Data Mining I Project

A Comprehensive Data Mining Analysis on a Spotify Dataset

Fabio Melasi, Giulio Leonardi, Matilde Contestabile

A.A. 2023/24

UNIVERSITÀ DI PISA

Contents

1 Data Understanding and Data Preparation

The first section of this chapter is devoted to data understanding, where various analyses are conducted to provide a thorough overview of our dataset. Building upon the insights gained during this initial phase, the data preparation section outlines the strategies employed to adapt the dataset to better suit our research needs. In the final section, we present our ultimate findings and conclusions.

1.1 Understanding the data

The groundwork for effective data mining tasks lies in a comprehensive understanding of the dataset at hand. This section begins by introducing the Spotify dataset selected for this research. Specifically, in Table 1 we present a description of the 24 features that define our dataset and in Table 2 we provide key statistics for our numerical attributes. These two tables will help us in formulating our initial hypotheses. Furthermore, in Section 1.1.1 we propose some charts in order to extract insights through data visualization. For the same purpose, Section 1.1.2 demonstrates the correlations among the numerical features within our dataset. Finally, in Section 1.1.3 we identify and quantify the presence of outliers in our data.

Attribute name	Description	Datatype	Example values
name	Name of the track	String	*Untrue*, *High Roller*
duration_ms	Track length in milliseconds	Integer	*206826*, *329733*
explicit	Whether the track's lyrics is explicit or not	Bool	*True*, *False*
popularity	How much the track is popular in a 0-100 range	Integer	*32*, *41*
artists	Track's authors name, separated by ";"	String	*Coal Chamber*, *The Crystal Method*
album_name	The album name in which the track appears	String	*Chamber Music*, *Vegas*
danceability	How much a track is suitable for dancing, in a 0.0-1.0 range	Float	*0.383*, *0.464*
energy	How much a track is perceived to be intense and active, in a 0.0-1.0 range	Float	*0.951*, *0.579*
key	The key the track is in, using standard Pitch Class Notation (0-11 range)	Integer	*0*, *5*
loudness	The overall loudness of a track in decibels	Float	*-3.743*, *-9.136*
mode	Indicates the modality of a track, Major (1) or Minor (0)	Integer	*0*, *1*
speechiness	Detects the presence of spoken words in a track, in a 0.0-1.0 range	Float	*0.1040*, *0.0596*
acousticness	A confidence measure of how much a track is acoustic, in a 0.0-1.0 range	Float	*0.006070*, *0.281000*
instrumentalness	Predicts the absence of vocals in the track, in a 0.0-1.0 range	Float	*0.827000*, *0.318000*
liveness	Detects the presence of an audience in the recording, in a 0.0-1.0 range	Float	*0.2610*, *0.0992*
valence	Describes the musical positiveness of a track, in a 0.0-1.0 range	Float	*0.6680*, *0.1400*
tempo	The track's estimated tempo in beats per minute	Float	*110.584*, *171.752*
features_duration_ms	Track length in milliseconds	Integer	*206827*, *329733*
time_signature	An estimated time signature	Integer	*4*, *3*
n_beats	Track's number of beats	Integer	*385*, *935*
n_bars	Track's number of bars	Integer	*96*, *235*
popularity_confidence	The confidence of the popularity of the song, in a 0.0-1.0 range	Float	*0.973*, *0.698*
processing	Unclear	Float	*3.349*, *2.367*
genre	The genre in which the track belongs	String	*industrial*, *breakbeat*

Table 1: Introduction of features

Table 1 provides an immediate source of valuable insights upon initial examination.

- Feature `duration_ms` appears to convey the same information of `features_duration_ms`.
- Features `duration_ms`, `n_beats` and `n_bars` appear to convey similar information.
- Feature `tempo` can be obtained by combination of `duration_ms` and `n_beats`.
- Feature `time_signature` consists of integers representing the numerator of a time signature fraction. For instance, a value of *3* corresponds to a 3/4 time signature, while *4* signifies a 4/4 time signature, and so forth. If the value is 0, it indicates that the track is not traditional music but, rather, may be nature sounds or similar non-musical content.
- Up to this point, we have no clue on what `processing` feature indicates.

Furthermore, it is worth highlighting two significant points: firstly, there are no duplicate data instances, and secondly, the 15,000 instances in our dataset are evenly distributed across 20 distinct music genres.

Before diving into the world of charting, it is essential to gain a comprehensive understanding of the statistical characteristics of our numerical features. This overview is thoughtfully provided in Table 2.

Feature	Type*	Mean	Median	Mode	STD	Range	MV**

duration_ms	C	246807.48	227826.0	180000	127994.055	8586 - 4120258	0
popularity	D	27.424	24.0	0	18.588	0 - 94	0
danceability	C	0.551	0.58	0.0	0.194	0.0 - 0.98	0
energy	C	0.656	0.709	0.961	0.264	0.0 - 1.0	0
key	D	-	-	7	-	0 - 11	0
loudness	C	-8.895	-7.303	-8.207	6.006	-49.531 - 3.156	0
speechiness	C	0.084	0.051	0.0	0.087	0.0 - 0.939	0
acousticness	C	0.304	0.155	0.994	0.33	0.0 - 0.996	0
instrumentalness	C	0.287	0.00313	0.0	0.383	0.0 - 1.0	0
liveness	C	0.217	0.131	0.11	0.195	0.0 - 0.994	0
valence	C	0.437	0.416	0.0	0.277	0.0 - 0.995	0
tempo	C	123.117	124.188	0.0	31.931	0.0 - 220.525	0
features_duration_ms	C	246794.684	227818.5	180000	127984.991	8587 - 4120258	0
time_signature	D	-	-	4.0	-	0 - 5	2062
n_beats	C	501.862	461.0	0.0	280.69	0.0 - 7348.0	0
n_bars	C	128.393	117.0	97.0	75.114	0.0 - 2170.0	0
popularity_confidence	C	0.49	0.48	0.083	0.291	0.0 - 1.0	12783

Table 2: Statistics on numerical features, considering all the 15000 instances.
* C for Continuous, D for Discrete
** MV for Missing Values

Although deriving conclusive findings from the numerical matrix presented in Table 2 may be challenging, there are still notable observations that can be made:

- Features `time_signature` and `popularity_confidence` exhibit missing values. Also the `mode` feature introduced in Table 1 records 5911 instances of missing data. In Section 1.2.2, we will elaborate on our methodology for addressing this issue.
- It is not clear why the `loudness` feature, which should be measured in decibels, displays negative values. In any case, data shows that for actual low volume music corresponds a low value of the feature and vice versa.
- Statistical outcomes relative to `duration_ms` and `features_duration_ms` reinforce the suspicion that these two features convey the same information. The examination of the correlation matrix may help dispel any uncertainties.

Up to now, we are not considering the presence of outliers. We will quantify them in Section 1.1.3.

The `processing` feature. After an analysis aimed at understanding the significance of the `processing` feature, it has become clear that its values are entirely contingent upon the values of the `key` feature. In essence, while the inherent meaning of this feature may still be obscure, we have observed a direct correspondence between each `key` value and its corresponding `processing` value. As a result, it is logical to contemplate its exclusion from the dataset since it does not provide any distinctive or original information.

1.1.1 Data visualization

In this section, we introduce the phase of data visualization. By generating a variety of charts, we conducted an initial examination of record distributions and the relations between different attributes. The aim is to formulate preliminary insights and hypotheses regarding the dataset through visual depictions.

To begin, we created histograms for continuous attributes to assess their distribution. Upon a thorough analysis of the charts, we observed that the records displayed a normal distribution for the **danceability** and **tempo** attributes. Hence, one of the resulting visualizations is showcased for your consideration.

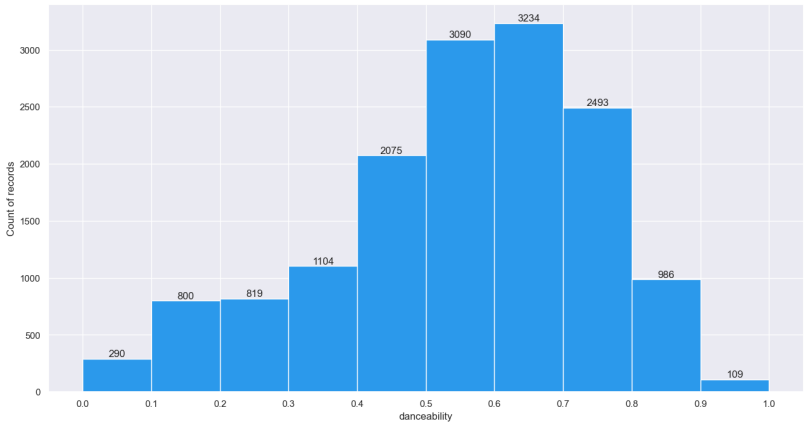
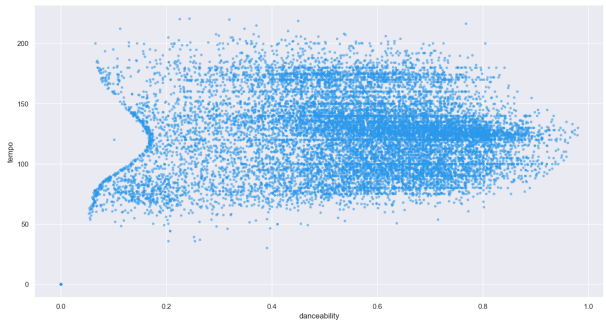
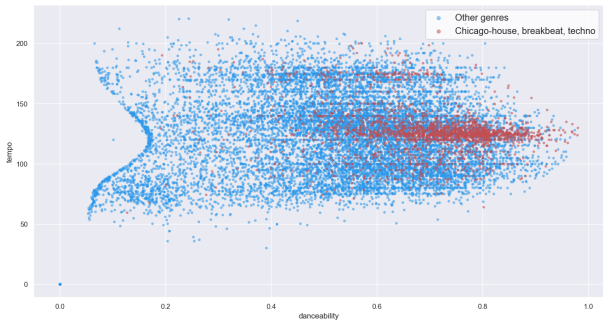


Figure 1: Histogram depicting the distribution of records for the **danceability** attribute.

Subsequently, scatter plots were generated for each pair of continuous attributes. These plots revealed potential correlations between attributes, which will undergo quantitative analysis in the next section. Additionally, the graph depicting the relationship between the **danceability** attribute and the **tempo** attribute exhibited a distinctive distribution, as illustrated in the forthcoming figures.



(a) Scatter plot depicting the distribution of records for the **danceability** attribute and the **tempo** attribute.



(b) Distribution of records for the **danceability** attribute and the **tempo** attribute with highlighted genres.

Figure 2: Compared scatterplots.

As Figure 2a illustrates, there is a noticeable trend where songs tend to exhibit higher **danceability** in the vicinity of 120/130 bpm. This observation led to an intriguing insight: within this **tempo** range, it appears that songs from genres traditionally associated with "danceability" may tend to aggregate. Subsequently, additional scatter plots were generated for the **danceability** and **tempo** attributes, with each plot exclusively featuring records belonging to a specific genre. The objective was to investigate the composition of the graph depicted in Figure 2a in relation to the **genre** attribute.

The findings revealed that a substantial portion of songs belonging to the *chicago-house*, *breakbeat*, and *techno* genres exhibit high **danceability** and fall within the 120/130 bpm range. For better clarity, Figure 2a has been enhanced by introducing the color red to emphasize the records from the *chicago-house*, *breakbeat*, and *techno* genres, while the records from the remaining genres stayed in blue.

Hence, according to the visual analysis of Figure 2b, the hypothesis that the 120/130 bpm range is associated with genres that are notably more danceable than others in the dataset appears plausible.

1.1.2 Correlation Analysis

This section presents an extensive examination of the correlations between all numerical variables within our dataset. The analysis aims to reveal how these attributes interact and whether they exhibit significant associations. To gain a comprehensive perspective on these relationships, we constructed a *correlation matrix* that quantifies the strength and direction of associations. The magnitude of the correlation coefficients reveals the *degree of dependence*, while the sign (positive or negative) indicates the *direction* of the relationship.

The matrix's size (17x17) precludes us from displaying it here[1]. Therefore, we will focus on summarizing the most interesting[2] results in the upcoming list.

[1] The original matrix can be consulted at this *link*

[2] We consider correlations with values surpassing 0.5 as the criteria for what we find "interesting"

Key observations

- `duration_ms` presents a full correlation (0.999) with `features_duration_ms`. This confirms our hypothesis, leading us to consider eliminating one of the two. Additionally, a strong correlation exists between `n_beats` (0.847) and `n_bars` (0.844): the similarity arises from these two features being fully correlated with each other (0.984). Given that (as we mentioned earlier) `n_bars` values can be obtained by combination of `duration_ms` and `tempo`, it is reasonable to eliminate both `n_bars` and `n_beats` features.
- `danceability` presents a moderate correlation (0.560) with `valence`. This suggests that more positive songs are often associated with music that is easier to dance to.
- `energy` presents both a positive correlation (0.721) with `loudness`, and a negative correlation (0.699) with `acousticness`. This is immediately understandable and is connected to the negative correlation (-0.553) between `loudness` and `acousticness`.

While these correlations are relevant to the entire dataset, certain ones only emerge when we consider specific music genres. For example:

- A correlation between `energy` and `valence` exists within the `bluegrass, disney` and `indian` music genres.
- A correlation between `energy` and `liveness` exists within the `sleep` genre.
- A correlation between `loudness` and `instrumentalness` exists within the `indian, disney, forro` and `mandopop` music genres.

1.1.3 Outliers

We produced a boxplot for every continuous feature of the dataset in order to analyze its distribution and quantify the outliers. We show a couple of examples below.

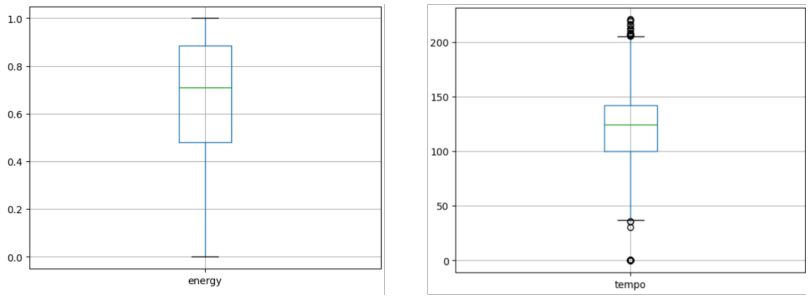


Figure 3: Boxplot of **energy** and **tempo** features.

According to our analysis, there are 7 attributes presenting outliers, indicated in Table 3 below.

duration_ms	danceability	loudness	speechiness	liveness	tempo	popularity
616	104	1004	1679	1149	113	5

Table 3: Number of outliers.

Section 1.2.3 will present the method we used to manage outliers.

1.2 Preparing the data

With a clear grasp of our dataset's characteristics, we can now outline the necessary data adjustments to enhance the quality of our subsequent analyses. Specifically, in Section 1.2.1 we identify redundant features that can be safely removed without loss of information. Then, the following two sections elaborate on our solutions for addressing missing values and outliers. The last section provides an explanation of our data normalization process.

1.2.1 Feature Selection

Considering the analysis conducted in the previous section, we find reasonable to eliminate from our dataset the following features:

- `features_duration_ms, n_beats` and `n_bars`, because of their strong correlations, as described in Section 1.1.2.
- `processing` because of its dependence on `key` feature, as described in Section 1.1.
- `popularity_confidence` because of its large number of missing values (see Table 2).

1.2.2 Missing Values

The dataset displays missing values in three distinct attributes: `popularity_confidence` (12783 missing values), `mode` (4336 missing values), and `time_signature` (2032 missing values). Due to the substantial number of missing values, `popularity_confidence` was eliminated from the dataset. For the other two attributes, we opted to replace the missing values. To handle missing values, we developed an algorithm to assign attribute values based on record similarity (limited to continuous attributes only).

Given *v_mv* (a vector with a missing value), the algorithm calculates the similarity between *v_mv* and each *v* (a vector without missing values) in the dataset. All vectors *v* are sorted by similarity, and the algorithm retains only the top 200 (i.e., the 200 most similar vectors to *v_mv*). Subsequently, a list of 200 similarity-output pairs is created. The algorithm groups these pairs by output value (as the attributes to be replaced are both categorical) and sums the similarities for each group. The output with the highest sum is then assigned to *v_mv*.

1.2.3 Outliers

The presence of outliers in a dataset can lead to a reduced efficiency in task execution. Simultaneously, extensive record removal can negatively impact algorithms. In order to balance the number of outliers with the number of records, we have employed a rule that determines whether an instance should be removed: we remove every instance that contains outliers for more than two attributes, with consideration given to the attributes `danceability, loudness, speechiness, liveness` and `tempo`, excluding `duration_ms` and `popularity`.
Using this approach, we have identified 387 instances eligible for elimination.

1.2.4 Normalization

In order to enhance the performance of our clustering and classification algorithms, the final step involves data normalization. Specifically, after removing the selected columns (features) and rows (outliers instances), and filling missing values, we applied the **MinMax algorithm** to the numerical continuous features.

1.3 Conclusions

The Data Understanding and Preparation stages have yielded their intended outcomes, resulting in a **refined, cleansed, and normalized dataset** that will now take center stage in our future analyses, along with **insights** that hold the potential to significantly shape our research direction. Notably, we have observed strong correlations between specific attributes that may indicate potential underlying relationships deserving further investigation. Additionally, the consistent attributes observed within distinct musical genres offer a compelling opportunity for in-depth scrutiny, suggesting genre-specific features and trends that warrant closer examination.

2 Clustering

In this section, we extend our data exploration by employing clustering to uncover and understand the underlying patterns and structures in our dataset. We investigate a range of clustering algorithms and techniques to determine their compatibility with our dataset. Our primary goal is to evaluate their performance, identify the most appropriate approach, and grasp the implications and significance of the resulting clusters.

2.1 Clustering by Centroid-Based Methods

In this subsection, we focus on well-known centroid-based clustering methods, specifically K-Means and Bisecting K-Means. We discuss the process of selecting attributes, determining the optimal number of clusters (k), and interpreting the resulting clusters. The analysis aims to shed light on the effectiveness and limitations of these algorithms in our dataset.

2.1.1 K-means

For K-Means clustering, we filtered the dataset to exclude categorical attributes, retaining only numerical continuous attributes. We then attempted to identify an optimal value for k using the *Elbow method* and the *Silhouette coefficient analysis*. However, as demonstrated by the accompanying graphs, the results did not reveal a standout k value to indicate clear and distinct clusters.

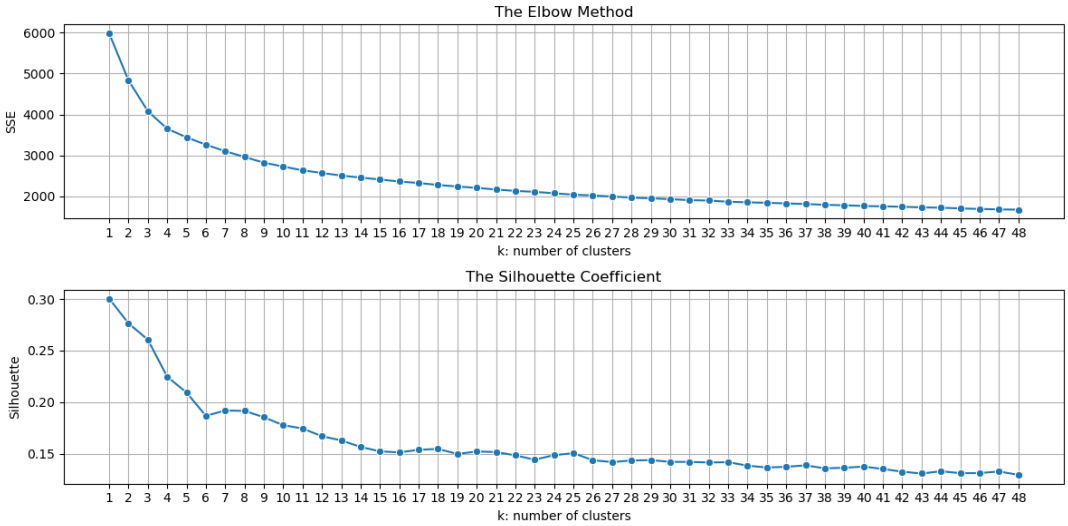


Figure 4: Comparing the Elbow Method to Silhouette Analysis for optimal k exploration

To gain a deeper understanding, we conducted clustering with low k values (e.g., k=3 and k=4) and generated graphs to explore how data points are distributed in a two-dimensional space based on pairs of variables. This analysis revealed a generally high degree of data uniformity and the absence of a distinct globular structure. Nevertheless, in Figure 5 we present the graphs in which the clusters were most pronounced, highlighting noteworthy clustering patterns.

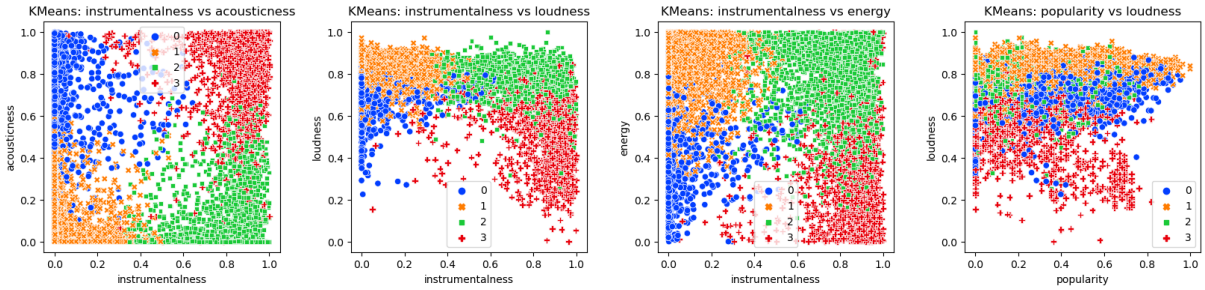


Figure 5: The scatter plots in which the clusters were most distinct using K-Means

Seeking further insights, we performed additional clustering based on these specific attribute pairs and increased the value of k to observe the outcome. The graphs in Figure 6 depict the results of this approach applied to the case study of the pair (`instrumentalness`, `acousticness`).

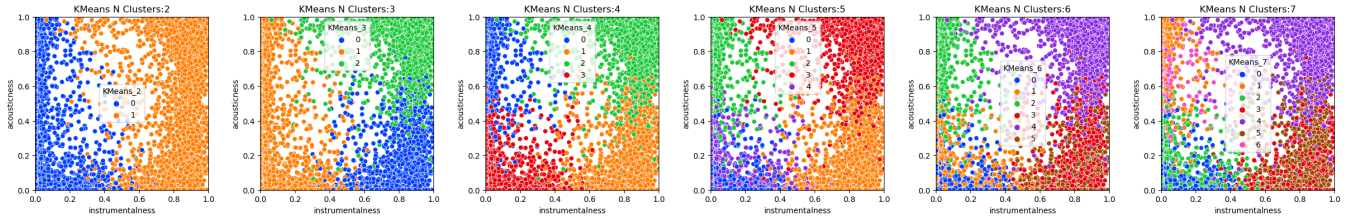


Figure 6: The outcome of K-means clustering on (`instrumentalness`, `acousticness`) using different k values

2.1.2 Bisecting K-means

For Bisecting K-Means clustering, we employed the same preprocessed dataset as in K-Means and opted for the *"biggest_inertia"* bisecting strategy. Since our dataset is relatively small, computational cost savings were not a primary concern, making this strategy an apt choice.

As in K-Means, we then conducted the Bisecting K-Means clustering analysis across a range of k values to assess their consistency as we altered the number of clusters (k).

The following graphs serve as empirical evidence, facilitating a direct comparison with the K-Means results.

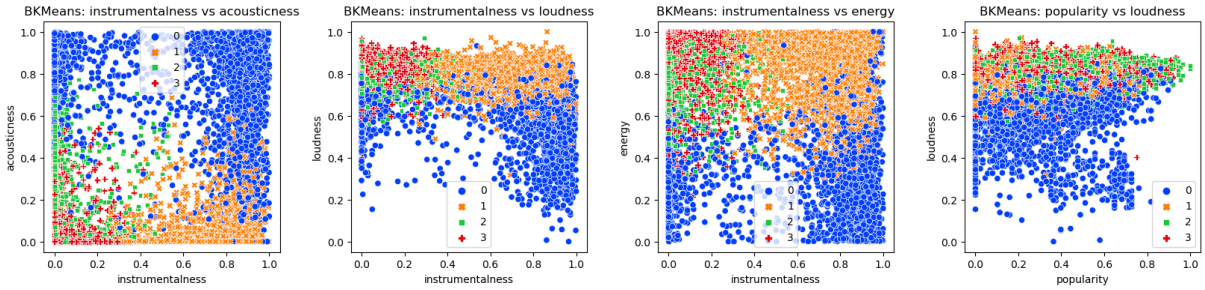


Figure 7: The scatter plots in which the clusters were most distinct using Bisecting K-means

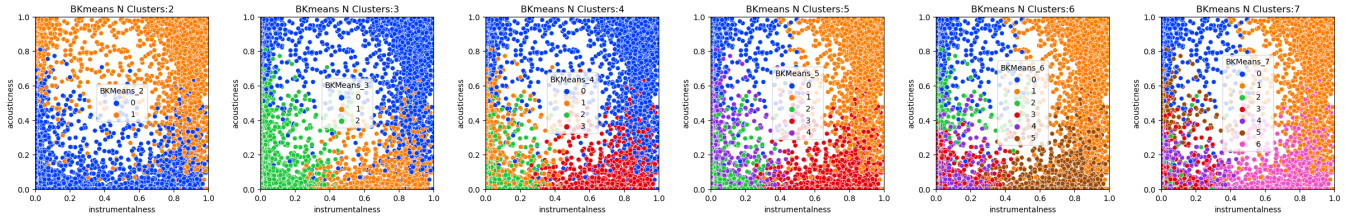


Figure 8: The outcome of Bisecting K-means clustering on (`instrumentalness, acousticness`) using different k values

2.2 Clustering by Density-Based Methods

This subsection is dedicated to density-based clustering techniques, primarily DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and OPTICS (Ordering Points To Identify the Clustering Structure). We investigate the identification of suitable parameter configurations and the interpretation of clusters. The discussion aims to uncover insights into the performance of density-based methods on our dataset.

2.2.1 DBSCAN

The first step in using the DBSCAN clustering algorithm is to determine the values for the *min_points* and *eps* parameters. We conducted an assessment for *min_points* with values set at 4, 8, 16, 32, 64, 128, and 256. The kth-neighbor distance method was employed to find a suitable *epsilon* (euclidean) distance for each of these values.

Through a visual analysis of the plots, we aimed to identify the "knee" point, which would provide an *epsilon* distance for each possible *min_points* value.

min_points:	4	8	16	32	64	128	256
epsilon:	0.37	0.41	0.43	0.49	0.54	0.6	0.68

Table 4: Pairs of *min_points-epsilon* used for clustering with DBSCAN.

Following this, clustering experiments were conducted using DBSCAN for each combination of *min_points* and *epsilon*. The results consistently demonstrated a primary large cluster with approximately 15000 points and a low number of outliers. It appears that, with these parameters, DBSCAN fails to provide meaningful insights into our hypotheses.

Subsequently, a second parameter selection phase was conducted, this time with a more arbitrary approach. The objective was to decrease the *epsilon* distance, aiming to decrease the count of core points in the dataset. This was done in pursuit of achieving denser clusters of considerable size, understanding that it would also lead the algorithm to identify more points as outliers.

Several experiments were conducted by gradually reducing *epsilon*, seeking significant-sized clusters. Halving the *epsilon* values yielded some promising results: experiments with *min_points* set to 64 and 128 produced three meaningful clusters.

The next step was to conduct further experiments to find the maximum epsilon that still produced the same three clusters mentioned earlier, with the aim of reducing the number of outliers. Following this analysis, the best *min_points-epsilon* combination found was 128-0.323, which yielded the same three clusters (exposed below) with a silhouette value of 0.335.

C0	**C1**	**C2**	**outliers**
8426	2088	589	3510

Table 5: Clusters identified by DBSCAN with *min_points*=128 and *epsilon*=0.323.

Subsequently, scatter plots were created for each pair of attributes, color-coding the records by their cluster membership, with the aim of understanding if they exhibited any interesting patterns in two-dimensional spaces. Below, there are the scatter plots that have exhibited the most interesting distributions.

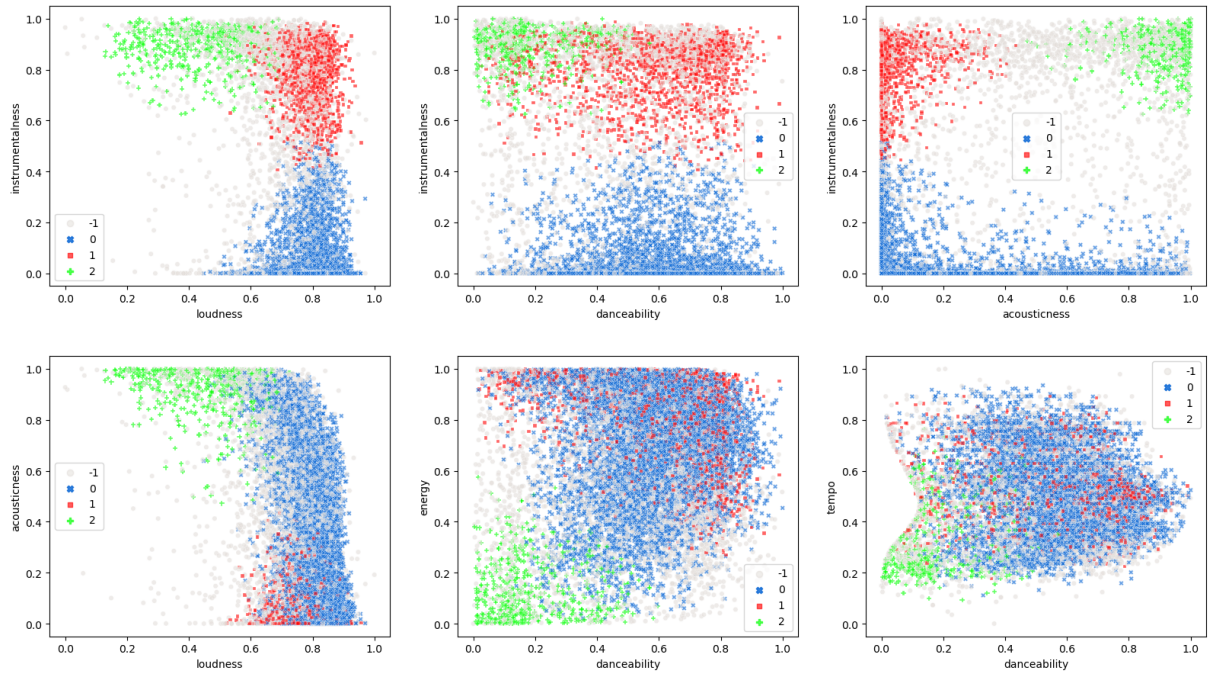


Figure 9: Scatterplots with highlighted clusters (*0*, *1* and *2*) and ouliers (labeled as *-1*)

The visual examination of the plots resulted in the following observations:

- C0: the least characterized cluster, shows high **loudness** and low **acousticness**; it seems scattered across other attributes.
- C1: appears to group highly danceable, noisy songs with low **acousticness** and high **instrumentalness**.
- C2: appears to contain songs that are more "relaxing" (low **loudness** and **danceability**) and have high values of **instrumentalness** and **acousticness**.

Furthermore, the scatter plot between the attributes **danceability** and **tempo** (second row, third column in Figure 9) indicates that C1 partially corresponds to the chart shown in Figure 2a (during the data visualization phase). This relates to the three genres (`chicago-house`, `breakbeat`, and `techno`) within the range of 120-130 bpm, which appear to be more danceable compared to the others.

In conclusion, the distribution of different genres within each cluster was analyzed. C0 does not appear to be predominantly composed of specific genres. On the other hand, C1 is primarily composed of records belonging to the genres *techno* (19%),

Chicago-house (15%), *black-metal* (12%), and *breakbeat* (12%). Lastly, C2 is predominantly composed of the genres *sleep* (31%), *Iranian* (27%), *Disney* (19%), and *IDM* (13%).

2.2.2 OPTICS

The OPTICS clustering algorithm was employed to seek additional insights compared to the clustering performed with DBSCAN, given its capability to identify clusters with varying densities.

Clustering experiments were conducted using the same *min_points* parameter values as those tested with DBSCAN (4, 8, 16, 32, 64, 128, 256). For each of these experiments, the resulting clusters and reachability plots were analyzed.

The results revealed that OPTICS assigns all points as outliers (except for micro-clusters) for experiments with $min\_points \leq 32$, while all points are assigned to a single cluster for other experiments ($min\_points \geq 64$). Therefore, no insightful information was found for this study.

Finally, by performing clustering with *min_points=128* and drawing a horizontal line at the epsilon distance used in DBSCAN, we observed that the three clusters identified by the DBSCAN algorithm in the previous section can be easily discerned. The graph illustrating this is presented below.

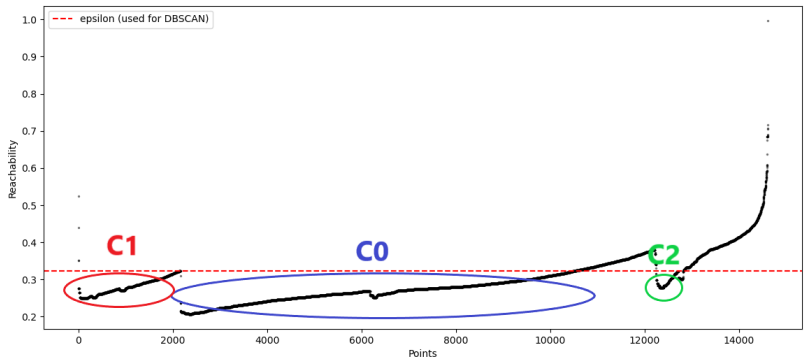


Figure 10: Reachability plot for *min_points=128*, with the *epsilon* threshold used for DBSCAN, and the resulting clusters manually highlighted (labels refer to clusters found with DBSCAN in section 2.2.1).

2.3 Hierarchical clustering

Finally, we explore hierarchical clustering, a method that organizes data into a hierarchical structure known as a dendrogram. We discuss the choice of distance functions and analyze multiple dendrograms. The goal is to gain a comprehensive understanding of hierarchical clustering's suitability for our dataset and the insights it provides.

2.3.1 Building Dendrograms

As an initial step in hierarchical clustering, we opted to construct a dendrogram for each distance function, utilizing Complete, Single, Average, and Ward linkage methods. Then, after carefully examining each dendrogram, we made the decision to "cut" at a specific level to yield a desired number of clusters. The resulting set of nested clusters is illustrated in Figure 11.

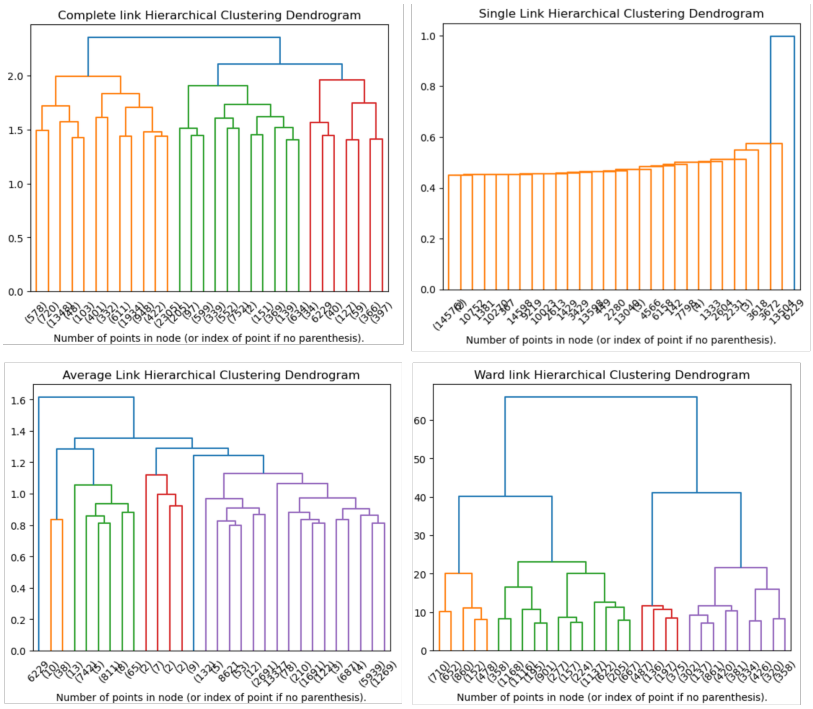


Figure 11: Dendrogram for every distance function

While the Complete Link and Ward methods allowed us to obtain respectively three and four sufficiently balanced clusters, the Average method and especially the Single Link method appeared to be inappropriate to organize our data into meaningful clusters.
These insights are confirmed by looking at the Silhouette values obtained for each linkage method and resumed in Table 6.

Complete Link	Single Link	Average	Ward
0.176	0.368	0.197	0.228

Table 6: Silhouette values for each linkage method

While a Silhouette value close to 1 is generally preferred, in this case we have to interpret the data with regards to our specific context.

The high value of Single Link Silhouette is probably due to the significant disproportion of the (only) two clusters obtained. The same reasoning may be applied to the Average method, which dendrogram appears to be inhomogeneous. Instead, even if the Complete link method yields the smallest Silhouette value, this is likely due to the little separation between clusters. The Ward method, on the other hand, produces well-separated clusters, as evidenced by the long vertical lines of the dendrogram and the good Silhouette value.

2.3.2 Clusters and Scatterplots

Now we propose some meaningful two-variables scatterplots in order to verify how the different linkage methods behave on clustering our data. For each pair of variables, we provide the resulting clusters for each of the four methods.

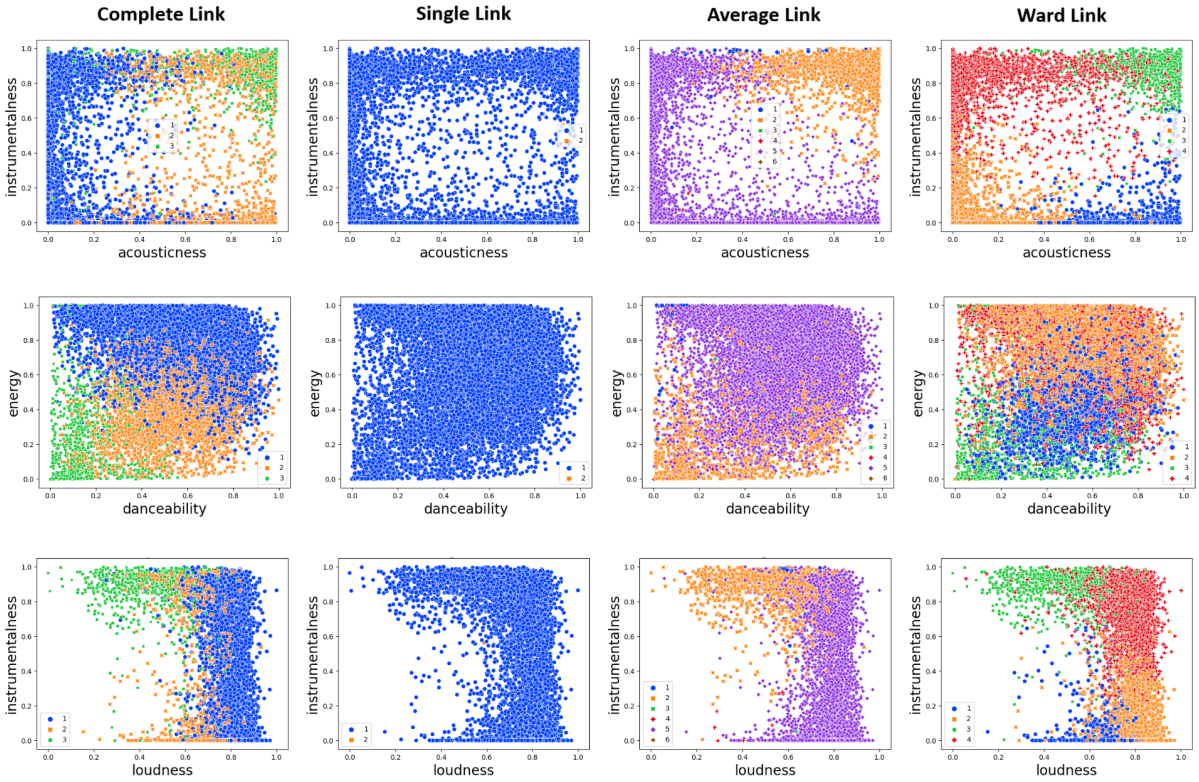


Figure 12: Scatterplots with highlighted clusters

This image provides valuable insights. Firstly, it is evident that the Single Link method (at least considering the 'cut' we have chosen) is unsuitable for cluster analysis because the vast majority of data points are categorized within a single cluster. Modifying the cut level of the dendrogram could potentially enhance clustering quality, but it is reasonable to assume that the Single Link method is just not well-suited for our dataset.

As anticipated from our examination of dendrograms, the Average method exhibits inadequate performance. Despite identifying six clusters, the majority of data points predominantly belong to just two clusters. In contrast, both the Complete Link and Ward Link methods demonstrate superior performance, as evidenced by the well-distinguishable clusters observed in the scatterplots.

2.4 Conclusions

During the application of the K-Means algorithm in centroid-based clustering analysis, we observed that conventional methods for determining the optimal number of clusters (k) did not yield significant results. This was attributed to the lack of a clear globular structure and the notable uniformity in terms of distribution and shape exhibited by the data. However, intriguing dynamics were noted within the data, particularly in the variable pairs `loudness`, `energy`, and `instrumentalness`, which seemed to have a predominant influence on the clustering process. When comparing Bisecting K-Means to traditional K-Means clustering, we observed variations in cluster structures resulting from inherent distinctions between the two algorithms. However, we do not find these differences particularly influential for our research.

Regarding density-based clustering, DBSCAN, using a low epsilon value yielded 3 interesting clusters, supporting certain observations from the data understanding phase. Specifically, the cluster labeled as C1 appears to group loud and danceable songs, while C2 contains songs typically considered "relaxing". Conversely, clustering with OPTICS did not provide particularly intriguing information.

With regards to hierarchical clustering, we realized a dendrogram for every distance function. The Ward method emerged as the top performer, closely followed by the Complete Link method. Conversely, the Single Link and the Average methods appear to be not well-suited for our dataset. In any case, even if it's not easy to measure the quality of a clustering algorithm, the Silhouette measure and the scatterplot visualizations helped us in reaching these conclusions.

3 Classification

This chapter focuses on implementing and evaluating three classifiers — Naive Bayes, K-Nearest Neighbours, and Decision Trees — with the aim of determining the most effective one for predicting `genre`, our chosen target feature. To optimize the performance of the classifiers as well as enhancing the models' ability to capture meaningful patterns in the data, input features were tailored differently for each algorithm. For Naive Bayes and K-Nearest Neighbors classifiers, all categorical features were excluded from our dataset, as these algorithms perform optimally with datasets comprising only continuous attributes. Conversely, for the Decision Tree classifier, specific categorical features (`explicit`, `key`, `mode`, and `time_signature`) were retained, considering the classifier's noteworthy ability in handling such features.

Optimal parameters for KNN and DT were determined via 10-fold cross-validation on the training set. This approach was intentionally chosen for its comprehensive use of available information for both training and testing, in contrast with the conventional practice of data partitioning.

For each classifier, we constructed and trained models on the training set and evaluated their performance on the test set. The chapter concludes with a comparative analysis of algorithm performance, offering definitive results obtained from the conclusive examination on the test set.

3.1 Naive Bayes

The simplicity of the Naive Bayes Classifier allows for a straightforward discussion of its outcomes on our dataset. In presenting our evaluation, we provide both key metrics and the ROC curve.

	Precision	**Recall**	**F1-score**
macro average	0.42	0.41	0.37
weighted average	0.42	0.40	0.37
accuracy		0.40	

Table 7: Evaluation measures on training set using Naive Bayes Classifier

The macro-average and weighted-average values for precision, recall, and F1-score collectively depict the model's overall ability to classify instances across different genres, indicating a moderate performance with room for improvement.

Yet, it is important to highlight that the model exhibits considerably better predictions for certain genres compared to others. In particular, the **F1-score** reaches its peak (0.76) for the `study` genre, followed by `black metal` (0.61) and `sleep` (0.58). These results suggest that these genres may be more distinguishable based on the specific set of attributes used in our analysis[3] since it provides a granular view of the model's performance across individual genres[4].

Simultaneously, the AUC-ROC of 0.88 underscores the model's robust discriminatory ability across the entire dataset.

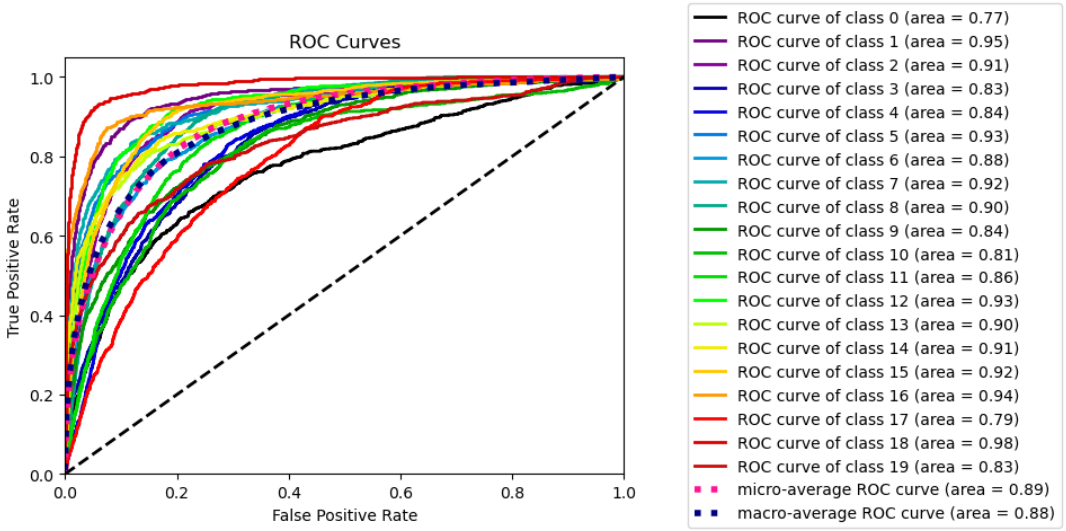


Figure 13: ROC curves of Naive Bayes Classifier

[3]For a deeper understanding of the model's predictions, we recommend consulting the published confusion matrix on GitHub accessible here.

[4]For simplicity genres were alphabetically mapped in a dictionary, assigned classes ranging from 0 to 19. E.g. *afrobeat*: class 0, *black-metal*: class 1...

Upon evaluating the model's performance on the balanced test set, consistency with outcomes from 10-fold cross-validation on the training set is evident. The results, summarized in the following Table 8 reveal a parallel pattern of performance metrics, reinforcing the model's reliability and generalizability.

	Precision	**Recall**	**F1-score**
macro average	0.42	0.41	0.37
accuracy		0.41	

Table 8: Evaluation measures on test set using Naive Bayes Classifier

3.2 K-Nearest Neighbors

To implement the K-Nearest Neighbors (KNN) classifier, we fine-tuned key hyperparameters for optimal performance. The hyperparameters in focus were k, representing the number of neighbors considered for prediction, and *weights*, determining the weighting strategy (*uniform* or *distance*).

Through rigorous experimentation using 10-fold cross-validation on the training set, we systematically explored the hyperparameter space. The range for k extended from 1 to 80, and for each k, we tested both *uniform* and *distance* settings for *weights*. A graph illustrating average accuracy and standard deviation across different k values guided our exploration.

The outcome of this process identified the optimal configuration as k=24 with *weights*=*distance*, achieving an impressive average **accuracy** of 0.4837. This finely-tuned configuration serves as the final setting for our KNN classifier and will be applied for the subsequent evaluation phase.

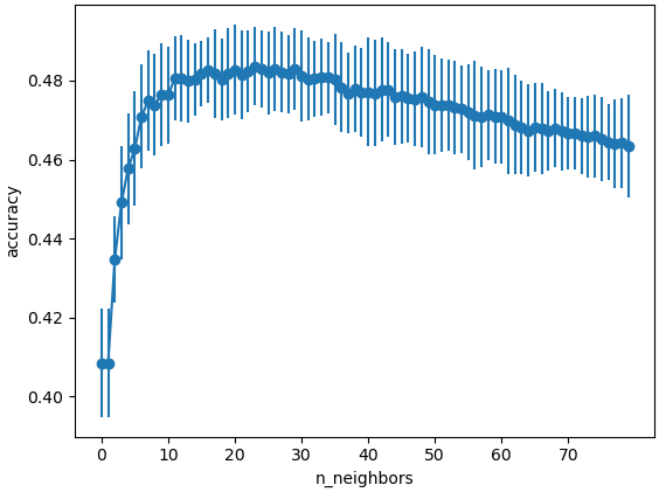


Figure 14: Accuracy of each classification with a different value of k (n_neighbors)

	Precision	**Recall**	**F1-score**
macro average	0.49	0.49	0.48
accuracy		0.48	

Table 9: Evaluation measures on training set using K-Nearest Neighbours

The classification report resulting from the 10-fold cross-validation on the training set, though not presented here in its entirety, provided valuable insights into the efficiency of the classifier, particularly in predicting specific genres. Notably, genres with the highest **F1-scores** include *sleep* (0.76), *study* (0.70), *black-metal* (0.68), *iranian* (0.63), and *forro* (0.61). It's noteworthy that the first three genres also exhibited high F1-scores with the Naive Bayes Classifier. This observation suggests that these genres are exceptionally well-characterized by the dataset compared to others, showcasing a consistent and robust representation across different classification algorithms.

Following this, we constructed a ROC curve graph, and upon examination, we notice that the genres with the maximum area under the curve (0.95) are indeed *black-metal* (class 1), *iranian* (class 12), *sleep* (class 16), and *study* (class 18). This further emphasizes the discriminative power of the classifier for these specific genres, as reflected in the ROC curve analysis.

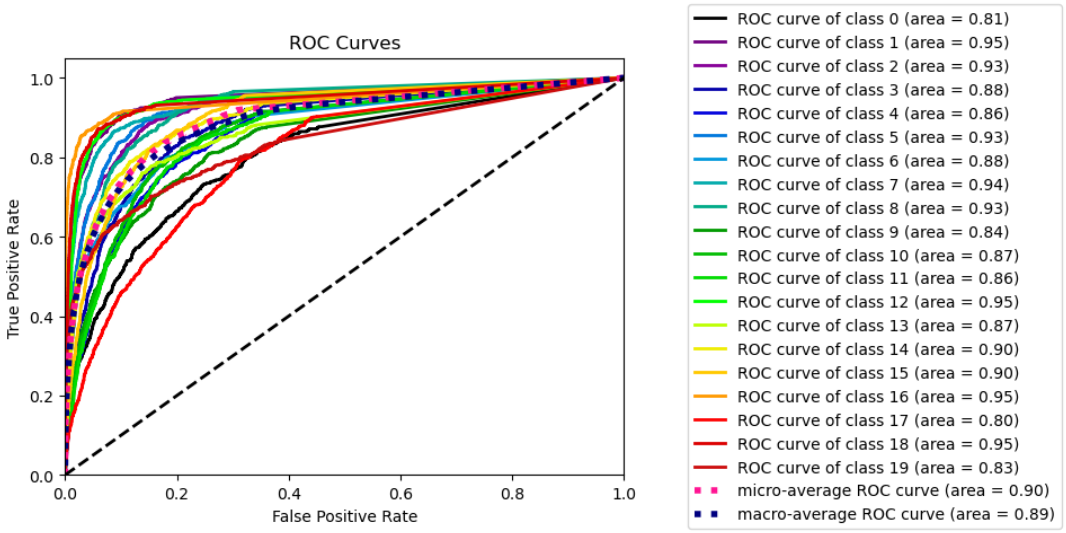


Figure 15: ROC curves of K-Nearest Neighbours

The results of the performance on the test set are reported here.

	Precision	**Recall**	**F1-score**
macro average	0.47	0.47	0.46
accuracy		0.47	

Table 10: Evaluation measures on test set using K-Nearest Neighbours

Although they show a slight decrease compared to the validation phase, the algorithm's effectiveness remains noteworthy, especially given the challenging task of classifying 20 different classes. Overall, the evaluation underscores that the K-Nearest Neighbors (KNN) classifier, in this dataset, outperforms the Naive Bayes Classifier, demonstrating superior accuracy (+0.06) and F1-score (+0.09).

3.3 Decision Trees

To optimize the classifier's predictive accuracy and generalization performance of this classifier, we executed a hyperparameter tuning process, focusing on *min_samples_split*, *min_samples_leaf*, *max_depth*, and *criterion* as key parameters. Through *Randomized Search Cross Validation*, we explored various parameter combinations. The *criterion* parameter was tested with *gini* and *entropy*; *max_depth* ranged from 2 to 60; *min_samples_split* had 9 values from 2 to 150 with incrementally increasing steps, and *min_samples_leaf* followed a similar pattern from 1 to 140.

As a result, the optimal configuration was identified as follows: *min_samples_split*=10, *min_samples_leaf*=5, *max_depth*=11, and *criterion*=*Gini*. Notably, the training set produced an accuracy of 0.48.

	Precision	**Recall**	**F1-score**
macro average	0.48	0.48	0.48
weighted average	0.48	0.48	0.48
accuracy		0.48	

Table 11: Evaluation measures on training set using Decision Trees Classifier

Reiterating our findings, the ROC curve plot serves as a visual confirmation that genres *black-metal* (class 1), *iranian* (class 12), *sleep* (class 16), and *study* (class 18) consistently exhibited the highest F1-scores, respectively 0.67, 0.62, 0.72, 0.75. This consistency is notably reflected in the ROC curve, further affirming the model's proficiency in accurately classifying these specific genres. The alignment between F1-scores and the ROC curve underscores the model's robustness and reliability in distinguishing and classifying these particular music genres with precision. This is further underscored by the AUC-ROC score of 0.83.

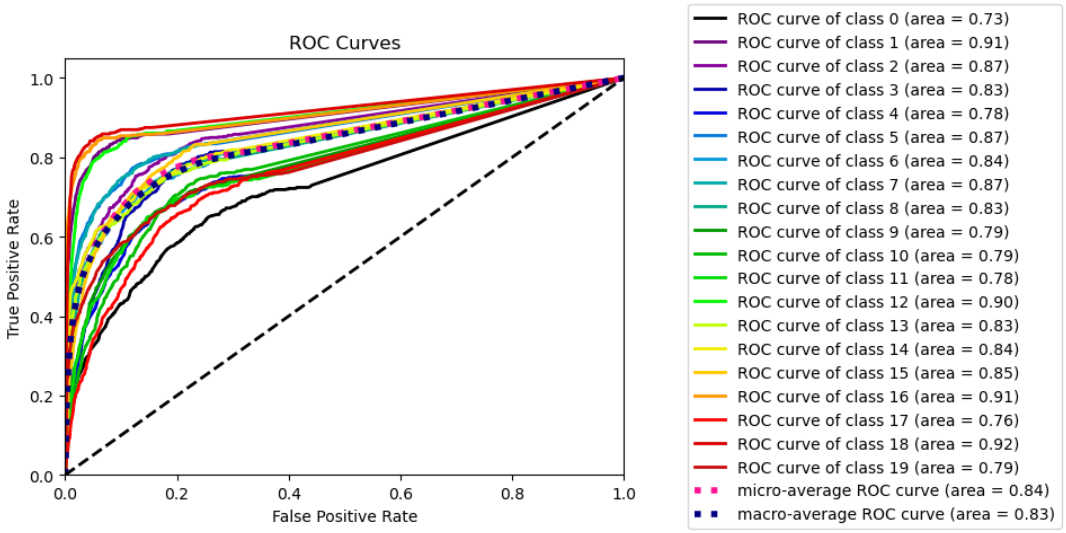


Figure 16: ROC curves of Decision Trees Classifier

However, it's essential to note that when applied to the test set, the accuracy slightly decreased to 0.43. This discrepancy between training and test performance prompts further exploration, suggesting the need for careful consideration of potential overfitting and the generalization capability of the model to unseen data.

	Precision	**Recall**	**F1-score**
macro average	0.45	0.43	0.44
accuracy		0.43	

Table 12: Evaluation measures on test set using Decision Trees Classifier

Finally, we analysed the feature importance in the classification task and found out that the most influential ones are *popularity* (0.20), *duration_ms* (0.13), *acousticness* (0.12), *danceability* (0.11).

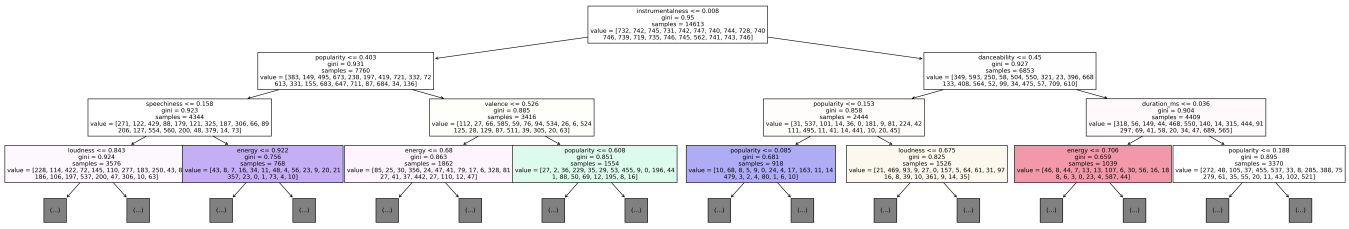


Figure 17: The head of the Decision Tree

3.4 Conclusions

All classifiers exhibit **accuracy** above 0.40 and within the 0.50 range, indicating overall satisfactory performance. However, a clear distinction emerges in their comparative effectiveness on the test set. We can therefore establish a ranking. In terms of overall performance, Naive Bayes performs the least favorably, achieving an accuracy of 0.41. Following this, the Decision Tree classifier attains an accuracy of 0.43, while KNN stands out as the top performer with an accuracy of 0.47.

The varying performance levels can be attributed to the inherent strengths and weaknesses of each algorithm. Naive Bayes, despite its simplicity and efficiency, may struggle to capture intricate relationships within the data. Decision Tree, while versatile, might face challenges in handling complex decision boundaries. On the other hand, KNN's reliance on proximity-based classification can be advantageous in capturing local patterns within the data, contributing to its superior performance in this specific context.

4 Regression

This chapter addresses a multiple regression task applied to the dataset, aiming to predict song `popularity` based on various numerical attributes. In the first section, we present the results of linear regression evaluation, considering scenarios without regularization, with Lasso regularization, and with Ridge regularization. The second section shifts focus to non-linear regression using the K-Nearest Neighbors and Decision Trees algorithms, previously employed in classification tasks. The performance of the regressors was assessed using three metrics: the coefficient of determination (R2), mean squared error, and mean absolute error.

4.1 Linear Regression

To evaluate linear regressors, the constant multiplying the regularization terms was consistently set to the default value of 1.

	R2	MSE	MAE
Linear	0.115	0.035	0.154
Ridge	0.115	0.035	0.154
Lasso	-0.008	0.040	0.168

Table 13: Evaluation measures on test set for linear regression

The outcomes of regression without regularization and with Ridge regularization are nearly identical (although the hyperplanes are not exactly the same), showcasing comparable predictive capabilities for popularity. Conversely, the results of regression with Lasso regularization exhibit lower performance, notably indicated by the negative R2, suggesting that the regressor's predictions are less accurate than those derived from a simple mean.

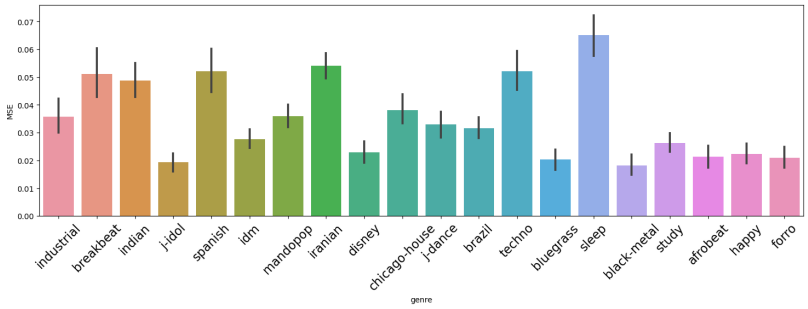


Figure 18: Distribution of the `popularity` prediction error with respect to the genres using linear regression

Something noteworthy emerges when considering genres: the efficiency of linear regression in predicting `popularity` varies across different musical genres. Specifically, while the predictions for the `sleep` genre were less effective, there was relatively higher accuracy for genres such as `black-metal`, `bluegrass`, and `forro`. This variability underscores the influence of genre-specific characteristics on the predictive performance of the linear regression model and suggests that predicting the popularity of songs in certain genres is definitely challenging.

4.2 Non-linear Regression

For non-linear regression, K-Nearest Neighbors and Decision Trees algorithms were employed. These algorithms require a validation phase to select the best model for predicting records in the test set. The techniques employed for hyperparameter tuning and validation mirror those utilized in the previous chapter for classification and will not be reiterated, except for differences in evaluation metrics.

The chosen hyperparameter combination for K-Nearest Neighbors and for the Decision Trees algorithm were respectively `(weights=distance, n_neighbors=64)` and `(min_samples_split=150, min_samples_leaf=5, max_depth=6, criterion=squared_error)` and the evaluation of the models based on their predictive power on the test set is displayed here.

	R2	MSE	MAE
K-Nearest Neighbors	0.179	0.032	0.145
Decision Trees	0.174	0.033	0.144

Table 14: Evaluation measures on test set for non-linear regression

From the results in Table 14, it can be observed that the performances of non-linear regression using K-Nearest Neighbors and Decision Trees were overall better compared to linear regression. The reason behind this result may be attributed to

the ability of non-linear regression methods to capture more complex relationships within the data, resulting in enhanced predictive capabilities for complex datasets like ours.

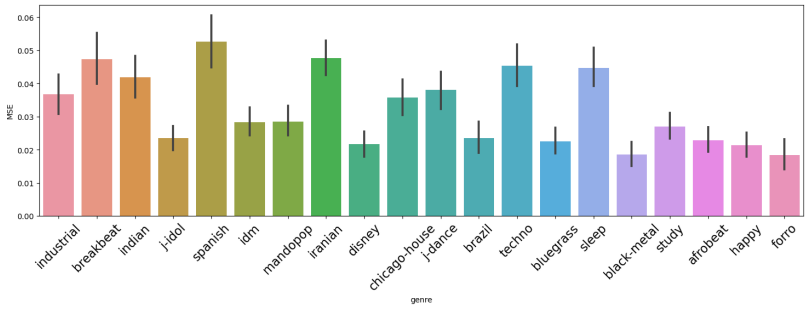


Figure 19: Distribution of the popularity prediction error with respect to the genres using the KNN algorithm for regression

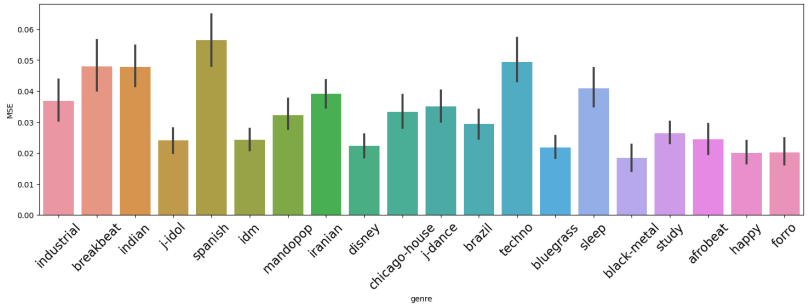


Figure 20: Distribution of the popularity prediction error with respect to the genres using Decision Trees algorithm for regression

If we focus on genres in plots 19 and 20, we notice a similar distribution of error with respect to genres for both the K-Nearest Neighbors and Decision Trees algorithms as well as more pronounced differences compared to the linear regression plot (Figure 18). This confirms once again the challenge of predicting the popularity of songs in certain genres (specifically: `sleep`, `iranian`, `spanish`, `techno`, and `breakbeat`).

Nonetheless, it is interesting to observe that in the case of non-linear regression, the genre `spanish` appears to be the most uncertain, but the error associated with it is still lower (below 0.06) than that of `sleep` (below 0.07) in the linear regression context.

In conclusion, the Decision Tree algorithm emerges as the most effective model for our regression task, also allowing us to identify the most relevant features for predicting `popularity`. Notably, `instrumentalness` takes the lead with a feature importance score of 0.51, followed by `loudness` with a score of 0.10.

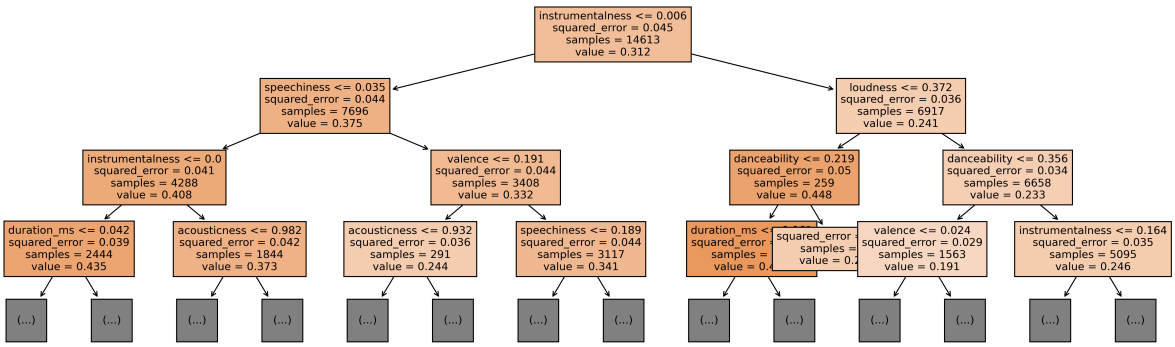


Figure 21: The head of the decision tree built for the popularity regression

5 Pattern Mining and Association Rule Mining

5.1 Preprocessing

To discern significant patterns within our dataset, it is imperative to transform it into a *transaction dataset*.

At the outset, we excluded intuitively irrelevant features for pattern identification from the analysis. Specifically, categorical features with an excessive number of unique values such as `name`, `artists`, `album_name`, as well as features where a single value consistently prevailed across all records, including `time_signature`, `mode` and `explicit`.

Subsequently, it became necessary to transform continuous values into categorical equivalents. This transformation was accomplished through a quantile-based discretization function, resulting in bins of uniform size. Each bin was then assigned a nominal description ranging from *veryLow* to *veryHigh*. Following this, a string containing the name of the respective feature was appended to each categorical value. The following tables illustrate the comparison between the original dataset and its modified version, onto which we applied our pattern mining algorithms.

key	**genre**	**duration_ms**	**popularity**	**danceability**	...
5	j-dance	0.053798	0.522727	0.687095	...
1	iranian	0.103137	0.000000	0.017048	...
2	black-metal	0.076181	0.034091	0.334268	...

key	**genre**	**duration_ms_bin**	**popularity_bin**	**danceability_bin**	...
5_key	j-dance	medium_dur	high_pop	high_dan	...
1_key	iranian	veryHigh_dur	veryLow_pop	veryLow_dan	...
2_key	black-metal	veryHigh_dur	veryLow_pop	veryLow_dan	...

Table 15: Transaction Dataset transformation

5.2 Pattern Extraction

To extract patterns from our transaction dataset, we employed the FP-Growth algorithm. Through experimentation with various support thresholds, we observed that compelling patterns emerged primarily with low support thresholds (10% or slightly above), as illustrated in the subsequent plot.

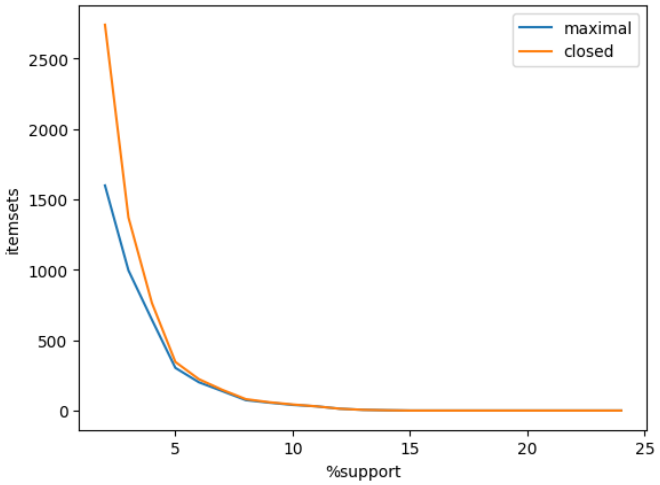


Figure 22: Number of maximal and closed itemset for different support thresholds

In the end, we adopted a 10% threshold, and the following patterns emerged:

- Upon examining the `energy` feature, we uncovered patterns such as `(veryLow_lou, veryLow_ene)` with a support of 13.90% and `(veryHigh_aco, veryLow_ene)` with a support of 13.75%. These findings suggest a correlation between a song's energy level and its loudness and acousticness. A more concise representation of this information is encapsulated in a pattern with a support of 10.42%: `(veryHigh_aco, veryLow_lou, veryLow_ene)`. The underlying intuition is that a song tends to exhibit higher energy levels when it is loud and lower energy levels when it leans towards being acoustic.

- In accordance with the pattern `(veryLow_val, veryLow_dan)` with a support of 11.41%, there appears to be an association between the positivity of a song and its danceability. Specifically, it suggests that if a song lacks positivity, it is likely not very danceable. Contrastingly, when examining the pattern, `(veryHigh_val, veryLow_ins)` with a support of 10.61% it becomes apparent that a song with a positive valence is inclined to be less instrumental.

- The pattern `(veryHigh_pop, veryLow_ins)` with a support of 11.43% suggests that popular songs are usually less instrumental.

- In relation to the `instrumentalness` feature three distinct patterns emerge when a song exhibits a very low level of instrumentalness: `(medium_aco, veryLow_ins)`, `(veryHigh_lou, veryLow_ins)`, `(low_dur, veryLow_ins)` respectively with supports of 11.09%, 11.07% and 10.62%.

Up to this point, we have focused on identifying frequent patterns within our data and formulating hypotheses about their significance. In the next section, building on the discovered patterns, we proceed on extracting effective rules that shed light on how the presence of a particular value could imply the presence of another.

5.3 Rules Extraction

In extracting rules, selecting a suitable confidence threshold is crucial. Thus, in order to make an informed decision, we constructed a plot depicting the relationship between confidence levels and the number of rules generated.

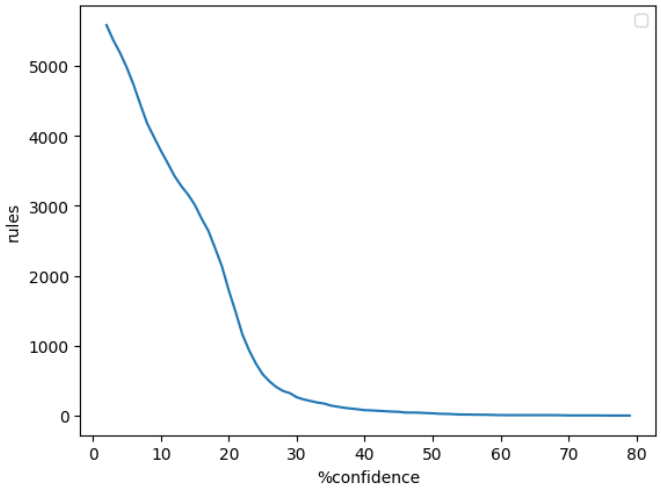


Figure 23: Number of rules for different confidence thresholds

After exploring various possibilities, we settled on a confidence threshold of 60%. As anticipated, the rules we uncovered are closely linked to the patterns we previously described.

- `(veryHigh_aco, veryLow_lou)` → `veryLow_ene`: This rule, characterized by a notable lift value of 4.43, is directly associated with the initial pattern identified in the previous section. It articulates that when there is a high level of acousticness coupled with a low level of loudness, the likelihood is that the song is not particularly energetic. Although additional rules involving various combinations of these three parameters exist, they exhibit lower lift levels. Consequently, we acknowledge their existence without delving into detailed analysis at this point.

- `(veryLow_val, veryLow_dan)` → `veryLow_lou`: With a lift value of 2.97, this rule indicates that when a song possesses a low valence and is not danceable, it is inclined to be not loud.

- `(veryHigh_pop)` → `veryLow_ins`: In this case, the rule with a lift of 1.51 is asymmetric. Hence, we can infer that a song with high popularity implies a low level of instrumentalness, but the reverse is not necessarily true.

We noticed that none of the previous rules incorporated information about music genres. This was somewhat disappointing as our objectives in both clustering and classification centered around identifying music genres or groups of genres. In pursuit of further investigation, we generated the following plot.

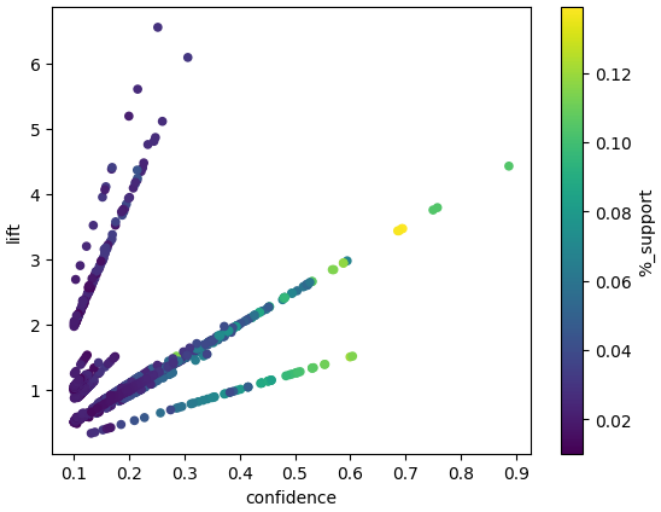


Figure 24: Lift value for different confidence levels

Figure 24 indicates that a confidence threshold of 60% leads to the exclusion of numerous rules characterized by high lift but very low confidence and support. Upon closer examination, it became evident that these rules are specifically associated with music genres. A selection of these genre-related rules is exemplified in Table 16.

`(veryLow_val, veryLow_dan)` → `sleep`	`(veryHigh_spe, veryLow_ins)` → `j-dance`
`(high_pop, veryLow_ins)` → `forro`	`(veryLow_spe, veryLow_ins)` → `mandopop`
`(veryLow_val, veryLow_dan)` → `iranian`	`(veryHigh_lou, veryLow_ins)` → `j-idol`

Table 16: Example of rules related to music genres.

Nevertheless, as previously mentioned, these rules exhibit a confidence level too low to be deemed meaningful. In the upcoming section, where we explore classification algorithms based on the indications from the rules, we are compelled to disregard them and concentrate on more robust and informative rules.

5.4 Exploiting Rules

In this section, we leverage the three rules extracted in the previous section to conduct classification tests. Specifically, we applied the Naive Bayes classifier, using the antecedent of each rule as input data and the consequent as the target variable.

The results of this analysis are summarized in Table 17.

Input Features	Target Feature	Accuracy	F1-score
`acousticness, loudness`	`energy`	50.48%	49%
`valence, danceability`	`loudness`	29.61%	26%
`popularity`	`instrumentalness`	43.50%	38%

Table 17: Rule-based classification results

The classification is conducted on the dataset outlined in Section 5.1, consisting solely of categorical data. The presence of categorical variables inherently facilitates the classification process. However, despite this advantage, the results presented in the preceding table do not attain exceptional levels; nevertheless, they still outperform those of a random classifier.

In conclusion, our dataset revealed a limited subset of intriguing patterns, resulting in a correspondingly smaller set of rules. Despite their modest number, the three rules we identified are intuitively comprehensible and can be applied for classification tasks, albeit without achieving remarkable performance levels.