957	4847
df_rap.csv	9805	3265
df_RB_soul.csv	8309	3716
df_country	9951	3023
df_additionl_data	31023	27861

The datasets shown in Table 6 were combined into a unified dataset. As a result of this operation, the dataset contains 71785 unique records, and 17 columns.

[18] All datasets contain the columns*: id, title, all_artists, popularity, release_date, danceability, energy, key, loudness, mode, acousticness, instrumentalness, liveness, valence, tempo, duration_ms, and time_signature.*

Data types: all datasets share identical features, which include: *id, title, all_artists, popularity, release_date, danceability, energy, key, loudness, mode, acousticness, instrumentalness, liveness, valence, tempo, duration_ms, and time_signature*. The data types of the features, shown in Table 6 were consistent in all raw datasets.

Table 7. Data types in the raw datasets.

Feature	Data Type
id	object
title	object
all_artists	object
popularity	float64
release_date	object
danceability	float64
energy	float64
key	float64
loudness	float64
mode	float64
acousticness	float64
instrumentalness	float64
liveness	float64
valence	float64
tempo	float64
duration_ms	float64
time_signature	float64
release_year	int32[19]

The format of the values in the 'release_date' column, which is present in all the datasets, exhibited inconsistency. Some of the values in the 'release_date' column included complete dates, comprising the day, month, and year, while others contained only the year. The complete dates utilized characters '/' and '-' as separators. For example, a song released on November 1, 1964, could appear as '1964-11-01', '1964/11/01', or simply as '1964'.

This was an issue and was corrected at this phase of the project, as it impeded including release dates in the exploratory analysis.

To address this problem, a new column was introduced in all datasets, specifically for the release year (*release_year*) of the songs (its data type is presented in Table 7). Consequently, the column *release_date* was disregarded, as its information became irrelevant after the inclusion of the new release year column.

Null Values: the number of null values in each of the columns of the datasets is shown in Table 8.

[19] The '*release_year'* column was converted to integer type to facilitate analysis in chapter 3.3.3.

Table 8. Number of null values in each raw dataset.

DataFrame	Column	Count
df_classic.csv	all_artists	74
df_classic.csv	popularity	148
df_classic.csv	danceability	148
df_classic.csv	energy	148
df_classic.csv	key	148
df_classic.csv	loudness	148
df_classic.csv	mode	148
df_classic.csv	acousticness	148
df_classic.csv	instrumentalness	148
df_classic.csv	liveness	148
df_classic.csv	valence	148
df_classic.csv	tempo	148
df_classic.csv	duration_ms	148
df_classic.csv	time_signature	148
df_latino.csv	title	9
df_latino.csv	all_artists	9
df_rap.csv	title	3
df_rap.csv	all_artists	3
df_RB_soul.csv	title	1
df_RB_soul.csv	all_artists	1

The dataset containing classical music features had, by far, the highest number of missing values. Null values were present in 14 out of 17 columns in this dataset.

Although the number of empty rows (148) is relatively low compared to the total number of rows in the classical dataset (9715 tracks), these rows with null values span across almost all columns (148 rows with empty values in 14 columns). This information led to decide to drop the rows containing the missing values during the data preparation stage of the project (see section 3.3.3.1).

On the other hand, the remaining datasets in Table 8 contain fewer empty values, which are only present in the *'title'* and *'all_artists'* columns. As these columns might not be relevant for further analysis, this suggested to ignore them instead of dropping the entire rows. However, dropping the rows was not a problem either, given the substantial amount of data available in the datasets, shown in Table 6.

3.3.1.2. Descriptive Statistics

Descriptives statistics of all the numerical features in the dataset [20] are shown in Table 9.

[20] Duplicate records and rows containing null values have been removed prior to calculating descriptive statistics.

Table 9. Descriptive statistics of the dataset.

Feature	count	mean	std	min	25%	50%	75%	max
Popularity	71785	33.495	25.950	0	3	36	55	100
Danceability	71785	0.5624	0.1772	0	0.442	0.57	0.696	0.988
Energy	71785	0.6191	0.2682	0	0.446	0.673	0.841	1
Key	71785	52.989	35.524	0	2	5	8	11
Loudness	71785	-9.285	60.503	-60	-11.27	-7.468	-5.262	4.14
Mode	71785	0.6416	0.4795	0	0	1	1	1
Acousticness	71785	0.3108	0.3418	0	0.0155	0.15	0.581	0.996
Instrumentalness	71785	0.1738	0.3193	0	0	0.00026	0.123	0.999
Liveness	71785	0.1911	0.1606	0	0.0939	0.125	0.243	1
Valence	71785	0.5107	0.2555	0	0.307	0.515	0.718	0.993
Tempo	71785	120.52	28.822	0	98.031	120.011	137.932	247.986
Duration [ms]	71785	236538	106382	4067	180173	219267	267373	4628227
Time Signature	71785	39.048	0.4037	0	4	4	4	5
Release Year	71785	2005.9	15.288	1820	1999	2010	2018	2023

A thorough descriptive analysis was conducted on each of the musical features:

- *Popularity:* The popularity metric for tracks in our dataset mostly hovers around a moderate level. The mean of 33,5, slightly lower than the median, reveals a presence of songs with considerably lower popularity scores. Such low scores tug the mean downwards, indicating a skew in the distribution of popularity. If most tracks were equally popular, the mean and median would converge. The difference between them emphasizes the diversity in song popularity within the dataset.
- *Danceability:* The average danceability score across the tracks is 0,56, suggesting a slightly leaning tendency towards tracks that are more suitable for dancing. However, with values ranging from 0 to 0,988, the dataset encompasses a vast array of tracks, from those completely non-danceable to those that strongly encourage dancing.
- *Energy*: The energy metric has an average score of 0,62. This indicates that a considerable number of tracks in the dataset possess a more energetic and intense sound. This metric can reflect genres that are more upbeat, such as dance, pop, or rock.
- *Key*: With an average value around 5,3, it showcases that various musical keys are represented in this dataset, ensuring diversity in terms of tonal characteristics.
- *Loudness:* The average loudness of the tracks sits at -9,29 dB. Tracks are spread between the quietest at -60 dB and the loudest at 4,14 dB. The presence of tracks with positive loudness values might seem unconventional but could be attributed to the way loudness is normalized or processed within the platform.
- *Mode:* A mean of 0,64 indicates a slight preference towards tracks in a major key, which is often associated with a happier and brighter sound.

- *Acousticness:* The mean score of 0,31 suggests that while a significant portion of tracks has an acoustic element to them, the dataset also comprises electronically produced or non-acoustic tracks.
- *Instrumentalness:* An average score of 0,17 shows that most of tracks likely contain vocals, as they lean away from being purely instrumental.
- *Liveness:* The mean value of 0,19 suggests that most tracks are studio recordings, with a smaller percentage being live recordings or having audience noise.
- *Valence:* A mean valence score of 0,51 reveals that the dataset has a balanced mix of tracks, with both positive (happy) and negative (sad) vibes.
- *Tempo:* The average tempo stands at 120,52 BPM, placing it in a mid-tempo range which is common across many genres. The range, however, extends to a fast 247.986 BPM, showcasing the presence of faster-paced genres or tracks.
- *Time Signature:* An average of 3,90 indicates a dominance of tracks with a 4/4 time signature, which is the most common time signature in contemporary music.
- *Release Year:* The average release year is approximately 2005,9, which reveals that the dataset leans towards more contemporary tracks. The presence of songs from as early as 1820, however, ensures historical representation.
- *Duration:* The average track length is approximately 236,5 seconds (or close to 4 minutes), which is typical for popular music tracks. The dataset, however, contains tracks as short as 4 seconds and as long as 4628 seconds (or 77 minutes), indicating the presence of both brief clips and longer compositions or mixes.

Boxplots: the following figures provide a visual representation of some descriptive metrics. There are three boxplots, since the range of the features *tempo*, *loudness*, and *popularity* is substantially different from the rest, their distribution is represented in different figures (Figure 11, Figure 12, and Figure 13 respectively).

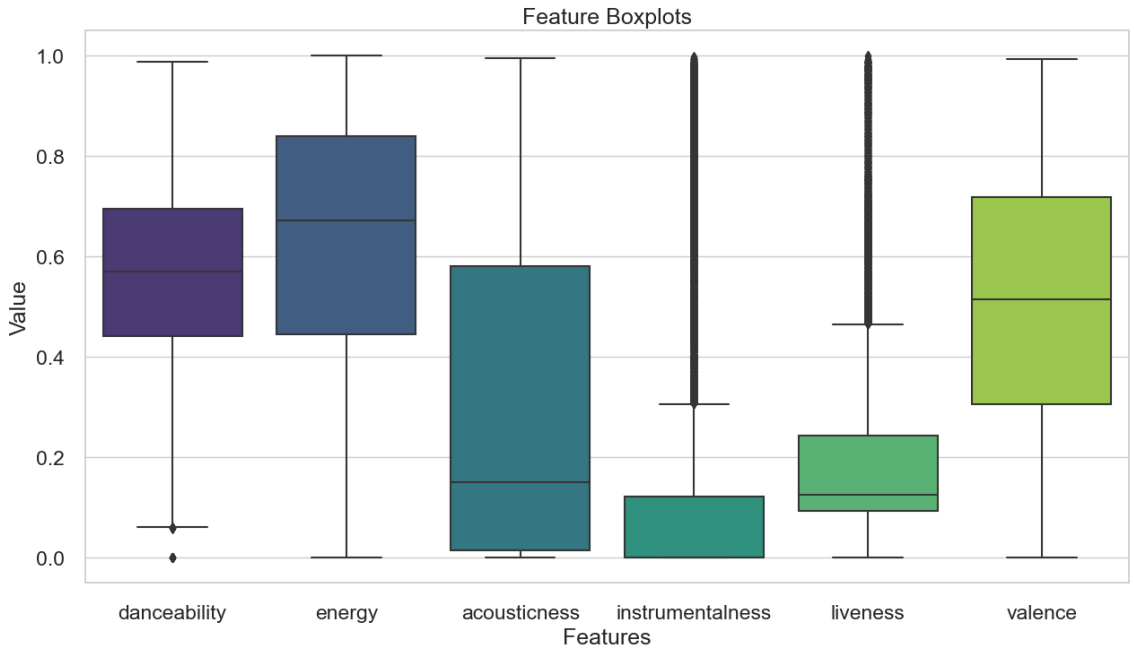


Figure 10. Boxplot of danceability, energy, acousticness, Instrumentalness, liveness, and valence.

The boxplot provided in Figure 10 consolidates several of the latter findings:

- The median *danceability* score, closely aligned with the mean of 0.56, underscores the dataset's general inclination towards tracks suitable for dancing.
- *Energy's* higher median value reiterates our assessment of the dataset having a predominantly upbeat character.
- *Acousticness*, with a median considerably lower than its maximum, underlines the blend of both acoustic and electronic elements in our tracks.
- The median for *instrumentalness* is notably low. This fact reaffirms that a majority of the tracks are not purely instrumental.
- *Liveness* scores lie below 0.2, confirming the collection's bias towards studio-recorded tracks over live performances.
- *Valence's* median (positioned around the 0.51 mark), validates our understanding of the dataset's balanced emotional spectrum, containing both uplifting and melancholic tracks.
- The length of the whiskers is consistent with the values of STD shown in Table 9. For instance, danceability, which has the smallest standard deviation of 0.1772, is distinctly represented in the boxplot with the most compact whiskers, underscoring its limited variability.

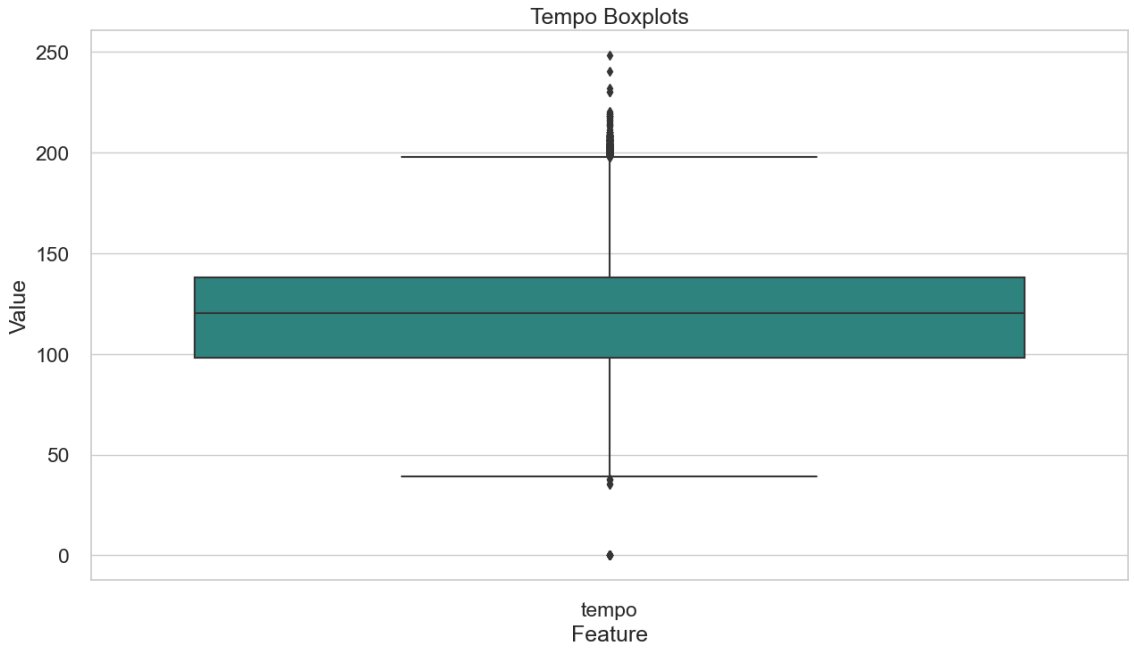


Figure 11. Boxplot of tempo.

The *tempo* boxplot (see *Figure 11*) confirms the prior observations. The central tendency of the distribution, hovering around the mean of 120.52 BPM, reflects the popular mid-tempo range present in numerous genres (given the fact that the dataset contains various genres). The spread of the data, however, highlights the varied tempo of tracks within the various genres of the dataset.

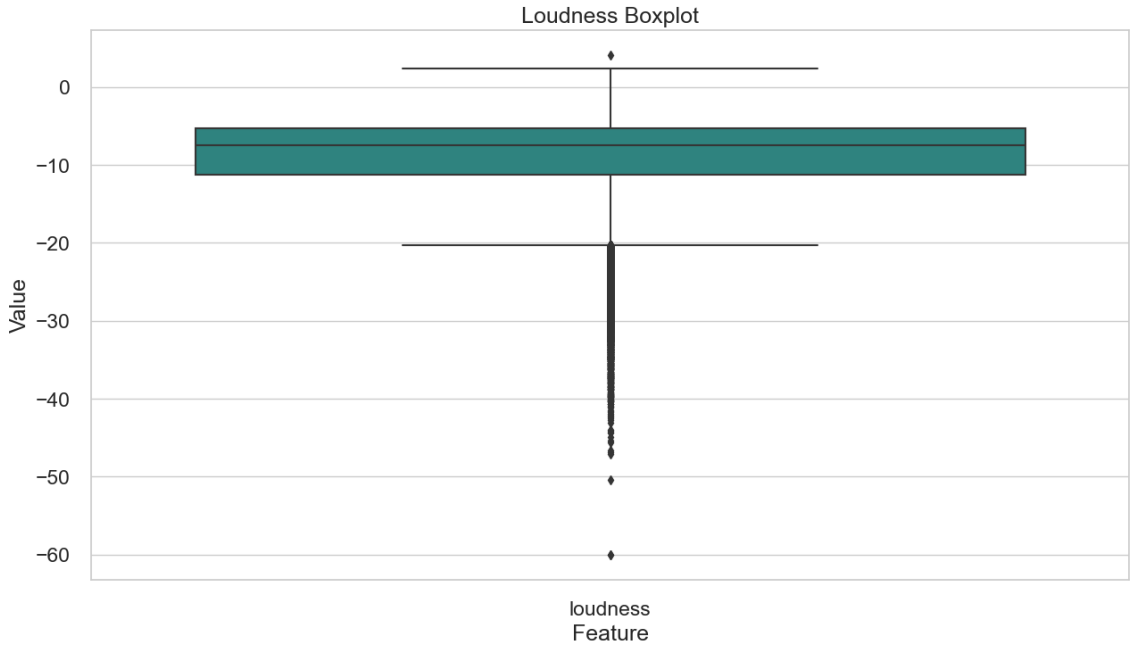


Figure 12. Boxplot of loudness.

The *loudness* visualization (see *Figure 12*) reveals a comprehensive representation of volume levels in our tracks. The central location of the median closer to the -9.29 dB mean endorses the previous findings. It also provides a graphical affirmation of the extensive range of loudness values in the dataset, from particularly soft tracks to exceptionally loud ones.

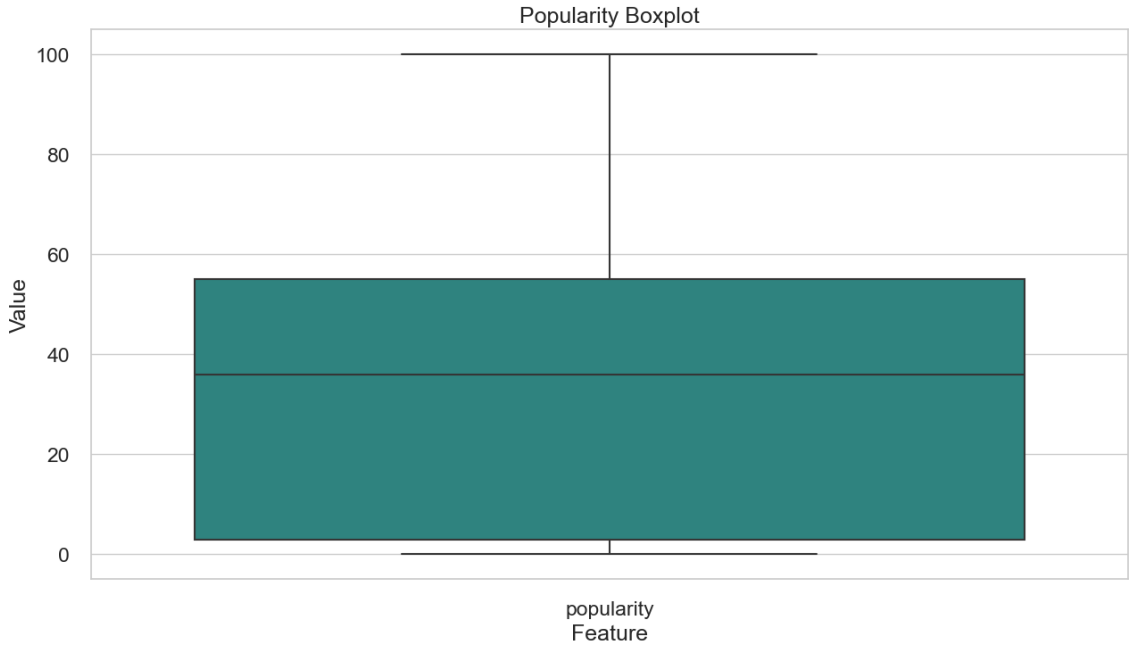


Figure 13. Boxplot of popularity.

The boxplot dedicated to popularity (see *Figure 13*) offers a visual corroboration of the latter observations. The slight discrepancy between the median (36.0) and the mean (33.5) reinforces the skewed nature of track popularity. This graphical representation underscores the diversity of the dataset, where there are many tracks with lower popularity scores, as seen in *Table 9*.

3.3.1.3. Observations

The fact that most of features have a considerable standard deviation suggests the dataset presents a comprehensive overview of various musical attributes from tracks of diverse genres, release years, and characteristics. The slight skews in certain attributes like popularity offer insights into the nature of the dataset and the potential biases or preferences in the data collection or the platform (Spotify) itself.

Because this dataset was manually constructed based on genres, it is important that it contains a wide variety of artists.

Given that the dataset was curated manually based on genres, it is crucial that it contains a wide variety of artists. Artists are typically consistent in their style and genres, and a limited roster might introduce bias, potentially complicating the cluster formation process.

The dataset is comprised by 24793 unique artists and 68899 unique tracks. A search for prominent artists names based on popularity, revealed names such as Adele, Bad Bunny and Coldplay, which makes sense given their current relevance in the music industry. In *Table 10,* the top ten artists contributing the most songs to the dataset are displayed. Notably, none of these artists shown in the table account for more than 0.5% of the total dataset, which guarantees that the dataset contains a wide variety of genres.

Table 10. Top ten artists contributing to the dataset.

Artist	Number of Tracks
María Luisa Piraquive	402
Paul O'Dette	258
Megadeth	220
Ramones	191
Hespèrion XXI	178
Metallica	171
Julian Bream	168
Red Hot Chili Peppers	161
Aerosmith	155
Nirvana	154

3.3.2. Data Quality verification

The performance of a machine learning algorithm heavily relies on two crucial factors: the quality of the data and the abundance of relevant information within it. (To gain insights on data quality, refer to Chapter 2.3)

Table 11. Valid values for each feature.

Feature	Valid Valeus
Id	String of 22 characters
Title	String
All artists	String
Popularity	Numerical value. Ranges from 0 to 100.
Danceability	Numerical value. Ranges from 0.0 to 1.0
Energy	Numerical value. Ranges from 0.0 to 1.0
Key	Numerical value. Ranges from 0 to 11.

Loudness	Numerical value.
Mode	Numerical value. Valid values are 0 and 1.
Acousticness	Numerical value. Ranges from 0.0 to 1.0
Instrumentalness	Numerical value. Ranges from 0.0 to 1.0
Liveness	Numerical value. Ranges from 0.0 to 1.0
Valence	Numerical value. Ranges from 0.0 to 1.0
Tempo	Numerical value (positive)
Duration (in ms)	Numerical value (positive)
Time Signature	Numerical value (positive)
Release Year	Numerical integer. Ranges from 0 to 2023.

According to the minimum and maximum values displayed in Table 11, all features are in the permitted ranges.

As specified in section 3.3.1, the dataset does not contain duplicate rows or null values, since these were removed previously.

3.3.3. Exploratory Data Analysis (EDA)

3.3.3.1. Correlation of features

The relationships among features is examined in order to identify potential associations or interdependencies. This is especially relevant within machine learning projects as it aids in the process of feature selection. Highly correlated features might encompass redundant information, and excluding them can lead to simpler (and more precise) models.

The diagrams presented below consist of a series of scatterplots and histograms formulated based on the subsequent considerations:

- A random selection of 1000 songs from the dataset was utilized to enhance visual clarity (n = 1000).
- Features containing the track's identifier, title, and artist were excluded from the correlation analysis of variables, given their lack of relevance.

- In the pair plots, a specific parameter[21] was configured to correspond to the variable 'popularity'. This allowed a visual distinction within the plots, allowing for the differentiation of data points base don their respective popularity levels.

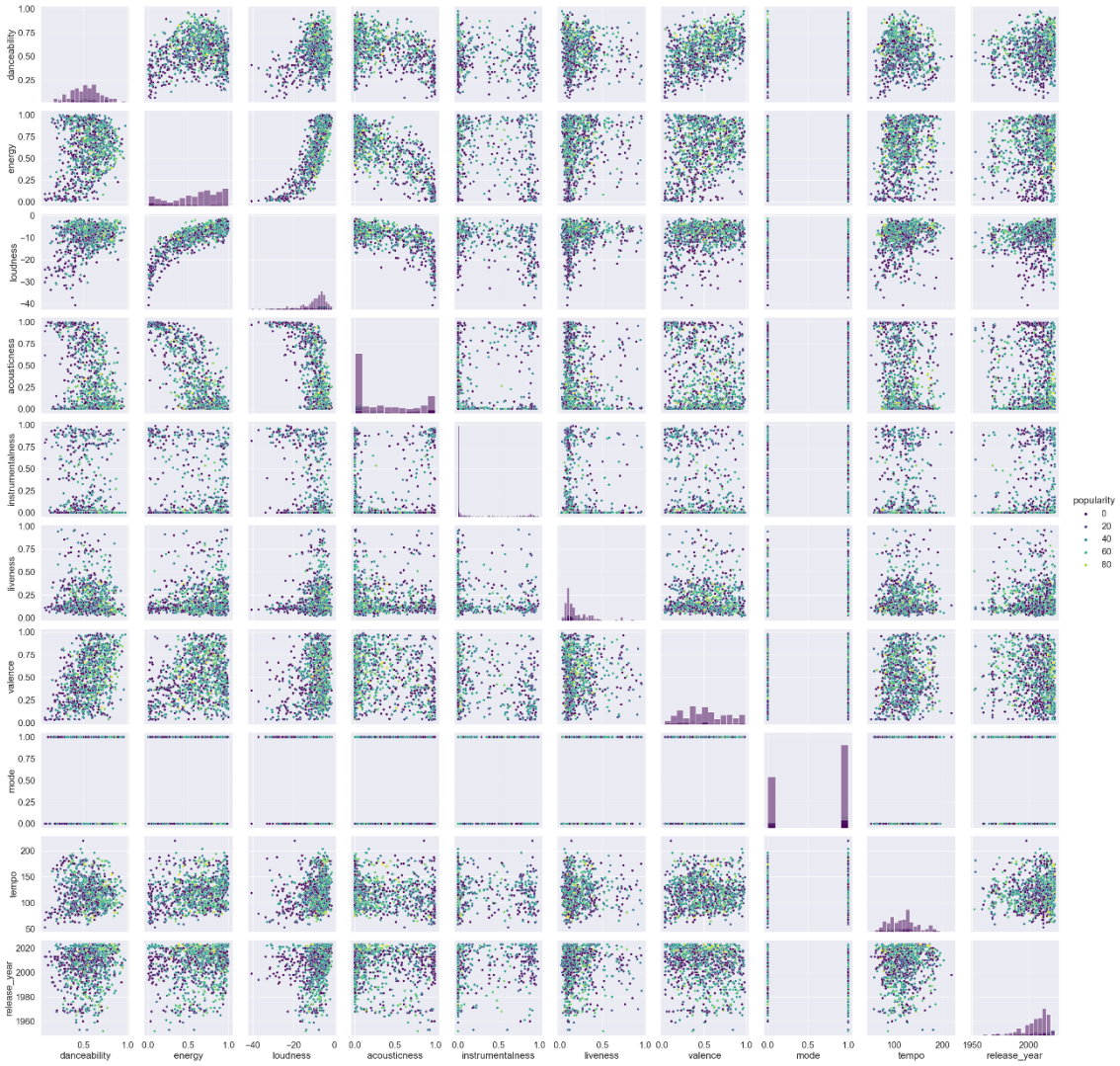


Figure 14. Scatter plot and histograms of the features.

The plot in Figure 14 allowed to identify linear and non-linear relations between pairs of features (in the scatter plots), and the distribution of each feature (in the histogram).

Regarding the distribution of variables, the histograms indicated that most features in the dataset were unevenly distributed. However, *'valence'* and *'energy'* had a more balanced

[21] The library used to visualize the grids is Seaborn, which allows the configuration of parameters to customize its graphics. More information about the seaborn library and its attributes is available at: https://seaborn.pydata.org/generated/seaborn.pairplot.html

distribution. The rest of observations regarding the distribution of features confirm the observations commented in section *3.3.1.2 Descriptive Statistics.*

The most significant correlation exists between the features energy and loudness, whose nature is positive and exponential. Danceability seems to have a slight positive correlation with valence, which is positively correlated with energy and loudness as well. This is consistent with the idea that songs with higher 'valence' values, indicating happiness, tend to be more energetic, louder, and suitable for dancing.

For the 'popularity' layer, two supplementary plots (*Figure 15,* and *Figure 16*) were created to enhance visualization. The first graph showcases songs that fall within the top 25% in terms of popularity (third quartile). In contrast, the second graph displays a selection of songs from the bottom 25% (first quartile). Both graphs consist of a sample of 1,000 songs. To simplify visualization, the features 'mode' and 'release year' were omitted.

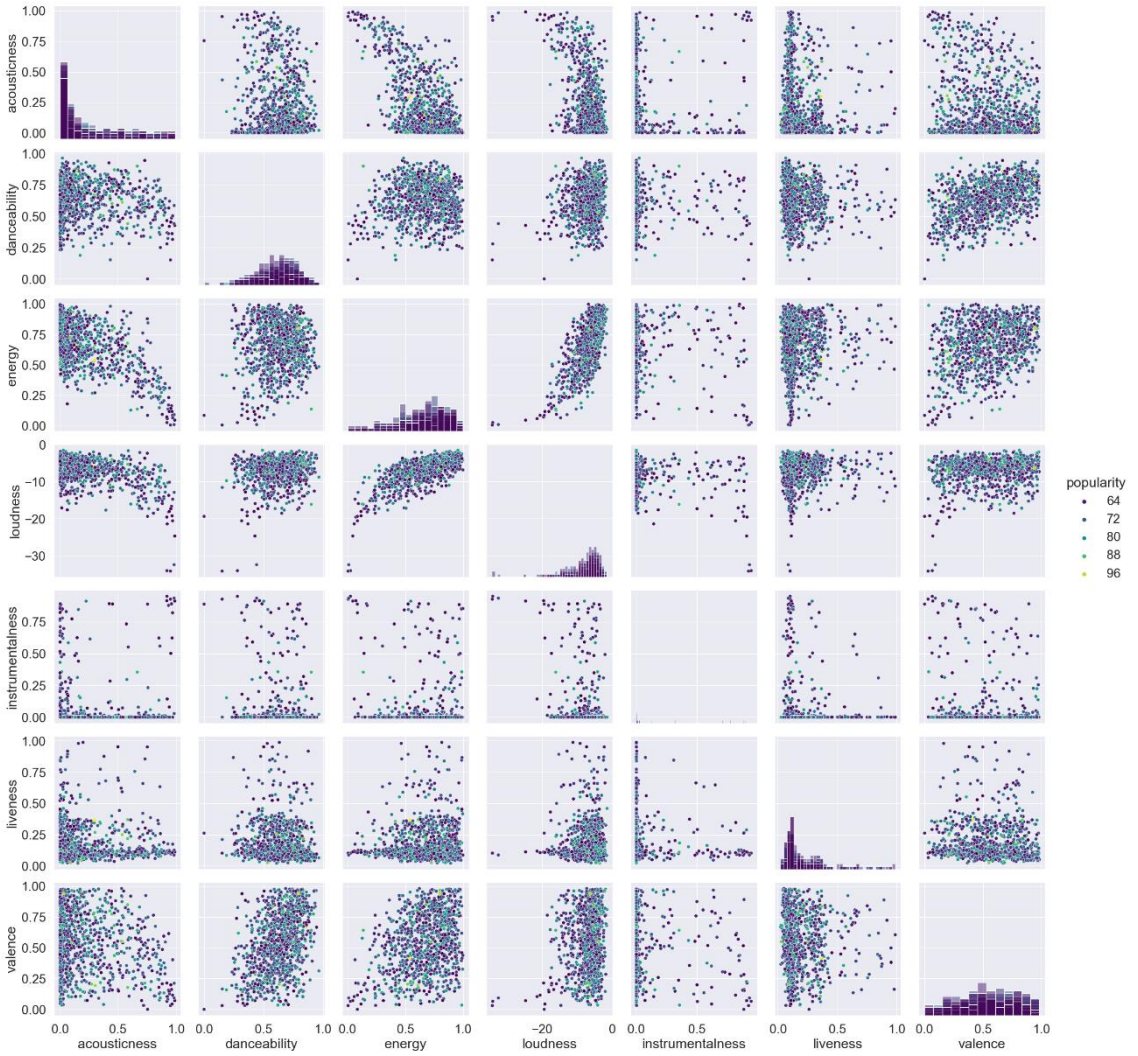


Figure 15. Scatter plot and histograms of the most popular songs.

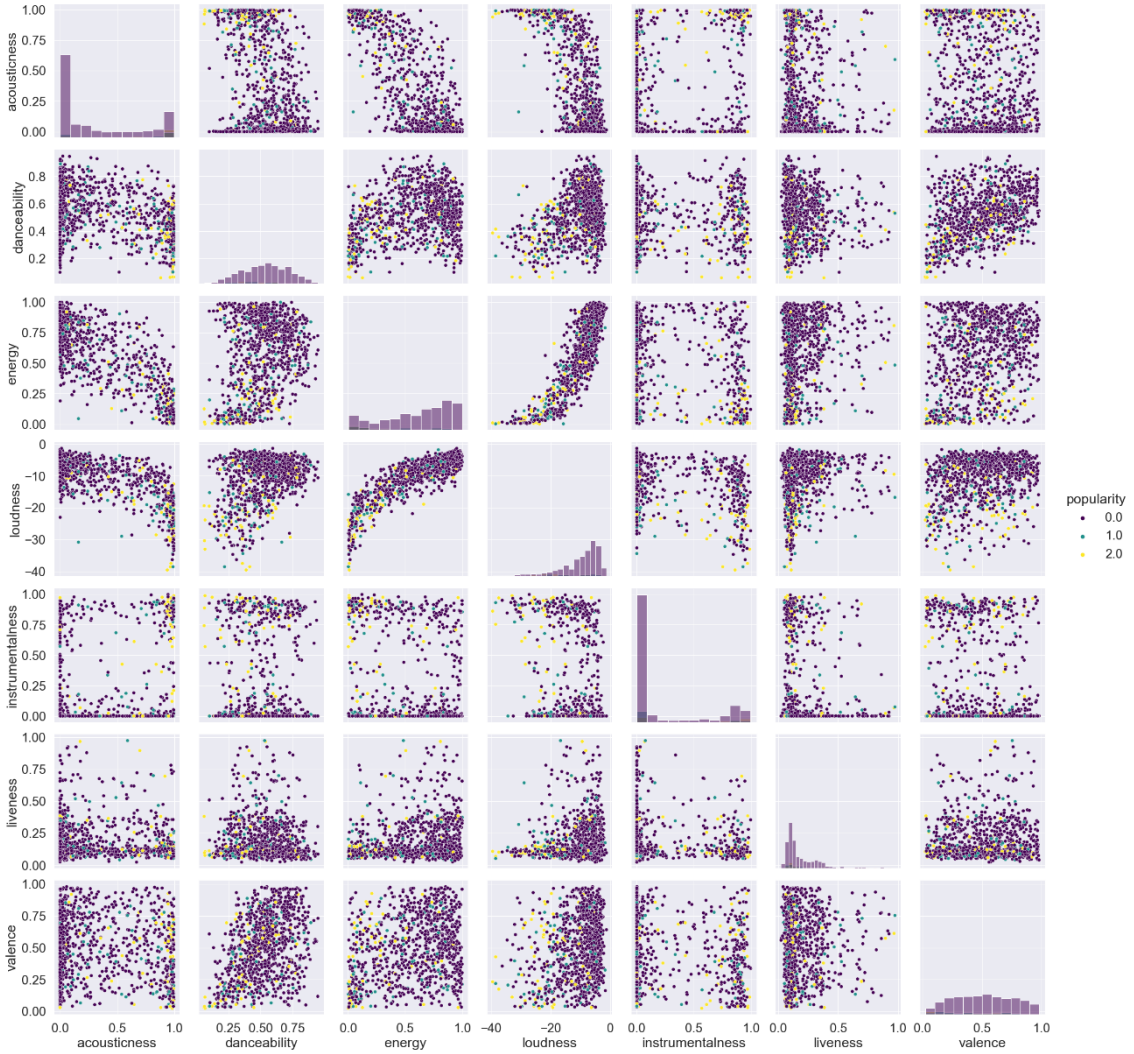


Figure 16. Scatter plot and histograms of the least popular songs.

Distribution: the histograms of the features in the above figures show similar distributions in almost all features except for the feature *Instrumentalness*, which appears to be almost null in the most popular songs of the dataset.

Sparsity observations: in examining the scatterplots, the relationships between features are similar, but there is a notable sparsity observed for the least popular tracks (see *Figure 16)* in comparison to the more popular ones (see *Figure 15*). Several factors could contribute to this observation:

- *Nature of Popularity Distribution:* song popularity (or any item's popularity) does not always exhibit a uniform distribution. Distributions usually show a pronounced skew, with a large pool of tracks having low popularity scores, and

a select few achieving high popularity. Such skewed distribution can lead to a sparsity when visualized (see section 1.1.1).

- *Variability in Features:* The least popular songs might exhibit greater variability in their features in comparison to the top songs in popularity. This can result from the former not necessarily adhering to the typical characteristics seen in mainstream hits, leading to a wider spread in their feature values when plotted.
- *Heterogeneity of Less Popular Songs:* The least popular category might encompass a vast range of tracks, such as experimental or niche genres. Such diversity is manifested as increased sparsity in visual representations. The fact that least popular songs show sparser scatter plot makes sense, as unpopular tracks inherently deviate significantly from mainstream musical trends.
- *Sampling Bias:* the lack of true randomness in the sampling process can lead to sparse patterns in visual depictions. However, the sample of 1000 songs from the least popular songs has been obtained randomly through a Python method that generates random samples given the desired sample size [22]. In addiction the dataset employed in this project was constructed by assembling tracks from diverse random playlists and exploring an extensive array of genres and subgenres. This approach enhances the randomness of the dataset, minimizing the potential for bias and exaggerated sparsity in visual representations.

[22] More information about the *pandas.DataFrame.sample(n)* method can be found at: *https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sample.html*

3.3.3.2. Evolution over time

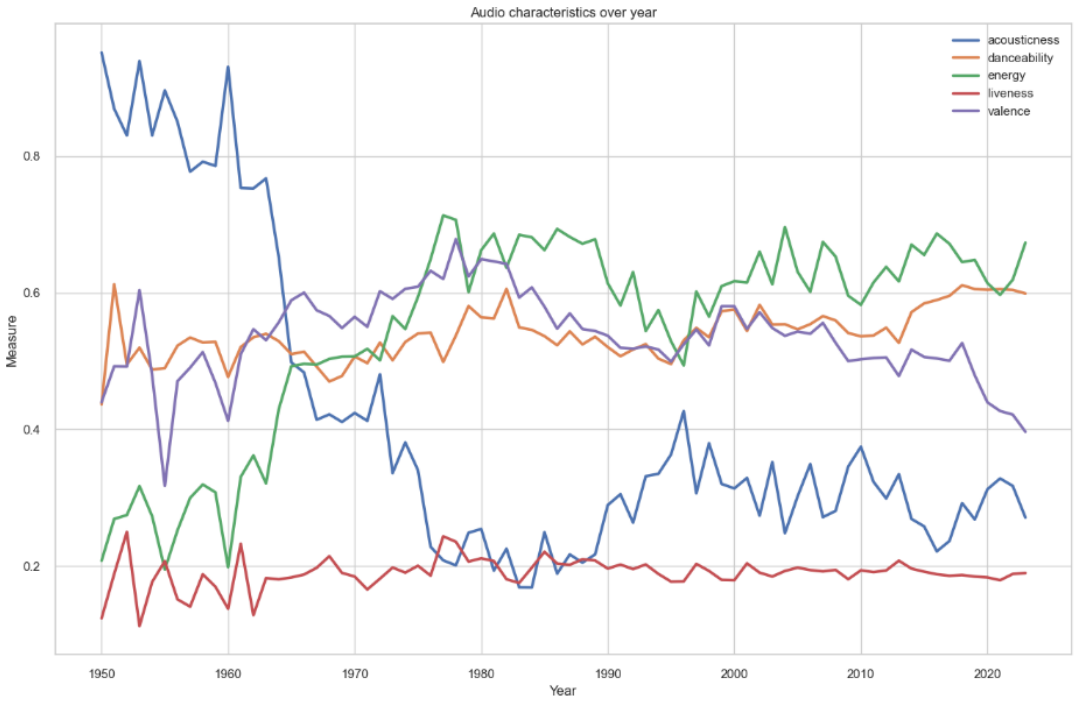


Figure 17. Evolution of audio features over time.

The line plot in *Figure 17* illustrates a significant drop in acousticness levels during the 1960s. This could be attributed to the emergence of genres like rock and electronic music that favored electric and synthetic sounds over acoustic elements. Consequently, the acousticness levels experienced a notable decrease during this period.

Conversely, the energy levels demonstrated an upward trend throughout the 1960s. This increase could potentially be linked to the rise of dance-oriented genres and experimental music [34], which often featured higher energy arrangements and instrumentation.

Furthermore, the metric of valence reached its peak values in the 1980s. This rise might be associated with the prevalence of upbeat and positive-sounding genres like pop and new wave during this decade [34]. These genres often conveyed a sense of optimism and exuberance, contributing to the heightened valence scores observed in the music of that era.

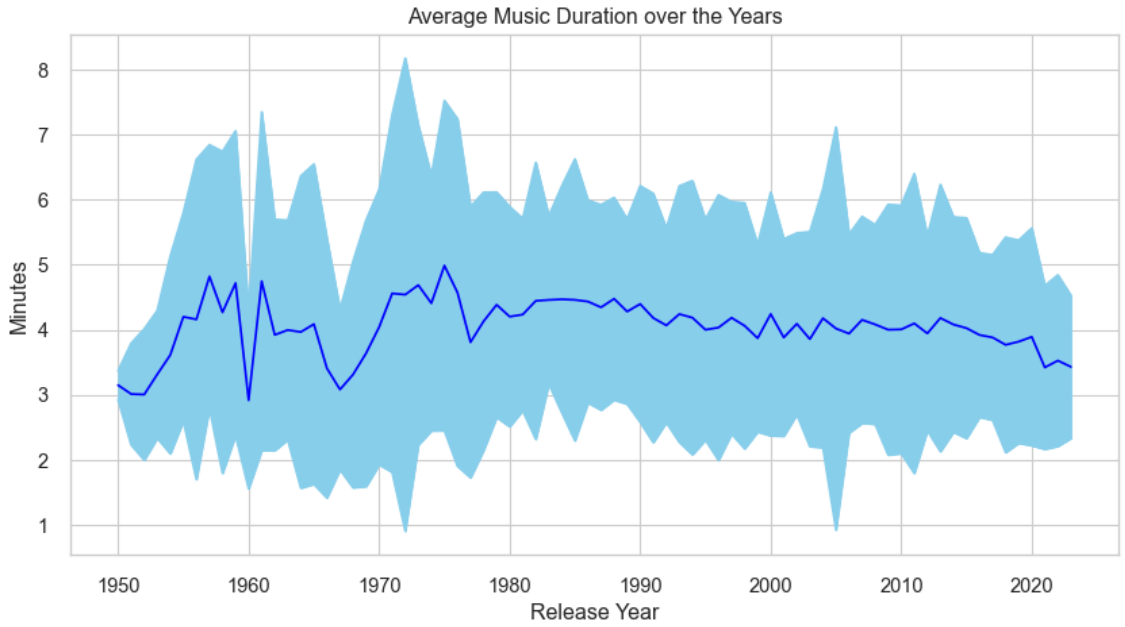


Figure 18. Average music duration over the years.

Figure 18 illustrates the evolution of duration over the years. There is a consistent trend where the mean duration of songs has hovered between approximately 3 and 5 minutes over the last 70 years. Nonetheless, songs released during the 1970s exhibit a notable elongation in duration, while more recent years showcase a gradual reduction in song durations.

The decline in song durations in recent decades could potentially be attributed to shifting music consumption habits. As technology has evolved, platforms like streaming services and social media have become prevalent, encouraging the creation of shorter songs that are better suited for quick and repetitive listening.

In contrast, the higher song durations during the 1970s might be linked to the era's preference for instrumental solos and experimental compositions. Genres like progressive rock, which was popular in the 1970s, often incorporated extended instrumental breaks and elaborate sequences, contributing to the overall longer durations of songs released during that time.

Lastly, it is crucial to consider the potential discrepancy between the dataset and the reality of music trends. For instance, for periods characterized by shorter song durations, the dataset could predominantly contain specific genres known for their briefer compositions. However, due to the absence of labels in the data, this hypothesis remains unverified and cannot be further explored.

3.4. Data preparation

Fortunately, the metadata obtained from the Spotify API was largely prepared for analysis. The following steps were implemented for data cleaning and feature engineering:

- Identified missing values.
- Deleted rows containing missing values.
- Detected and removed duplicate tracks from the dataset.
- Introduced the *'release_year'* column, extracting only the year from each entry in the *'release_date'* column.
- Eliminated the *'release_date'* column.
- Formulated a *'duration'* column, converting song durations from milliseconds to seconds.
- Deleted the *'duration_ms'* column.

3.5. Modeling

Given that the dataset is not explicitly labeled, an unsupervised learning approach is suitable. In this strategy, data is initially clustered (see Section 2.4.5). Following this, the recommendation engine is provided a list of two songs, and the system predicts their respective cluster to which the given songs belong. This identified cluster subsequently guides song recommendations, ensuring that the suggestions align with the characteristics of the given songs.

For the clustering process, two distinct types of algorithms were selected. In broad strokes, these can be into two categories based on their clustering methodologies: algorithms that do not require a specific number of clusters as an input (density-based models), and algorithms necessitate the specification of an anticipated number of clusters (centroid based models) (see Section 2.4.5).

Density-based models: these algorithms identify dense regions within the data space characterized by a high concentration of data points, distinct from sparser regions. A notable advantage of these models is their capability to identify complex cluster shapes, coupled with an inherent robustness against noise and outliers.

An example of density-based algorithms is DBSCAN (Density-Based Spatial Clustering of Applications with Noise). DBSCAN identifies clusters based on the proximity of data points, while marking isolated points in low-density regions as outliers.

Centroid-based models: These algorithms necessitate the specification of an anticipated number of clusters, commonly referred to as 'k'. The data is partitioned into 'k' clusters,

which are typified by a central point or 'centroid'. A representative algorithm of this type is K-Means. K-Means, functions by iteratively optimizing the positions of centroids to ascertain the most appropriate cluster assignments.

Through the utilization of both clustering approaches, the goal is to determine the most coherent groupings within the data, thereby enhancing the accuracy and relevance of song recommendations.

The resulting clusters were scored using two distinct metrics: the Silhouette coefficient and the Davies-Bouldin index.

Silhouette Coefficient: The Silhouette Coefficient ranges from -1 to +1. Values close to 1 indicate that the object is well matched to its own cluster and poorly matched to neighboring clusters. Conversely, values close to 0 indicate clusters may be overlapping. A negative value indicates that samples might have been assigned to the wrong clusters.

Davies-Bouldin Index: this index "measures the average similarity between clusters, where similarity compares the size of clusters against the between-cluster distance. A lower score means that the cluster is relatively small compared to the distance to another cluster, hence well-defined" [35]. Similarity is gauged by the ratio of distances within a cluster to distances between clusters. In this context, scores are on a scale where the minimum is zero; lower values signify superior clustering. Essentially, when the average similarity is minimal, clusters are more distinct, indicating a better clustering outcome.

Inertia: metric that "can be recognized as a measure of how internally coherent clusters are" [36]. It is calculated as the sum of square distances between a data point and its centroid within a cluster. For this reason, it is commonly referred as "within-cluster sum of squares" (WCSS) value.

3.5.1. Data Scaling

As a consistent practice, data scaling is implemented prior to clustering, ensuring that variables are adjusted to a common scale. The method applied to scale the data was Standard Sailing, also known as Z-score normalization through StandardScaler[23], a module from the python library *scikit-learn* (see Section 3.1.2). The reason for scaling the data is to

[23] *StandardScaler* is a preprocessing module in *scikit-learn* that is used to scale features to zero mean and unit variance. This means that the transformed data will have a mean of 0 and a standard deviation of 1. More information is available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

standardize the range of independent variables, ensuring that none of the features dominates others when the machine learning algorithms are applied.

The dataset before scaling contains 71785 rows and 18 columns. The scaled dataset contains the same number of records, but 14 columns, since columns containing non-numerical values were omitted.

3.5.2. Data Projection

TSNE (T-distributed Stochastic Neighbor Embedding) is used to project the data in a 2D space. This algorithm takes the original high dimensional dataset and reduces it into a lower dimensional graph, preserving the clusters of the original dataset.

The graph in Figure 19 is the result of applying the t-SNE algorithm to the high dimensional dataset, where some clusters are visible.

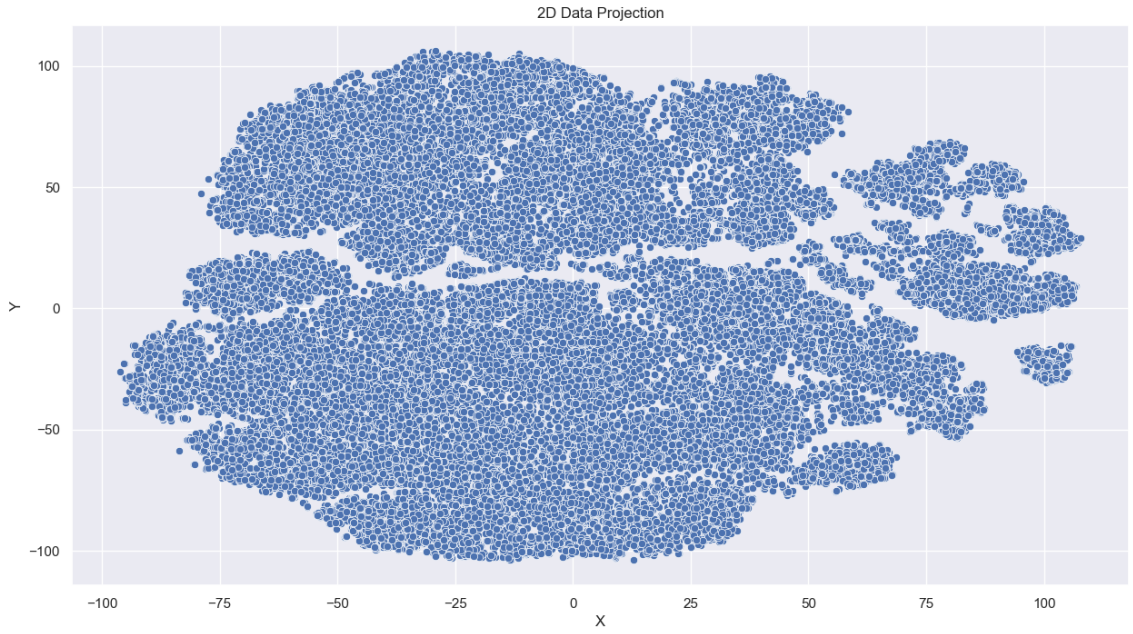


Figure 19. 2D Projection of the data.

3.5.3. Clustering

3.5.3.1. DBSCAN

DBSCAN is a density-based clustering algorithm that forms clusters of dense regions of data points ignoring the low-density areas (considering them as noise).

- Advantages:
  - Good performance with noisy datasets.
  - Identifies outliers.

- Clusters can take any irregular shape unlike K-Means, where clusters are spherical.

- Disadvantages:
  - Does not work very well for sparse datasets or datasets with varying density.
  - Sensitive to the epsilon parameter.
  - Not partitionable for multiprocessor systems.

Setting parameter epsilon:

The behavior of the model is dictated by several parameters, one of them being epsilon, known as “eps” [37]. “Eps” is a parameter that deals with the radius of the clusters being sought. A suitable value for “eps” was identified through a process that involved the calculation of distances to the nearest *n* points for each data point. These distances were subsequently plotted. An examination was conducted to identify the point of most pronounced change. The value corresponding to this point was then chosen as the selected epsilon parameter.

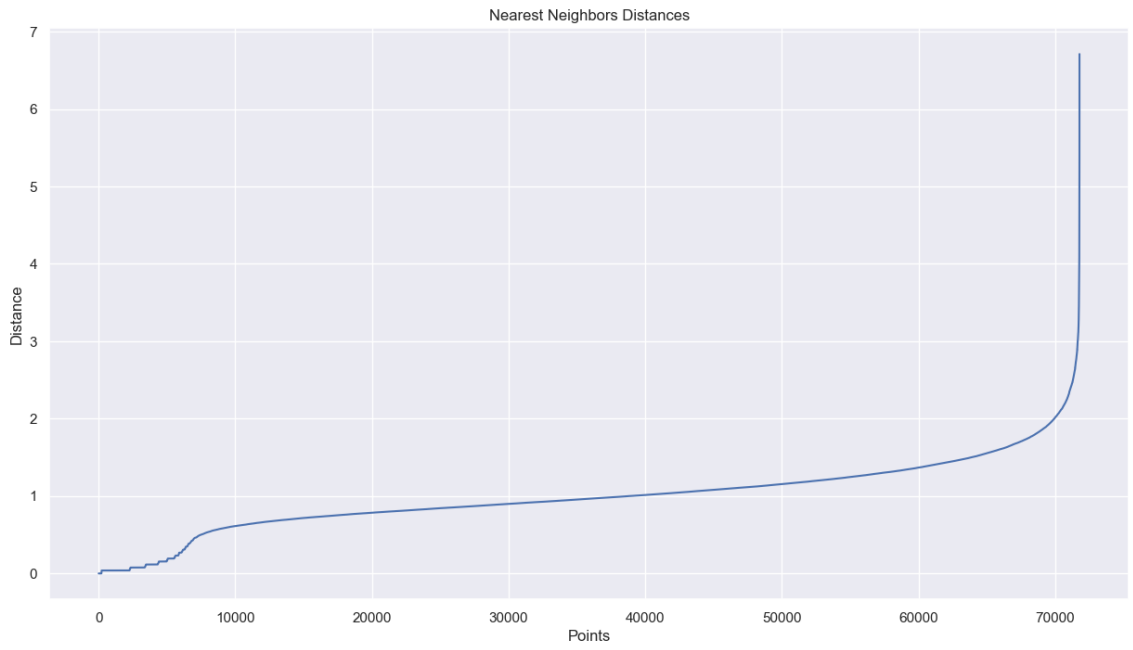


Figure 20. Nearest Neighbors distances.

Figure 20 is the result of the latter process, from which is it observed that 2 is the optimal value for the epsilon parameter.

In order to determine epsilon with more accuracy, the Silhouette coefficient and the Davies-Bouldin index were calculated for different values of epsilon varying from 1.8 to 2.9. Both scores offer a concise measure of the effectiveness of the clustering process. The clustering

process is most effective when the samples of each cluster are cohesive (unified) and the clusters are well separated from one another.

The highest values for both scores resulted for an epsilon value of 2.2, which was chosen as its optimal value.

Clustering with DBSCAN:

The clustering of the data using the DBSCAN algorithm with an epsilon value of 2.2 and visualization of the resulting clusters in the two-dimensional space (transformed through t-SNE), resulted in the graph showed in Figure 21, which portraits the different clusters, distinguished by color and style. From this graph, it was estimated that the optimal number of clusters for this dataset is 7.

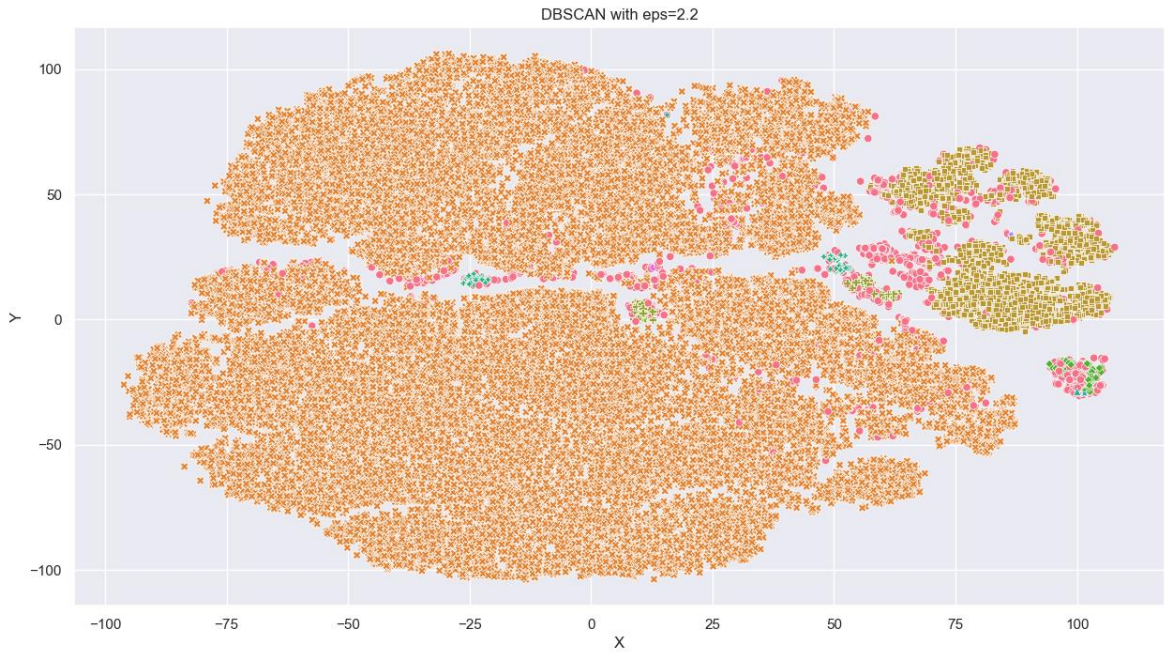


Figure 21. Clustering with DBSCAN.

3.5.3.2. K-Means

The K-Means clustering is one of the most renowned techniques in unsupervised machine learning. This method is a centroid-based model, which implies that it requires a determination of the number of centroids (clusters) before running the model. At the beginning of the clustering procedure, the given clusters are placed randomly.

- Advantages:
  - The technique boasts a straightforward conceptual understanding and ease of implementation.
  - It exhibits robust performance when handling extensive datasets.

- Disadvantages:
    - Its efficacy is contingent upon the accurate selection of the number of clusters or centroids.
    - The algorithm's performance is compromised in the presence of outliers.
    - The computational efficiency diminishes as the dimensionality of data escalates.

Determining the number of clusters: to determine the number of clusters (value of parameter 'k'), the performance of the K-Means clustering was evaluated for different values, ranging from 2 to 29. For each cluster, three metrics were calculated and plotted: silhouette score, Davies-Bouldin score, and inertia (also referred as WCSS).

The selection of the value 'k' for clustering the dataset was made based on the *Elbow method.* This method consists in plotting 'k' versus their resulting WCSS value, and observing the value of 'k' where the graph starts to exhibit a linear appearance.

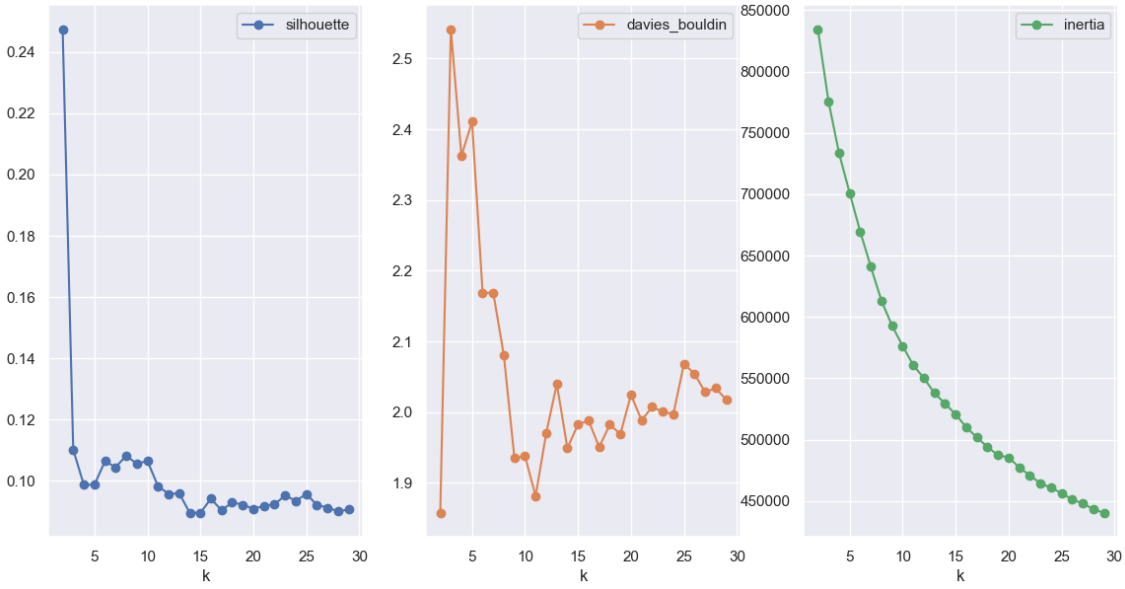


Figure 22. K-Means scores.

Referring to the third line plot in *Figure 22*, the *elbow-point* becomes evident at the point when k=10. To validate this estimation, attention is directed towards the other two evaluation metrics. The priority of this clustering process is to discern well-defined clusters that encapsulate akin songs. This is made effective through lower values of the Davies-Bouldin index, which serves as the discerning criterion for establishing the optimal number of clusters (value of 'k').

```
Clusters = 3 and silhouette_score = 0.11321022259769027 and davies_bouldin_score = 2.423078653485882
Clusters = 4 and silhouette_score = 0.10388277125521518 and davies_bouldin_score = 2.398280242081156
Clusters = 5 and silhouette_score = 0.09764058698133722 and davies_bouldin_score = 2.2585025804999654
Clusters = 6 and silhouette_score = 0.11384206915132744 and davies_bouldin_score = 2.107406698560658
Clusters = 8 and silhouette_score = 0.10817196647100556 and davies_bouldin_score = 2.0808398653498763
Clusters = 9 and silhouette_score = 0.10552581408804909 and davies_bouldin_score = 1.93482304966372
Clusters = 11 and silhouette_score = 0.09536328989139803 and davies_bouldin_score = 1.9214524176437602
Clusters = 15 and silhouette_score = 0.09243411293131663 and davies_bouldin_score = 1.9189832553240196
Best k=15
Best silhouette_score=0.09243411293131663
Best davies_bouldin_score=1.9189832553240196
```

Figure 23. Screenshot of the scores for the computed values.

The Davies-Bouldin index was calculated for values of k ranging from 3 to 15 to confirm the latter estimation of the optimal value of 'k' being 10. Scores were not computed for values of 'k' exceeding 15, since the second line plot in Figure 22 illustrates how the values of the index commence to raise for values above 15 approximately.

Based on the results shown in Figure 23, the optimal value of 'k' for the K-Means algorithm was set to 15.

The clustering was performed with 'k' = 15, and plotted in the 2D graph obtained through the t-SNE algorithm (see Section 3.5.2). The results are presented in Figure 24.

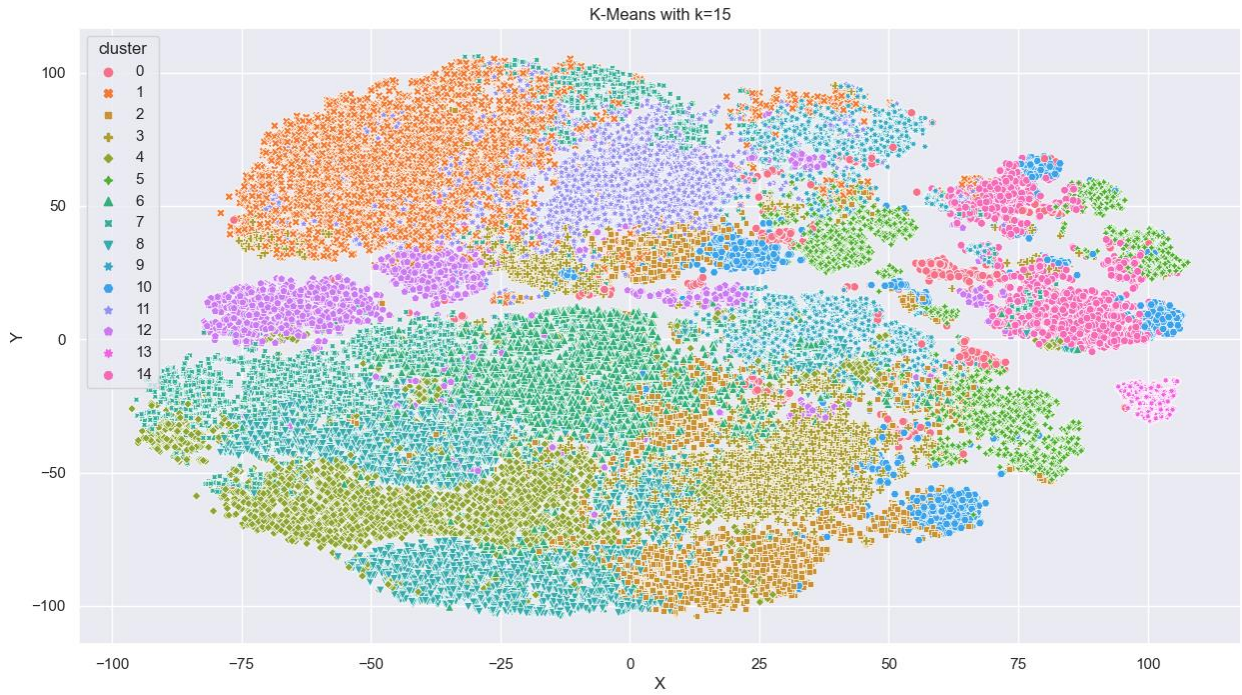


Figure 24. Clustering with K-Means.

3.5.3.3. Model Selection

For selecting a clustering model, the priority was to achieve high cohesion within individual clusters and considerable separation between different clusters. Hence, the guiding principle for selecting a model was to achieve a low Davies-Bouldin index.

The results for the Davies-Bouldin index obtained in sections 3.5.3.1 and 3.5.3.2 are presented in Table 12.

Table 12. Clustering scores comparison.

Clustering algorithm	Silhouette Coefficient	Davies – Bouldin Index
DBSCAN	0.13585	2.14288
K-Means	0.09243	1.91898

Prioritizing the cohesion within individual clusters, K-Means appeared to be the best model to create meaningful clusters (its score for the Davies-Bouldin index is lower).

3.5.4. Music Recommender

Based on the dataset created in chapter 3.2, and the selected model in section 3.5.3, a music recommendation was created based on clustering. The fact that each cluster encompasses akin songs is a key aspect of this recommender system, since each cluster served as a boundary. By having a boundary, the risk of recommending unrelated songs is limited.

The music recommender works as it follows: the user inputs two songs (and artists) as seed songs for a playlist of similar songs. The seed songs are searched in the Spotify API, and its features are retrieved.

The previously fit model creates a target vector calculating the feature means of the seed songs, and predicts which cluster the user's preferences fall in. Subsequently, ten songs from that cluster are returned as a recommendation.

This process is performed through the following functions:

find_song (list):

Takes a list of song titles and artists as input and searches for each song on Spotify using the Spotipy library. It retrieves the track information and audio features for each song and stores them in a *DataFrame*.

I. Parameters: list of two tuples containing a song and its artist.

II. Returns: DataFrame containing the features of the given songs.

recommend_songs (feats_df, track_data, y, n):

This function assigns songs to clusters based on their audio features and determines the cluster label for the input songs 'y'. Utilizing this label, the function selects the songs that share the same cluster as the input songs. Subsequently, the function computes the cosine similarity between the isolated songs and the input, returning the titles and artists of the n most similar songs from track_data.

I. Parameters:
   a. feats_df: a DataFrame containing the audio features of songs.
   b. track_data: a DataFrame containing information about the songs.
   c. y: the input song's audio features.
   d. n: the number of recommended songs to return.

II. Returns: the name and artist of 'n' recommended songs.

The images below provide an illustration of the process. Figure 25 captures a screenshot of the user interface, where the user inserts the name and author of two songs (as input parameters for the function *find_song*). Figure 26 showcases an example of the output returned by the functions described above. In this example, the specific seed songs were "Wish you Were Here" by Pink Floyd, and "Pictures of You" by The Cure.

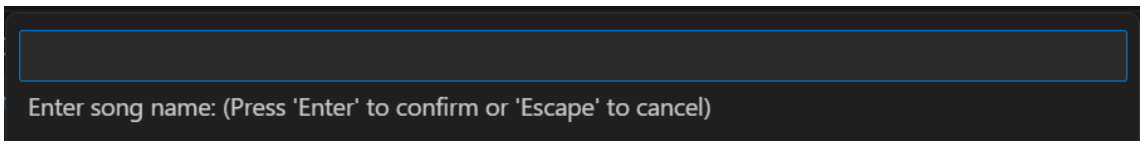
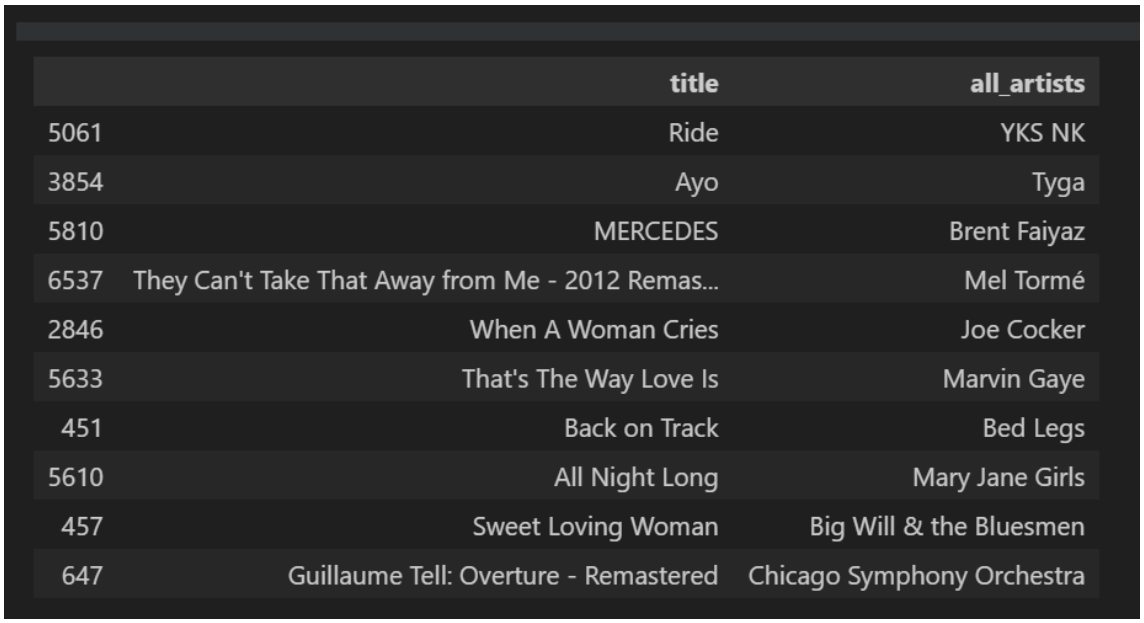


Figure 25. Window to introduce a seed song in Visual Studio Code.



	title	all_artists
5061	Ride	YKS NK
3854	Ayo	Tyga
5810	MERCEDES	Brent Faiyaz
6537	They Can't Take That Away from Me - 2012 Remas...	Mel Tormé
2846	When A Woman Cries	Joe Cocker
5633	That's The Way Love Is	Marvin Gaye
451	Back on Track	Bed Legs
5610	All Night Long	Mary Jane Girls
457	Sweet Loving Woman	Big Will & the Bluesmen
647	Guillaume Tell: Overture - Remastered	Chicago Symphony Orchestra

Figure 26. Recommended songs given two seed songs.

4. Results and Discussion

This section aims to summarize and discuss the results presented in section 3, where the examination of features is performed, and the recommendation engine is built.

Correlation of features and popularity:

The correlation observations (see section 3.3.3.1), especially those relating to the features *energy*, *loudness*, *danceability*, and *valence*, provide valuable insights into what musical features might be intrinsic to the popularity of songs. The absence of *Instrumentalness* in popular tracks (illustrated in Figure 15) suggests that the presence of lyrics and vocal components hold significant influence in the popularity of songs.

On the other hand, the sparsity observed in less popular songs (illustrated in Figure 16) and their divergence from the popular characteristics bring insight into the vast musical landscape beyond the mainstream trends. This variance is very likely due to the presence of experimental genres, niche musical tastes, or tracks that have not yet found their audience within the dataset. Referring to *Instrumentalness,* which is almost null in the 25% most popular songs of the dataset, tracks within the 25% least popular scores, present varying degrees of the feature. The latter tracks appear to cater to an audience that seeks instrumental music, classical tunes, or other genres where vocals play a secondary role.

These observations underscore the need for recommendation systems in today's digital age. Such systems can potentially narrow the gap between tracks outside the mainstream (those represented in Figure 16) and their respective audience, resonating with the concept of the long tail economy.

Temporal Evolution of Audio Features:

The study of the temporary evolution of audio features derives from the data displayed in Figure 17. The key findings on this topic are summarized and discussed below:

- Acousticness: Experienced a significant drop in the 1960s due to the emergence of rock and electronic music which leaned towards electric and synthetic sounds over traditional acoustic elements.

- Energy: Saw an upward trend during the 1960s, attributed to the rise of dance-oriented and experimental music genres such as dance-music [38].

- Valence: Reached peak levels in the 1980s, reflecting the popularity of upbeat genres like pop and new wave that conveyed a sense of optimism and exuberance. However, this metric appears to be consistent over the years. One of the central roles of music across cultures and eras is to evoke emotion and often to bring joy

or solace. While genres might change, this basic function of music remains, leading to a consistent valence.

- Danceability: Remained consistent over the years, suggesting a continuous preference for rhythms and beats conducive to dance across various musical genres and eras.

- Liveness: Maintained a stable presence across the decades, indicating that the representation of live music versus studio recordings in popular tracks did not undergo drastic shifts over time.

The stability of the features *danceability* and *valence,* prove that positive feelings [24]are universally relatable and timeless aspects of the human experience. Regardless of trends or technological shifts, audiences have always been drawn to music that moves them, both emotionally and physically.

Temporal Evolution of Duration:

Regarding the evolution of songs durations illustrated in Figure 18, from which three main aspects stand out:

- Consistency Over the Years: Song durations have largely remained stable, primarily ranging between 3 to 5 minutes over seven decades, suggesting a standard in music production that aligns with optimal listener engagement.

- 1970s Elongation: There was a notable increase in song lengths during the 1970s. This is likely attributed to the era's musical preferences, particularly the rise of genres like progressive rock, which were characterized by extended instrumental solos and complex compositions.

- Recent Decline: The last decades showcase a decline in song durations. This reflects the evolving music consumption habits, due to the prevalence of music streaming services, and the brevity-favoring nature of social media users.

Delving deeper into de recent decline in songs durations, it appears that the modern music landscape is shaped by technological advances, which are shifting consumer preferences towards more brief and concise content. Another possible explanation is the fact that platforms such as Spotify and Apple Music, which reign the music industry nowadays, reward artists based on their number of streams [39]. Since shorter songs can achieve more

[24] Valence represents the musical positiveness conveyed by a track. Higher valence values represent more positive and cheerful tracks, while lower values represent more negative and sad tracks.

streams in a given period, releasing shorter tracks can potentially maximize revenue for artists.

Furthermore, the rise of short-video platforms, such as TikTok and Instagram, amplify the appeal of brief, and catchy snippets of songs that can quickly capture attention [40], and potentially go viral, achieving more streams, thus maximizing revenue. This push for immediacy, reinforces the trend of reducing song lengths.

Music Recommender:

In the pursuit of building a recommendation engine based on the features of the data set, two clustering methods were evaluated. The method with better perform

Prior to implementing the recommendation engine, the data in the dataset was clustered, for which two methods, DBSCAN and K-Means, were compared. The succeeding summary presents the results derived from sections 3.5.3 and 3.5.4:

- DBSCAN vs K-Means Analysis: The optimal value for DBSCAN's epsilon was 2.2, resulting in seven clusters. However, K-Means, after thorough evaluation, was determined to be more effective with 15 clusters. Visualization further confirmed K-Means' better cluster separation compared to DBSCAN.

- Model Efficacy: Based on the results and the Davies-Bouldin scores, K-Means outperformed as the best model for clustering. Its low Davies-Bouldin score indicates well-defined and cohesive clusters, although having a Silhouette score indicates that the clusters are potentially overlapping, DBSCAN was chosen for scoring better than DBSCAN.

- Music Recommender Functionality: The recommender system, built on the K-Means clusters, offers song suggestions derived from two seed songs provided by the user. These songs are matched to a cluster, ensuring that the recommendations are musically akin.

Post-development reflections include acknowledging the complexity inherent to recommendation systems. While this system is functional and offers relevant recommendations, achieving a recommender system on par with platforms like Spotify demands vast amounts of user data and insights. In order to mimic the recommendations these platforms provide, the implemented engine should utilize a hybrid model, capable of merging collaborative and content-based filtering.

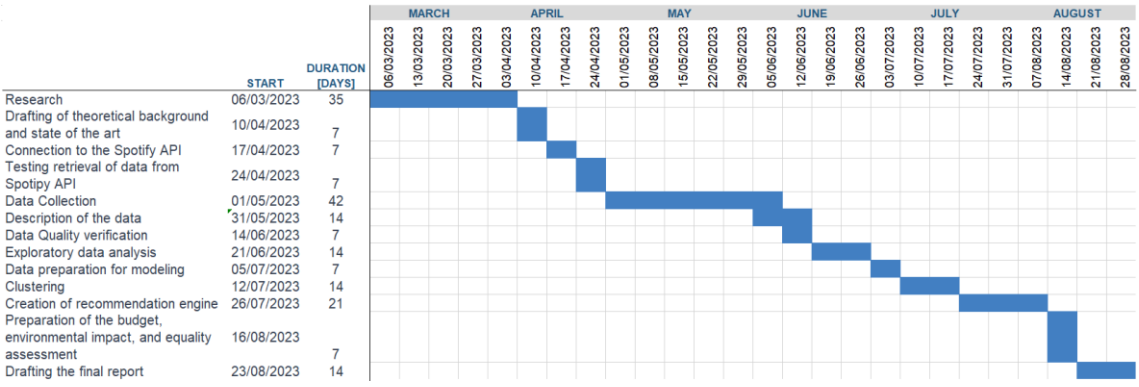
5. Planning

For any project to be executed, it's essential to devise a plan estabilishing how and when the various activities required by the project will be executed. In this case, at the beginning of the project, a plan was designed, which, due to various circumstances that arose, was not implemented. For this reason, a more realistic plan was formulated subsequently, as illustrated below. This planning was constructed with the aid of a Gantt chart.

It is necessary to compile a list of the tasks to be executed [41]:

1. Research.
2. Drafting of theoretical background and state of the art.
3. Connection to the Spotify API.
4. Testing retrieval of data from Spotipy API.
5. Data Collection.
6. Description of the data.
7. Data Quality verification.
8. Exploratory data analysis.
9. Data preparation for modeling.
10. Clustering.
11. Creation of recommendation engine.
12. Preparation of the budget, environmental impact, and equality assessment.
13. Drafting the final report.

Upon completion of the project, the duration of each task has been updated to provide a true temporal representation of the time taken for each activity. The Gantt chart that illustrates the aforementioned details is displayed in Figure 27.



	START	DURATION [DAYS]
Research	06/03/2023	35
Drafting of theoretical background and state of the art	10/04/2023	7
Connection to the Spotify API	17/04/2023	7
Testing retrieval of data from Spotipy API	24/04/2023	7
Data Collection	01/05/2023	42
Description of the data	31/05/2023	14
Data Quality verification	14/06/2023	7
Exploratory data analysis	21/06/2023	14
Data preparation for modeling	05/07/2023	7
Clustering	12/07/2023	14
Creation of recommendation engine	26/07/2023	21
Preparation of the budget, environmental impact, and equality assessment	16/08/2023	7
Drafting the final report	23/08/2023	14

Figure 27. Gantt Chart.

6. Economic Assessment

This section aims to provide a comprehensive overview of the investments involved in the undertaking of this project. Given the digital nature of the project, its financial assessment primarily revolves around the human capital invested in terms of hours worked and the experimental costs incurred during its execution.

The tasks performed in this project are akin to those of a junior data scientist. According to recent figures, the average hourly wage for a junior data scientist in Spain is approximately 17.95 €/h [42]. However, in this assessment, a rate of 15 €/h is considered, which aligns with the expected compensation. Based on the latter information, the valuation of human capital is calculated as follows:

- Dedicated hours: 410 hours.
- Hourly wage: 15 €/h.

Based on the latter information, the valuation of human capital calculated as follows:

$$410\ h\ \cdot 15\frac{€}{\text{h}} = 6{,}150\ € \qquad \text{(Eq. 7.1)}$$

Two resources were used to facilitate the assist in research and report drafting; Academia.edu and Microsoft Office 365 Personal. The latter was used during the entire development of the project for the drafting of this report. While the subscription to Academia.edu provided access to valuable books and reports. Both resources were subject to monthly charges, with their respective costs displayed below.

- Microsoft Office 365 Personal: 7 €/month.
- Academia.edu Subscription: 9 €/month.

The effective cost of software and subscriptions is calculated in Equation 7.2.

$$(7+9)\frac{€}{\text{month}} \cdot 6\ months = 96\ € \qquad \text{(Eq. 7.2)}$$

Regarding the cost of hardware tools, the entire project was developed in a single Personal Computer (PC). Considering the PC was purchased in 2019, and the principle of linear amortization over a span of 5 years, by 2023 the computer has reached its full amortization period.

All in all, the overall cost of the project amounts to a total of 6.246 €. To enhance visual clarity, all the aspects are grouped together in Table 13.

Table 13. Overall costs of the project.

Concept	Wage / Cost	Duration	Total
Human Capital	15 € / hour	410 hours	6,150 €
Microsoft Office 365	7 € / month	6 months	42 €
Academia Subscription	9 € / month	6 months	54 €
Total Cost			**6,246 €**

7. Environmental Assessment

This chapter endeavors to conduct an evaluation of the environmental impact associated with the project.

The following indicators have been chosen to provide insights into the environmental aspects of the project:

Carbon footprint: The project was executed remotely without any physical construction. However, electricity consumption for computer usage, internet connectivity, and data processing might have contributed to carbon emissions. Using available data on energy consumption and specific emission factors, an estimation of kg CO2 associated with the energy is calculated, taking into account the following considerations:

- The computer's energy consumption was high due to working on numerous documents and applications simultaneously. The laptop's specifications determine that the computer's consumption at high performance mode is 65 W [43].
- The router used for this project has an average consumption of 12.4 W [44].

To determine the total energy consumption, the number of hours worked must be considered. As already mentioned in the economic study, the project took a total of 410 hours. It is noteworthy that the Final Degree Project is worth 12 ECTS, which is approximately equivalent to about 360 work hours. However, due to the complexity of the work, especially in the Data Collection phase, the hours spent exceeded the initial expectations.

Given the two points, the power consumption during the development of the project is calculated in Eq. 8.1.

$$410\ hours \cdot (65\ W + 12.4\ W) = 31.734\ kWh \qquad \text{(Eq. 8.1)}$$

The emission factors report from the "Ministerio para la Transición Ecológica y el Reto Demográfico" [45] was consulted to obtain the CO_2 emission rate per kWh consumed by electrical emission factors. This allows for the calculation of the carbon footprint associated with the project's energy consumption. The emission rate for electrical emission factors from the energy provider "ENDESA ENERGÍA S.A.U" is 0.272 kg CO_2 / kWh.

$$0.272\ kg\ \frac{CO_2}{kWh} \cdot (31.734\ kWh) = 8.63\ kg\ CO_2 \qquad \text{(Eq. 8.2)}$$

As calculated in Equation 8.2, the generated carbon footprint amounts to 8.632 kg of CO_2.

Resource Consumption and waste generation: The resources utilized in the project were digital, including software tools, data sources, and online platforms. No physical resources were depleted, or consumed during the project's execution, and no waste materials were generated.

Water Usage: As the project was digital in nature and did not involve any industrial processes or physical products, water consumption is negligible and can be considered minimal.

In conclusion, this environmental assessment demonstrates the project's alignment with sustainable practices. By conducting the project remotely, leveraging digital resources, and minimizing waste production, the project exhibits a responsible approach towards environmental considerations.

8. Social and Gender Equality Assessment

This chapter aims to examine the inclusive approach undertaken in the development of this project, and underscore the equitable accessibility of resources and results. To do so, some aspects including access to technology, information, and gender balance, among others, have been addressed.

Technology: This project is primarily based in Python, and its libraries, which are all open-source. The code editor (Visual Studio Code), and the version control repositories on GitHub were free of costs as well. These tools are available to individuals with access to a computer and an internet connection, making them inclusive in today's context.

Access to information: Most of the resources accessed are available for free in the internet. However, some of the consulted books were exclusively available at Academia[25], a subscription-based platform for sharing academic research. Also, some resources were attained through Bibliotecnia[26], UPC's digital library.

Gender balance within contributors: The team engaged in this project consists of the director, and the author, myself. From a gender perspective, the project is balanced, since the author (myself) is a woman.

Gender balance within cited authors: There are a total of 50 authors within the references cited in chapter 11. Among the 50 authors, only 8 are female, 29 of them are written by male authors, and the other 7 have ambiguous names that lack genre clarity. These numbers reveal the disparity of voices within the field of data science and Machine learning. Acknowledging and rectifying such disparity in the authorship of the consulted resources could foster a more comprehensive and inclusive landscape.

Use of Images and Language: This report is written in an impersonal language, without gender distinctions. All images within the report have been created by the author, the majority of which are graphics presenting data. The images are not stratified by gender, as the gender of each author of the dataset was not retrievable information.

Future contributions: The project's code will be publicly accessible on GitHub, enabling individuals from all over the world, without charges, to employ the code in their own projects, enhancing its functionality, or utilizing it as the base for their own personal endeavors.

Sustainable Development Goals (SDGs): Considering the latter observations, the project contributes to the following objectives: (5) "Gender Equality" for the gender balance within

[25] Available at: https://www.academia.edu/

[26] Available at: https://bibliotecnica.upc.edu/

contributors, (10) “Reduced inequalities” for being publicly accessible, and (12) “Responsible consumption and production” for not having generated any physical waste, nor purchasing paper-based books.

All things considered, this project is not potentially discriminatory, nor influenced by gender or social status. The exploratory analysis and recommendation engine are based on a data set constructed by performing random searches from artists of many genres and subgenres. The fact that the searches utilized in the construction of the data set did not regard the popularity of songs, or address specific trends, is a guarantee that artists from all social statuses are included in the data set. In terms of genre, the genre of each artist was not available, so the analysis does not regard this aspect, although it could be interesting.

Lastly, it is important to highlight that while the references predominantly feature male authors, this project remains non-discriminatory and actively contributes to social and gender equality, as well as championing minimal environmental impact.

9. CONCLUSIONS

Based on the project's initial objectives, the following conclusions are reached regarding their fulfillment:

- A comprehensive dataset was successfully curated from Spotify, encompassing a diverse range of songs, inclusive of both mainstream and niche subgenres. This dataset embraces the *Long Tail* distribution principle (see Section 1.1), underscoring the relevance and importance of varied musical representation.

- The temporal evolution of various musical features was effectively analyzed. Changes in features like *acousticness*, *energy*, and *valence* over the decades were discerned, shedding light on the evolution of musical trends.

- The relationship between certain musical features and song popularity was explored. Correlations were identified, emphasizing the dynamic nature of popular music attributes and how they change over time.

- A recommendation engine was developed using the curated dataset. The system's capability to offer song suggestions based on user inputs ensures its functionality and alignment with the project's objective.

In summary, all the set objectives for this project have been addressed and achieved, providing a comprehensive understanding of musical trends, their correlation with popularity, and the construction of a recommendation system.

10. Bibliography

[1] M. Schedl, H. Zamani, C.-W. Chen, Y. Deldjoo, and M. Elahi, “Current challenges and visions in music recommender systems research,” *International Journal of Multimedia Information Retrieval*, vol. 7, no. 2, pp. 95–116, Apr. 2018, doi: 10.1007/s13735-018-0154-2.

[2] F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*. Springer, 2015.

References

[1] ANDERSON, Chris, 2010. *The Long Tail: How Endless Choice is Creating Unlimited Demand*. . Random House. ISBN 9781446409282.

[2] STEVENSON, Angus, 2010. *Oxford Dictionary of English*. . Oxford University Press, USA. ISBN 9780199571123.

[3] IBM Documentation, 2021. Online. [Accessed 12 June 2023]. Available from: https://www.ibm.com/docs/en/spss-modeler/saas?topic=dm-crisp-help-overview

[4] MARTINEZ-PLUMED, Fernando, CONTRERAS-OCHANDO, Lidia, FERRI, Cesar, HERNANDEZ-ORALLO, Jose, KULL, Meelis, LACHICHE, Nicolas, RAMIREZ-QUINTANA, Maria Jose and FLACH, Peter, 2021. CRISP-DM Twenty Years Later: From Data Mining Processes to Data Science Trajectories. *IEEE Transactions on Knowledge and Data Engineering*. 1 August 2021. Vol. 33, no. 8, p. 3048–3061. DOI 10.1109/tkde.2019.2962680.

[5] ESTY, Barbara. Yale University Library Research Guides: Research Data Management: Validate Data. Yale University Library Research Guides at Yale University. Online. 21 October . [Accessed 21 June 2023]. Available from: https://guides.library.yale.edu/datamanagement/validate

[6] MCGREGOR, Susan E., 2021. Practical Python Data Wrangling and Data Quality. . “O’Reilly Media, Inc.” ISBN 9781098112011.

[7] ANMUT, 2021. Data Quality vs Data Condition: The Power of Context. *Anmut*. Online. 12 July 2021. [Accessed 19 June 2023]. Available from: https://www.anmut.co.uk/data-quality-vs-data-condition-the-power-of-context/

[8] MOSES, Barr, GAVISH, Lior and VORWERCK, Molly, 2022. Data Quality Fundamentals. . “O’Reilly Media, Inc.” ISBN 9781098112011.

[9] ADEGEO, 2023. Standard date and time format strings. *Microsoft Learn*. Online. 5 February 2023. [Accessed 24 July 2023]. Available from:

https://learn.microsoft.com/en-us/dotnet/standard/base-types/standard-date-and-time-format-strings

[10] PYTHON SOFTWARE FOUNDATION, 2023. datetime — Basic date and time types. Python documentation. Online. 24 July 2023. [Accessed 24 July 2023]. Available from: https://docs.python.org/3/library/datetime.html

[11] SCHEDL, Markus, ZAMANI, Hamed, CHEN, Ching-Wei, DELDJOO, Yashar and ELAHI, Mehdi, 2018. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*. 5 April 2018. Vol. 7, no. 2, p. 95–116. DOI 10.1007/s13735-018-0154-2.

[12] LUNA, Zipporah, 2021. CRISP-DM Phase 5: Evaluation Phase - Analytics Vidhya - Medium. *Analytics Vidhya*. Online. 10 November 2021. [Accessed 25 July 2023]. Available from: https://medium.com/analytics-vidhya/crisp-dm-phase-5-evaluation-phase-d7bb3c75220a

[13] MAXAMADJONOVNA, Hamidova Muqiymaxon, 2022. TARIX FANINI O'QITISHNDA ZAMONAVIY YONDASHUVLAR. *Zenodo*. Online. 1 May 2022. Available from: https://zenodo.org/record/6516440

[14] RICCI, Francesco, ROKACH, Lior and SHAPIRA, Bracha, 2015. *Recommender Systems Handbook*. . Springer. ISBN 9781489976376.

[15] SCHEDL, Markus, ZAMANI, Hamed, CHEN, Ching-Wei, DELDJOO, Yashar and ELAHI, Mehdi, 2018. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*. 5 April 2018. Vol. 7, no. 2, p. 95–116. DOI 10.1007/s13735-018-0154-2.

[16] SEN, J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad, 2017. *LNCS 4321 - Collaborative Filtering Recommender Systems* Online. [Accessed 2 May 2023]. Available from: https://objects.scraper.bibcitation.com/user-pdfs/2023-05-02/baa92c28-9da2-4307-b5ae-8ba1deca6d3e.pdf

[17] FARNWORTH, Richard, 2020. The Six Dimensions of Data Quality — and how to deal with them. Towards Data Science. Online. 30 June 2020. [Accessed 8 August 2023]. Available from: https://towardsdatascience.com/the-six-dimensions-of-data-quality-and-how-to-deal-with-them-bdcf9a3dba71

[18] Black A. and van Nederpelt P.. 2020. Dimensions of Data Quality (DDQ) Research Paper. Retrieved June 10, 2021 from http://www.dama-nl.org/wp-content/uploads/2020/09/DDQ-Dimensions-of-Data-Quality-Research-Paper-version-1.2-d.d.-3-Sept-2020.pdf.

[19] PRIESTLEY, Maria, O'DONNELL, Fionntán and SIMPERL, Elena, 2023. A Survey of Data Quality Requirements That Matter in ML Development Pipelines. *Journal of Data and Information Quality*. 22 June 2023. Vol. 15, no. 2, p. 1–39. DOI 10.1145/3592616.

[20] BATINI, Carlo and SCANNAPIECO, Monica, 2006. Data Quality: Concepts, Methodologies and Techniques. Springer Science; Business Media. ISBN 9783540331735.

[21] MOSES, Barr, GAVISH, Lior and VORWERCK, Molly, 2022b. Data Quality Fundamentals. “O’Reilly Media, Inc.” ISBN 9781098112011.

[22] RASCHKA, Sebastian, 2015. Python Machine Learning. . Packt Publishing Ltd. ISBN 9781783555147.

[23] CELMA, Òscar, 2010. Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space. . Springer Science & Business Media. ISBN 9783642132872.

[24] STEVENSON, Angus, 2010. *Oxford Dictionary of English*. . Oxford University Press, USA. ISBN 9780199571123.

[25] BANOULA, Mayank, 2020. K-means Clustering Algorithm: Applications, Types, & How Does It Work? Simplilearn. Online. 22 April 2020. [Accessed 23 August 2023]. Available from: https://www.simplilearn.com/tutorials/machine-learning-tutorial/k-means-clustering-algorithm

[26] VAN DER MAATEN, L. and HINTON, G. Visualizing Data using t-SNE. Journal of Machine Learning Research. 2008, vol. 9, p. 2579-2605.

[27] HEALEY, C. G. and ENNS, J. T. Attention and visual memory in visualization and computer graphics. IEEE Transactions on Visualization and Computer Graphics. 2012, vol. 18, no. 7, p. 1170-1188.

[28] WATTENBERG, M., VIÉGAS, F. and JOHNSON, I. How to Use t-SNE Effectively. Distill. 2016.

[29] KOBAK, D. and BERENS, P. The art of using t-SNE for single-cell transcriptomics. Nature Communications. 2019, vol. 10, no. 1.

[30] Getting started with Web API, [no date]. *Spotify for Developers*. Online. [Accessed 16 May 2023]. Available from: https://developer.spotify.com/documentation/web-api/tutorials/getting-started

[31] ROBERTS, David, 2021. What is a Web Application? StackPath. Online. 24 May 2021. [Accessed 6 July 2023]. Available from: https://www.stackpath.com/edge-academy/what-is-a-web-application/

[32] Welcome to Spotipy! — spotipy 2.0 documentation, [no date]. Spotipy. Online. [Accessed 17 May 2023]. Available from: https://spotipy.readthedocs.io/en/2.16.1/#authorization-code-flow

[33] ROBERTS, David, 2021. What is a Web Application? StackPath. Online. 24 May 2021. [Accessed 6 July 2023]. Available from: https://www.stackpath.com/edge-academy/what-is-a-web-application/

[34] CRAUWELS, Kwinten , 2022. Musicmap. The Genealogy and History of Popular Music Genres. Online. May 2022. [Accessed 21 August 2023]. Available from: https://musicmap.info/

[35] WONG, Kay Jan, 2022. 7 Evaluation Metrics for Clustering Algorithms. Towards Data Science. Online. 9 December 2022. [Accessed 28 August 2023]. Available from: https://towardsdatascience.com/7-evaluation-metrics-for-clustering-algorithms-bdc537ff54d2

[36] HUBERT, Lawrence and ARABIE, Phipps, 1985. Comparing partitions. Journal of Classification. December 1985. Vol. 2, no. 1, p. 193–218. DOI 10.1007/bf01908075.

[37] MAKLIN, Cory, 2022. DBSCAN Python Example: The Optimal Value For Epsilon (EPS). Towards Data Science. Online. 9 May 2022. [Accessed 19 August 2023]. Available from: https://towardsdatascience.com/machine-learning-clustering-dbscan-determine-the-optimal-value-for-epsilon-eps-python-example-3100091cfbc

[38] BERNSTEIN, Bill, 2021. The History of Dance Music. Armada Music. Online. 14 January 2021. [Accessed 5 September 2023]. Available from: https://www.armadamusic.com/news/the-history-of-dance-music

[39] DALY, Rhian, 2021. Apple Music tells artists it now pays double than Spotify per stream. NME. Online. 16 April 2021. [Accessed 5 September 2023]. Available from: https://www.nme.com/news/music/apple-music-tells-artists-now-pays-double-spotify-per-stream-2922446

[40] LEWIS, Wendy, 2021. THE TIKTOK PHENOMENON AND THE RISE OF SHORT FORM VIDEO. PRIME Journal. Online. April 2021. [Accessed 5 September 2023]. Available from: https://www.prime-journal.com/the-tiktok-phenomenon-and-the-rise-of-short-form-video/

[41] STSEPANETS, Anastasia, 2022. La guía completa para los diagramas de Gantt: qué es un diagrama de Gantt, cómo se hace y cuándo se usa. Gantt Chart GanttPRO Blog. Online. 12 December 2022. [Accessed 1 September 2023]. Available from: https://blog.ganttpro.com/es/guia-completa-para-los-diagramas-de-gantt/#elementos-de-la-grafica-de-gantt

[42] LORETO, Alejandro Manzanares, 2023. Cuánto gana un Data Scientist en España en 2022. *HACK A BOSS*. Online. 29 June 2023. [Accessed 4 September 2023]. Available from: https://www.hackaboss.com/blog/cuanto-gana-un-data-scientist-en-espana-en-2022

[43] JCAYCHOR, 2016. Consulta, sobre consumo de energía HP240. *hp.es/comunidad*. Online. 22 June 2016. [Accessed 31 August 2023].

Available from: https://h30467.www3.hp.com/t5/Otros-Productos-Archive-Solo-lectura/Consulta-sobre-consumo-de-energ%C3%ADa-HP240/td-p/712057

[44] LLORACH, Joshua, 2023. El nuevo router de Movistar consume un 64% más electricidad que su predecesor el HGU. BandaAncha.eu. Online. 6 January 2023. [Accessed 1 September 2023]. Available from: https://bandaancha.eu/articulos/nuevo-router-movistar-consume-64-mas-10444

[45] MINISTERIO PARA LA TRANSICIÓN ECOLÓGICA Y EL RETO DEMOGRÁFICO, 2022. Factores de emisión. *Miteco*. Online. 2022. [Accessed 31 August 2023]. Available from: https://www.miteco.gob.es/content/dam/miteco/es/cambio-climatico/temas/mitigacion-politicas-y-medidas/factoresemision_tcm30-479095.pdf