Department of Computer Science
Master in Data Science and Business Informatics

DM1 – Data Mining Foundations

Submitted to:

Prof. Dino Pedreschi

Prof. Riccardo Guidotti

Submitted by:

Alessia Ferrero

Alessandra Amico

Sana Afreen

Academic Year 2023/24

CONTENT

Summary

This report provides a comprehensive analysis of the Spotify Tracks dataset, offering insights into the diverse audio tracks available within the Spotify catalog. The dataset encompasses 20 distinct genres, each featuring tracks with fundamental attributes like track name, artist, album, and popularity. Moreover, audio-derived features such as danceability, energy, key, and loudness enrich the dataset, allowing for a deeper exploration of track characteristics. Through detailed analysis and visualization, this report aims to uncover patterns, trends, and correlations within the dataset, shedding light on factors influencing track popularity and genre-specific attributes. The findings presented herein contribute to a better understanding of music preferences and track success within the Spotify ecosystem.

Chapter 1

1. Data Understanding

To gain comprehensive insights into the project, a foundational understanding of the dataset is imperative. Therefore, we present a succinct overview of our data, aiming to facilitate a clearer comprehension of the project's underlying information. Below are our interpretations.

1.1. Data Semantics

The dataset, consisting of 1,5000 records (rows) and 24 attributes (columns), presents a diverse array of data types crucial for understanding the Spotify track catalog.

Among these columns, one boolean attribute indicates binary conditions, while the majority, comprising 14 columns, are of float64 type, housing detailed numeric features like energy, loudness, and tempo. Four integer-type columns offer precise numerical information, likely related to track identifiers or categorical codes. Additionally, the presence of four object-type columns suggests textual or categorical data, potentially containing track names, artists, or genres.

This amalgamation of data types underscores the dataset's multifaceted nature, encompassing numeric audio features, categorical identifiers, and textual information, laying the groundwork for a comprehensive analysis of Spotify tracks.

	Attributes	Descriptions	Type
1	name	Name of the track	object
2	artists	The artists' names who performed the track.	object
3	album_name	The album name in which the track appears	object
4	genre	The genre in which the track belongs	object
5	duration_ms	The track length in milliseconds	int64
6	popularity	The popularity of a track is a value between 0 and 100, with 100 being the	int64
7	key	The key the track is in. Integers map to pitches using standard Pitch Class	int64
8	features_duration_ms	The duration of the track in milliseconds	int64
9	danceability	Danceability describes how suitable a track is for dancing. A value of 0.0 is least danceable and 1.0 is most danceable	float64
10	energy	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.	float64
11	loudness	The overall loudness of a track in decibels (dB)	float64
12	mode	Mode indicates the modality (major or minor) of a track. Major is represented by 1 and minor is 0	float64
13	speechiness	Speechiness detects the presence of spoken words in a track.	float64
14	acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic	float64
15	instrumentalness	Predicts whether a track contains no vocals. The closer the instrumentalness	float64
16	liveness	Detects the presence of an audience in the recording.	float64
17	valence	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a	float64
18	tempo	The overall estimated tempo of a track in beats per minute (BPM).	float64
19	time_signature	An estimated time signature.	float64
20	n_beats	The total number of time intervals of beats throughout the track.	float64
21	n_bars	The total number of time intervals of the bars throughout the track.	float64
22	popularity_confidence	The confidence, from 0.0 to 1.0, of the popularity of the song.	float64
23	explicit	Whether or not the track has explicit lyrics (true = yes it does; false = no it does not OR unknown)	bool
24	processing	Modify existing songs using manipulation. it's range is from 0 to 1	float64

Figure 1.1. Description of the attributes of the dataset

1.2. Distribution of the variables and statistics

In this section we will understand the variables more in depth using some count plots to better understand the distribution of each attribute. The distribution refers to the way in which the values of a variable are spread across different possible outcomes. It provides information about the frequency or probability of different values occurring. Figure 1.2 displays the distribution of each attribute and we can notice that only some of them have a normal distribution, such as 'danceability'. Then, some other variable like 'energy' has a negatively skewed distribution, while others, such as 'popularity', has a positively skewed distribution, with a tail on the right.

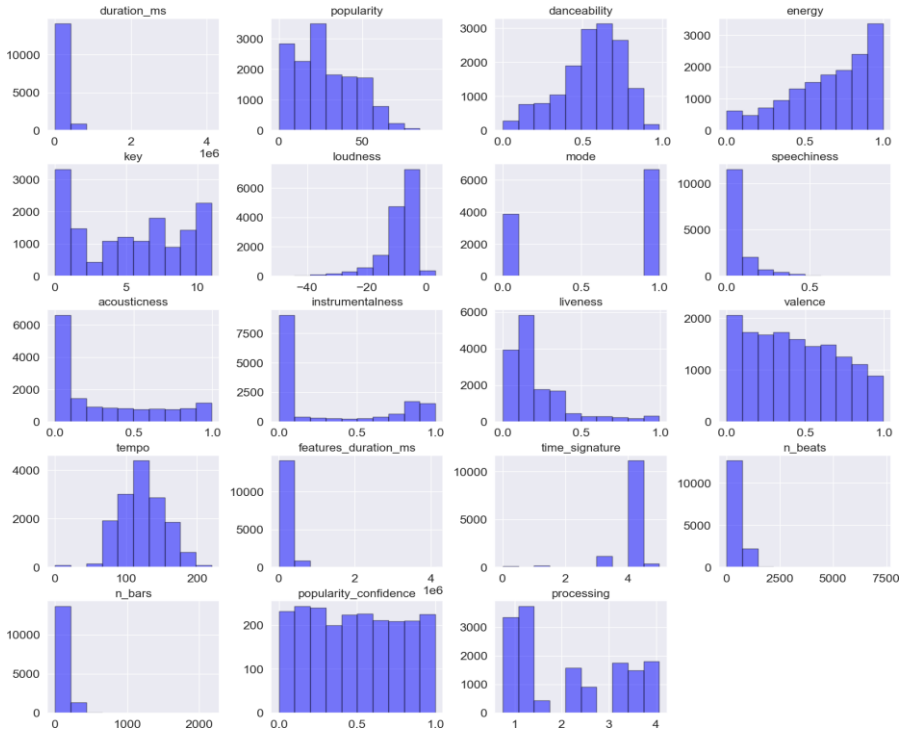

Figure 1.2. Histogram of the distribution of each attribute

For what concerns the positively skewed distribution, this means that the majority of the data values are concentrated on the left side, and there are relatively few larger values on the right side, while the negative skewed distribution represents the opposite situation, where the majority of data values are on the right side.

Talking about the normal distribution, also called bell-shaped distribution, it shows a distribution which is symmetric around its mean. This means that the values on the left and right sides of the mean are approximately equal.

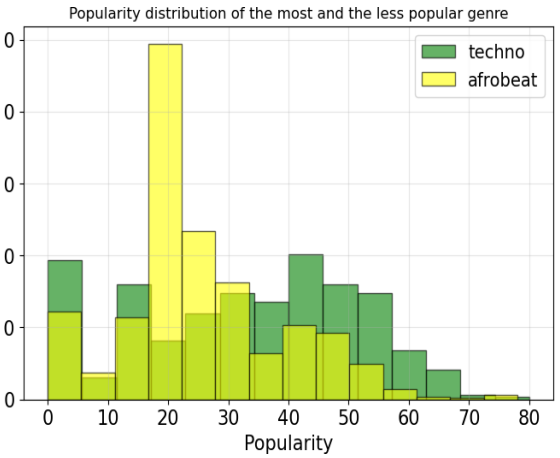

Figure 1.3

By using the aggregate operator, we counted the 'name' attributes grouped by genre and it showed that every genre has 750 songs.
At this point, we grouped the rows by genres and we computed the popularity mean. Then, we calculated the most and the less popular genres by using min and max functions. Finally, we plotted an histogram with popularity distribution for the most and the less popular genres (Figure 1.3).

1.3. Assessing Data Quality

Ensuring data quality is paramount for informed decision-making and accurate analysis. Reliable data forms the bedrock of any successful project. Data's quality involves assessing its completeness, accuracy, consistency, reliability, uniqueness and timeliness. We assess our data quality by employing techniques to detect and address null values/missing values, duplicated and outliers.

1.3.1. Handling Missing Values

Attributes	Presence of Null_Value	No. Of Missing Values
duration_ms	FALSE	0
popularity	FALSE	0
danceability	FALSE	0
energy	FALSE	0
key	FALSE	0
loudness	FALSE	0
mode	TRUE	4450
speechiness	FALSE	0
acousticness	FALSE	0
instrumentalness	FALSE	0
liveness	FALSE	0
valence	FALSE	0
tempo	FALSE	0
features_duration_ms	FALSE	0
time_signature	TRUE	2062
n_beats	FALSE	0
n_bars	FALSE	0
popularity_confidence	TRUE	12783
processing	FALSE	0

Figure 1.4

Regarding the specific attributes with missing values—mode, time_signature, and popularity_confidence—it is evident from the figure 1.4 that 4,450 instances lack mode information, 2,062 instances have missing time_signature, and 12,783 instances lack popularity_confidence details. This observation underscores the significance of these missing values, signifying potential areas where data imputation or careful handling may be necessary for a more complete and accurate dataset. Identifying these specific columns with substantial missing data allows for targeted strategies to address these gaps, ensuring a more robust dataset for analysis and decision-making.

In order to detect duplicate values, we noticed that feature_duration_ms attribute has similar values to duration_ms attribute so we decided to drop that column.

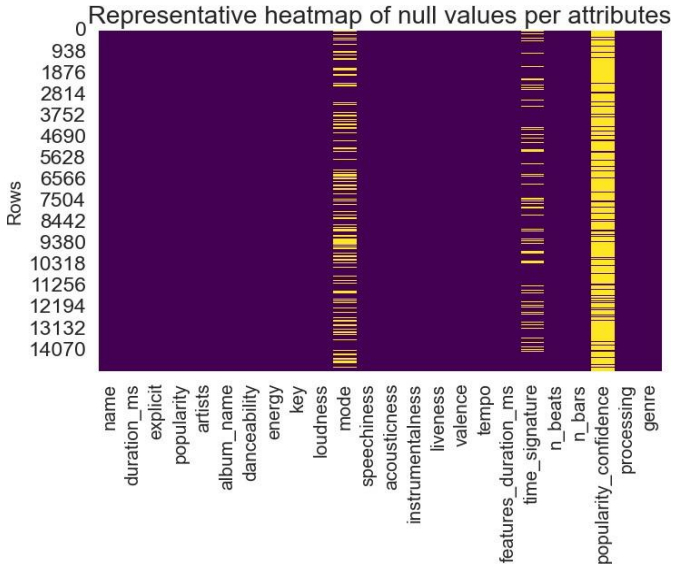

Figure 1.5

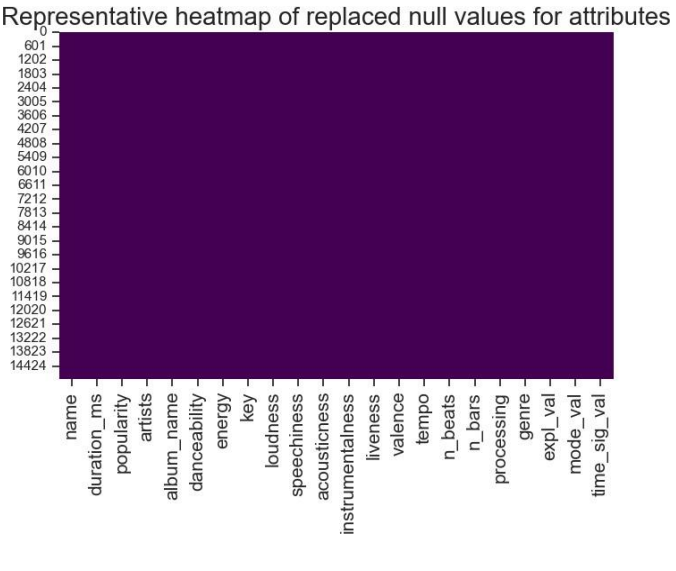

Figure 1.6

In Figure 1.5, the heatmap vividly illustrates the presence of missing values within our dataset. The heatmap showcases columns with substantial missing data, prominently highlighting attributes such as 'mode,' 'time_signature,' and 'popularity_confidence'. For what concerns the last attribute, we preferred to drop it definitely before beginning any kind of cleaning process, because of the very high number of its null values. To address these gaps, a meticulous data cleaning process was initiated, utilizing the median as a replacement for missing values due to its robustness, especially for numerical attributes. It's worth noting that while median imputation is often preferred for numerical data due to its resistance to outliers, mean and mode are alternative

imputation methods used for specific scenarios. Mean, representing the average of values, is favored for normally distributed data, while mode, denoting the most frequent value, is applied to categorical data or cases where values occur with high frequency. After data cleaning, in Figure 1.6 the heatmap showcases a marked improvement, indicating the successful replacement of missing values with medians, thereby enhancing the dataset's completeness for subsequent analysis.

We also tried to delete all the rows containing null values, but using this method the dataset would have been not very representative, so we decided to use the replacement strategy explained above.

1.3.2. Handling Outliers

Outliers in a dataset are data points that significantly differ from the majority of the other data points. Outliers can occur for various reasons, such as data entry errors, natural variation in the data, or the presence of anomalies. There are different ways to detect outliers, and common methods are Z-score, domain knowledge, IQR (Interquartile Range) and visual inspection. We used the last method and plotted the data using box plots, this helped us visually identify outliers which are placed outside the whiskers, and then we deleted the rows corresponding to the outliers values. For example, for what concerns the variable 'loudness', the outliers were located approximately between the values -18 and -50 (Figure 1.7), so we decided to delete all the rows contained in this range.

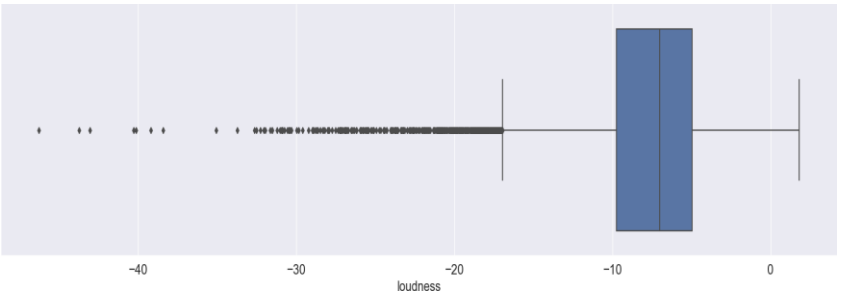

Figure 1.7. Boxplot of 'loudness' attribute

In this way, we tried to clean our dataset in order to improve the overall quality of our dataset. In fact, after deleting the outliers, the dataset still has some noise but is much more clean and has a more accurate representation of the majority of data. To prove this, Figure 1.8 displays the representation of the 'key' variable per loudness with the outliers, while Figure 1.9 the same boxplot after the elimination of the outliers in the 'loudness' attribute.

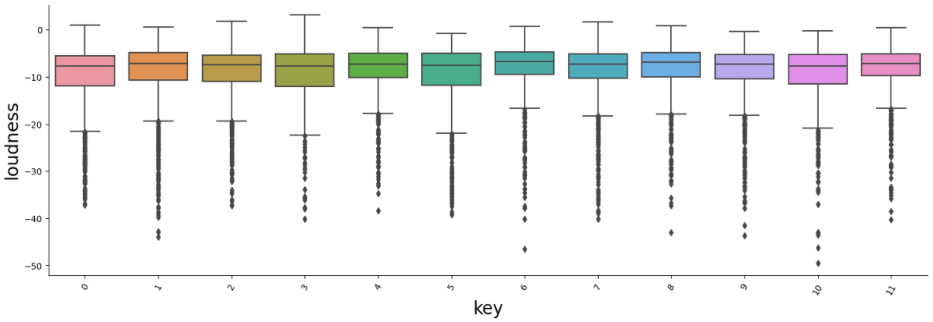

Figure 1.8

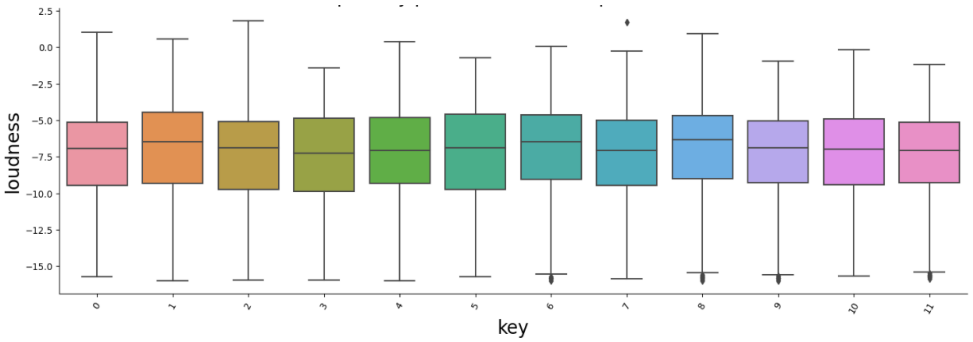

Figure 1.9

1.4. Variable Transformations

	explicit	expl_val
0	False	0
1	False	0
2	False	0
3	False	0
4	False	0
5	True	1
6	False	0
7	False	0
8	False	0
9	False	0

Figure 1.10. Label encoding

Variable transformation in data mining refers to the process of modifying or changing the scale, distribution, or form of a variable in a dataset. The goal of this process is to make the data more suitable for analysis and to improve the performance of a data mining model. This can be done in several ways, such as normalization, log transformation, label encoding, square root transformation, Z-score transformation, etc.

In our dataset there are some attributes which need label encoding, such as 'explicit', 'mode' and 'time signature'. So, for each attribute we transformed the categorical data into a numerical representation. For example, in the table in Figure 1.10 we created a new column 'expl_val' where the categorical values of 'explicit' (False and True) are replaced with corresponding numerical indices (0 and 1). This transformation is essential when working with machine learning algorithms that require numerical input, as many algorithms operate on numerical data.

In addition, we used the Min-Max scaler to normalize the dataset. This type of normalization technique specifically scales and transforms the values of a variable into a range which goes between 0 and 1. The purpose of this kind of normalization is to ensure that all features are on a similar scale, which can be important for certain machine learning algorithms that are sensitive to the magnitude of input features.

1.5. Pairwise Correlations and Eventual Elimination of Variables

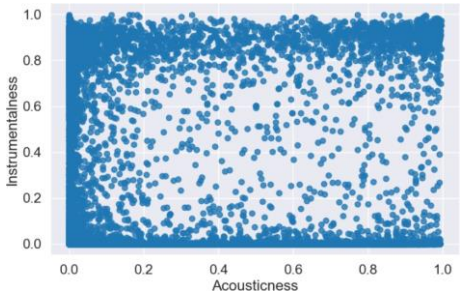

Figure 1.11

Databases might contain hundreds or even thousands of features and, most of the time, some of these features are useless, do not provide any additional information and can pollute our dataset. Also, there is an increased risk of overfitting, where the model performs well on the training data but fails to generalize to new, unseen data. Pairwise correlation measures the strength and direction of a linear relationship between two variables.

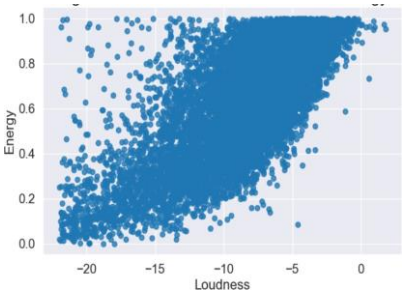

Figure 1.12

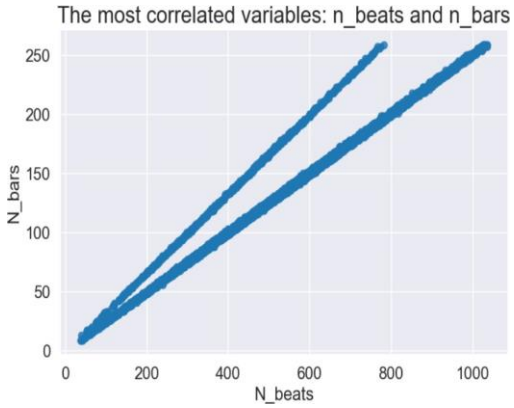

Figure 1.13

Using scatterplots, we checked the correlations between several pairs of features. According to the graphs, for example, it turned out that 'instrumentalness' and 'acousticness' have a very low correlation (Figure 1.11), while 'energy' and loudness' are highly correlated (Figure 1.12).

However, the most correlated features are 'n_beats' and 'n_bars' (Figure 1.13).

The heatmap in Figure 1.14 displays the correlation between each variable. The visualization allows us to understand how different variables relate to each other. This insight can be valuable in formulating hypotheses, guiding further analysis, and gaining a deeper understanding of the dataset. In order to understand the data from this heatmap, let's take into consideration the correlation coefficient between acousticness and speechiness, which is 0.55. This indicates that we can predict feature 2 (in this example, acousticness) 55% of the time starting from the value of feature 1 (speechiness). In other words, feature 2 will not add any information to the dataset if feature 1 already exists. So, it is useless to keep feature 2.

In fact, we excluded the feature 'n_bars' from the dataset because, as we've seen in Figure 1.13, it had a very high correlation with the 'n_beats' feature (0.98).

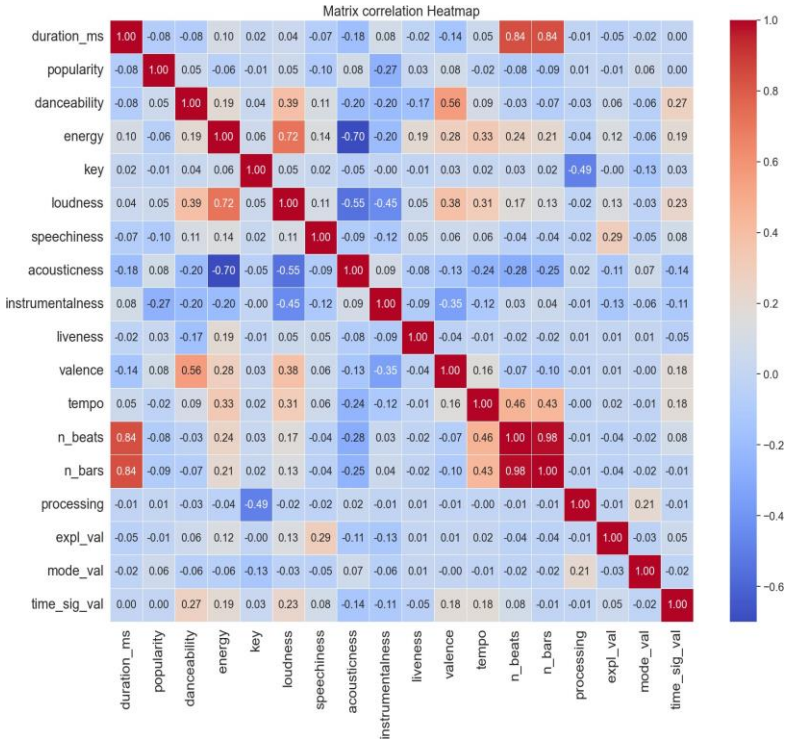

Figure 1.14 Correlation heatmap.

Chapter 2

2. Clustering

In the following chapter, we're going to apply three different clustering algorithms to our dataset: k-means, density base (DBScan) and hierarchical clustering.

2.1. Analysis by Centroid-based Method – K-means

Firstly, we have applied K-means clustering, a popular unsupervised machine learning algorithm used for partitioning a dataset into K distinct, non-overlapping subsets, called clusters. The goal is to group data points that are similar to each other while being dissimilar to those in other clusters. The number "K" represents the predefined number of clusters that the algorithm should create.
The K-means algorithm works through the following steps:

- Initialization: choose K initial cluster centroids randomly from the data points or using some other method.
- Assignment: assign each data point to the nearest centroid, forming K clusters.
- Update Centroids: recalculate the centroid of each cluster as the mean of all the data points in that cluster.
- Repeat: iterate steps 2 and 3 until convergence, which occurs when the centroids aren't changing significantly anymore or when a specified number of iterations is reached.

For analysis with k-means we have used the MinMaxScaler to normalize the original features.

2.1.1. Optimal K value

The choice of K value is very important, as it can impact the convergence and quality of the final clusters.

There are three common methods for initializing centroids, such as random initialization, manual initialization and K-means++ initialization, which is the one we used in our analysis.

This last method chooses the first centroid randomly and then selects subsequent centroids with a probability proportional to their distance from the existing centroids.

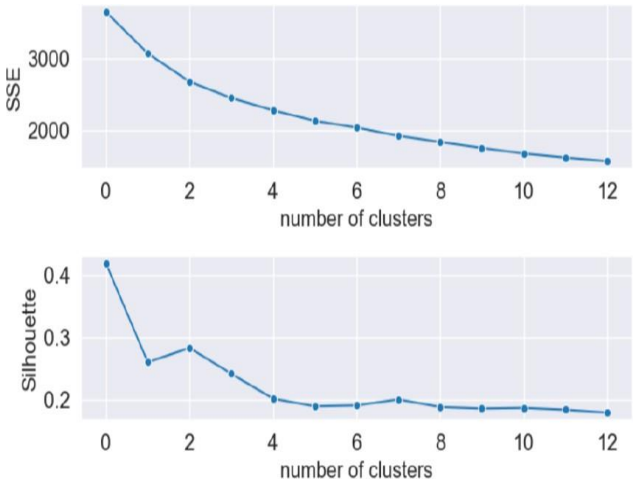

Figure 2.1

This helps in distributing the initial centroids more evenly. In fact, K-means++ tends to provide more stable results.

To choose the number of clusters, we run K-means for a range of values of K, starting from a minimum value of 2 clusters and gradually increasing it until 15. Then, we computed the SSE and the silhouette for each value

of K. By looking at the graph (Figure 2.1) it's evident a point where each curve starts to stabilize.
This is the "elbow" in the curve.
The elbow point is the value of K where adding more clusters does not significantly improve the model's fit.

```
for i in sse_list:
    print(i)
```

```
3639.606614282509
3065.1414416293437
2673.050369668814
2444.871179456006
2274.260195066594
2126.922304879595
2035.2150306655333
1920.7979128272534
1833.2436671780883
1750.9922993327991
1674.1566463099816
1618.1661726737418
1568.076545470555
```

```
for i in sil_list:
    print(i)
```

```
0.41849970934666597
0.2602923462069956
0.28317624830005533
0.24172679606463277
0.2018877710235353
0.1895989319864491
0.19101011530180664
0.20062307183316783
0.18846234375074508
0.18609610315790753
0.1870757415383829
0.18410256281729157
0.17946910502843366
```

Figure 2.2

At this point, we calculated the value of the SSE and of the silhouette for each value of K (Figure 2.2). Considering the SSE list of values, the optimal K would be at index 2, which is K=4, and looking at the silhouette list, the optimal value for K is at index 2 as well. For these reasons, we chose K=4, which also gets a quite high silhouette value, which is good as it indicates that the clusters are well-defined, distinct, and have a clear separation from each other.

Despite this, the value of the silhouette is not that satisfying.

2.1.2. K-Means with Selected Features

We did a line plot to visualize the cluster centers obtained from the K-means clustering algorithm. Each line in the plot represents the centroid for a particular cluster.

The graph provides a visual representation of how the cluster centers differ across the features for each cluster in the K-means clustering results. Each line shows the trend of the feature values within a specific cluster. As we can see in Figure 2.3, for some of the features the clusters have the same centroids. In this case, in which some lines represent clusters overlapping, it means that certain clusters are closely related or share common characteristics.

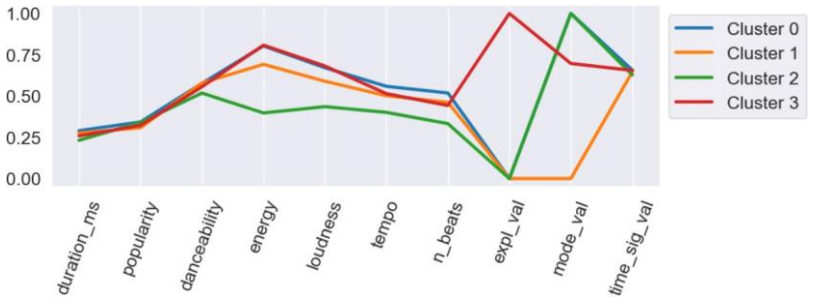

Figure 2.3

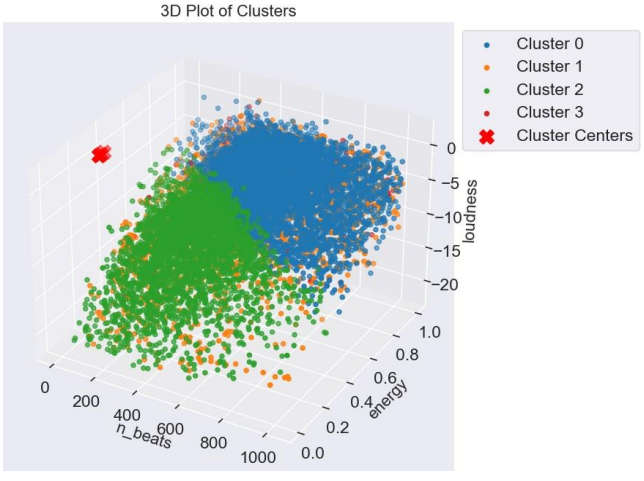

Figure 2.4. 3D plot of clusters

Looking at Figure 2.3, we can say that the clusters initially have a good distribution, with an homogeneous and uniform trend. Then, all of them present discontinuous values corresponding to the "expl_val" and "mode_val" features, whose attributes are 0 and 1.

Then, we just took into account the three variables 'n_beats', 'energy', 'loudness'.
By visualizing the data in 3D (Figure 2.4), we can see how separated the clusters are based on these three

features. The goal of clustering is to group similar data points together, and the plot shows how the algorithm has achieved this grouping in a three-dimensional space.

In our case, the clusters are not well-defined, this could be due to the nature of the dataset itself, and clustering algorithms may struggle to find distinct groups.

2.2. Analysis by Density-based Clustering – DBScan

For analysis with DBScan we have used the same MinMaxScaler to normalize the original features and we have tried DBScan with selected features as well as the selected features of the K-means algorithm. In this case, we decided to not use The 'Elbow method' to determine the optimal number of clusters in a dataset just because DBSCAN defines clusters based on the density of data points rather than assuming a specific number of clusters, by grouping together data points that are close to each other in dense regions and marking the isolated points as outliers. In order to get a suitable *eps* (ε) and *minPts* before applying the DBScan clustering algorithm, we visualized the result of various scatter plots by adjusting different values for the requested parameters until we will obtain meaningful clusters. We used an iterative approach by adding every time 0.1 to eps value till 0.5 and by doubling the min_samples till 20.

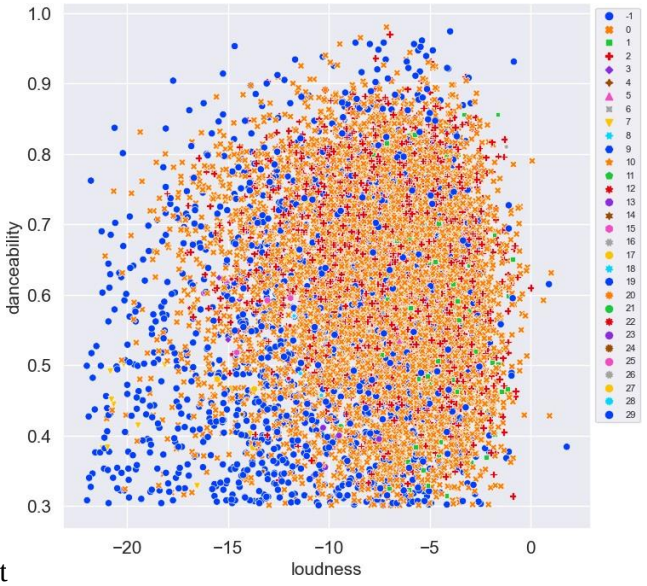

Figure 2.5

In the first place, we began by setting the eps value at 0.2 e the min_samples at 5: were produced 29 clusters, a Silhouette score of -0.22 (a score of -0.16 counting silhouette with reference to noise cluster -1), and 1502 data points that are regarded as outliers or noise. Keeping in mind the coefficient scale, this Silhouette score was not acceptable at all.

With the eps value at 0.3 and the min_samples value at 10, what we got was a Silhouette score of 0.07, 8 clusters and an amount of 475 points considered as outliers or noise. Even though the Silhouette score was still not satisfactory, the number of outliers/noise has decreased drastically.

With eps value to 0.4 and the min_samples to 15 and we received totally different results with an impressive Silhouette score of 0.42 (silhouette score of 0.43 with reference to noise cluster -1), 5 clusters and 95 as noise value.

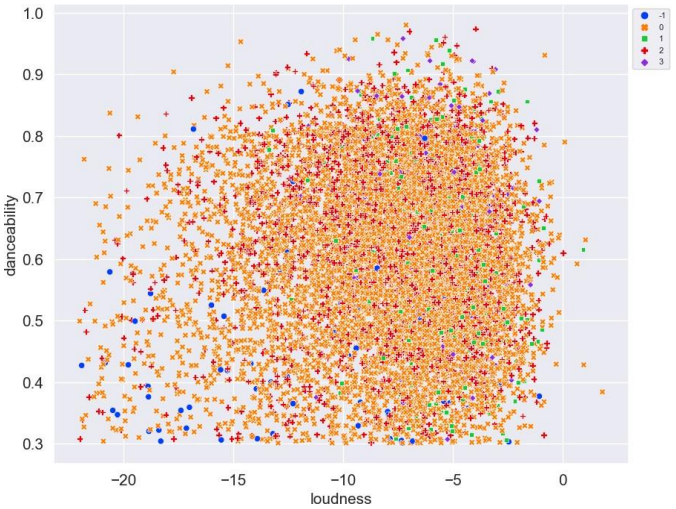

Figure 2.6

By keeping the same eps value (0.4) and by modyfing only the min_samples to 10, the Silhouette score and the number of clusters kept the same as before and the noise value decreased from 95 to 78.

We made one last attempt by increasing again the eps value to 0.5 and the minimum point value to 20 and again we didn't notice any change in the Silhouette score or in the number of clusters but only a huge reduction of noise points was produced, from 78 to 14.

In order to avoid being insensitive to smaller or less dense clusters - setting a too high value for min_samples - or considering some points as noise - setting a too small value for eps value - we found that an acceptable result was achieved by the eps at 0.4 and the minimum points at 10 (as displayed in the Figure 2.6) which gave us a Silhouette score of 0.42 and 5 number of clusters.

2.3. Analysis by Hierarchical Clustering

For analysis with hierarchical clustering, we have used the same MinMaxScaler and the same selected features we have used in our previous clustering algorithms. We employed Agglomerative (Bottom to Top) technique to form clusters and Euclidean metric to calculate distance between data points. In agglomerative clustering individual data points are iteratively agglomerated into higher-level clusters. Each single point constitutes a cluster and, subsequently, more and more points gradually merge together building increasingly populated clusters: the method stops when a certain number of clusters is reached. We decided to use an iterative approach to find out the best number of clusters to achieve a satisfying score of Silhouette.

At the beginning, a dendrogram was created and we decided to set the number of clusters to 3 to check the estimated score of Silhouette: we got 0.42 using the *complete* linkage method, which was a good score in terms of dimensionality of our dataset. Then, using the same method, we increased the n_clusters in a range of clusters from 3 to 20 and, except the first try, the most acceptable result of the Silhouette score was 0.41 by setting n_clusters to 4. In all the other attempts, the score was incredibly low.
In order to check if we can get better results, we decided also to change our linkage method with ward's and average method - the last one uses Manhattan metric - but unfortunately these two methods did not perform very well, obtaining 0.24 and 0.15 of Silhouette score respectively.

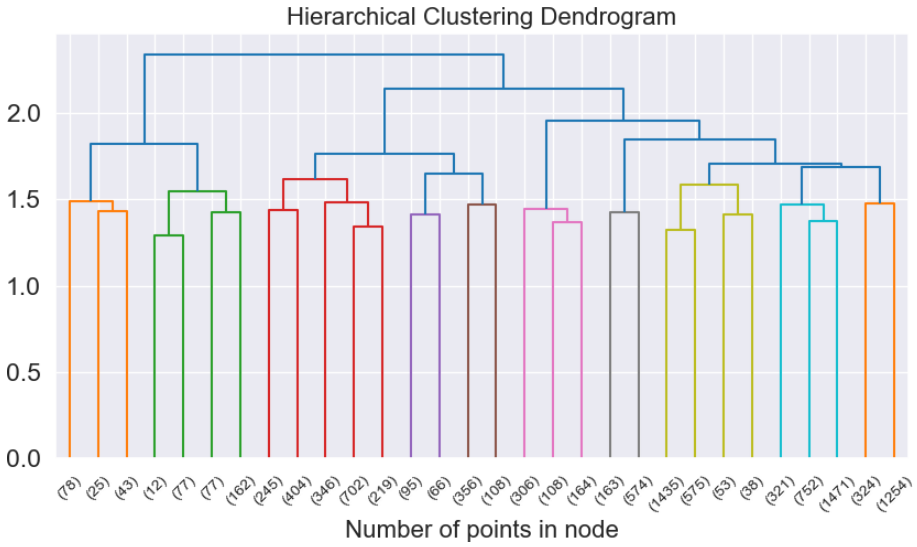

Figure 2.7

Figure 2.7 displays clustering dendrogram for 4 clusters using complete linkage method and a Silhouette score of 0.41.

2.4. Final discussion

After having implemented three different clustering algorithms, we had to accomplish the last part of cluster validation, that is choosing which algorithm best describes our Spotify dataset. In order to do this, we evaluated results using the analytical method of the Silhouette score computation.

By describing it with four values after the comma, for K-means we got 0.2650, for DBSCAN we got 0.4252 and for Hierarchical one we got 0.4292.

Discharging the K-means algorithm a priori for the Silhouette score, and considering the other two Silhouette scores, the Hierarchical is the best algorithm to describe our dataset. Although this, we think that every cluster's Silhouette score is definitely low and that the whole dataset does not lend well to any kind of clustering.