UNIVERSITÀ DI PISA

Data Mining 1 2023/2024 project report

The Spotify Dataset

Professors:
Dino Pedreschi
Riccardo Guidotti

Erica Neri
Ilaria Ritelli
Virginia Marletta

December 2023

Contents

1 Data Understanding

The goal of the project is to analyze the Spotify Tracks Dataset, comprising of 15000 audio tracks accessible from the Spotify catalogue, which span through 20 different musical genres, such as techno, edm, black-metal, acoustic etc.

Each record is characterized by 24 attributes, distinguished as features which inherently identify the track, i.e., track name, artist, album name and its popularity score within the catalogue, and audio-derived features including danceability, energy, key and loudness, which provide insights on the perception they give to the listener.

1.1 Data Semantics

The Spotify Tracks Dataset is composed of 15000 records and 24 attributes, of which 9 are categorical and 15 are numeric variables. Tables 1 and 2 describe the attributes of the dataset.

duration_ms	The track length in milliseconds.
popularity	The popularity of a track is a value between 0 and 100, with 100 being the most popular.
danceability	Describes how suitable a track is for dancing. A value of 0.0 is least danceable and 1.0 is most danceable.
energy	A measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.
loudness	The overall loudness of a track in decibels (dB).
speechiness	Speechiness detects the presence of spoken words in a track.
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.
instrumentalness	Predicts whether a track contains no vocals. The closer the instrumentalness value is to 1.0
liveness	Detects the presence of an audience in the recording.
valence	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track.
tempo	The overall estimated tempo of a track in beats per minute (BPM).
features_duration_ms	The duration of the track in milliseconds
n_beats	The total number of time intervals of beats throughout the track.
n_bars	The total number of time intervals of the bars throughout the track.
popularity_confidence	The confidence, from 0.0 to 1.0, of the popularity of the song.

Table 1: Numeric attributes.

name	The artists' names who performed the track. If there is more than one artist, they are separated by a ;
explicit	Whether or not the track has explicit lyrics (true = yes it does; false = no it does not OR unknown).
artists	The artists' names who performed the track. If there is more than one artist, they are separated by a ;.
album_name	The album name in which the track appears.
key	The key the track is in. Integers map to pitches using standard Pitch Class notation.
mode	Mode indicates the modality (major or minor) of a track. Major is represented by 1 and minor is 0.
processing	[No description provided.]
time_signature	An estimated time signature.
genre	The genre in which the track belongs.

Table 2: Categorical attributes.

Among categorical attributes, *explicit* and *mode* are both binary attributes, the former with domain $\{True, False\}$, the latter with domain $\{0.0, 1.0\}$.

1.2 Distribution of the variables and statistics

In this section we analyze the features in depth using several Data Visualization techniques, such as scatter plots, histograms, and bar charts. We excluded the features *name*, *artists* and *album_name*, since computing their frequency was irrelevant to the analysis. Therefore we produced bar charts for features *explicit*, *key*, *mode*, *time_signature*, *processing* and *genre*.

There are exactly 750 tracks for each musical genre, with the vast majority being non-explicit (or not specified) on a 14/1 ratio (figure 1), that use for the most part the time signature 4.0 (figure 2).

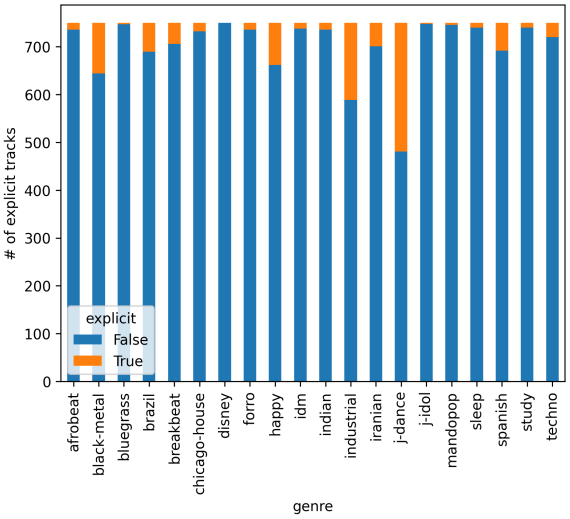

Figure 1: Distribution of explicit tracks among genres.

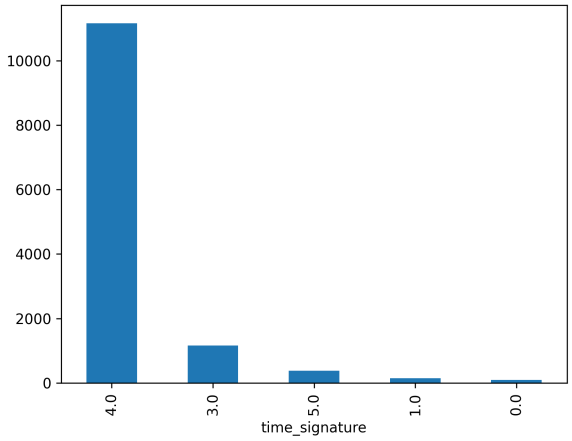

Figure 2: Frequency of keys.

By analyzing the mean value of *instrumentalness* for each value of *genre* (figure 3), we found that, coherently with the description of the first attribute, the genres with higher mean instrumentalness refer to electronic music and background noise, which will likely contain very few vocals; the genres with lower values of instrumentalness are instead pop or folk genres that heavily rely on sung lyrics. A similar analysis by considering *acousticness* (figure 4) yielded similar results: genres with lower mean acousticness are electronic ones (since acoustic music can be considered as the opposite of electronic music).

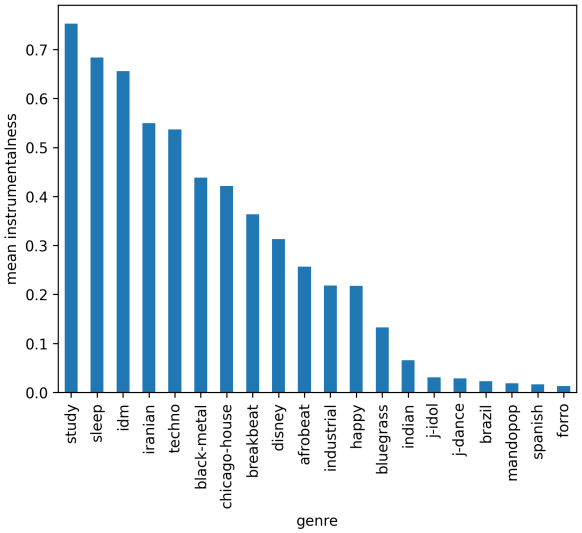

Figure 3: Mean instrumentalness for each musical genre.

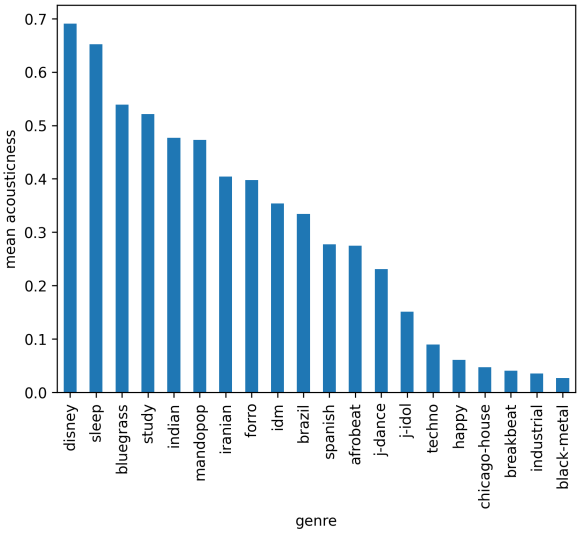

Figure 4: Mean acousticness for each musical genre.

Furthermore, we noticed how for most of the records, the value of *time_signature* is related to the value of *n_beats* and *n_bars*, as it's approximately equal to the ratio between the two (figure 5). Most tracks play on the major mode (1.0), with the keys 7, 0, and 1 being the most used (figure 6).

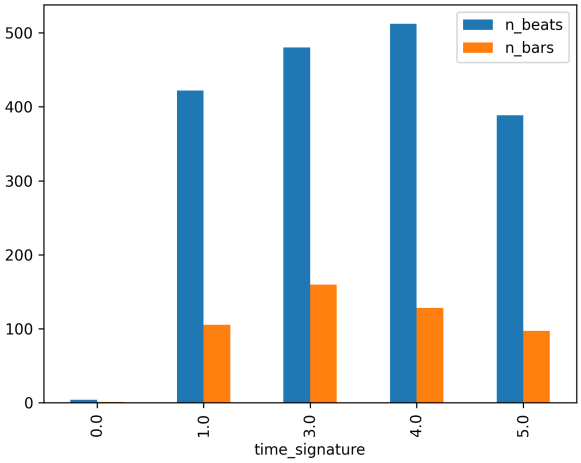

Figure 5: Relationship between *n_beats*, *n_bars*, and *time_signature*.

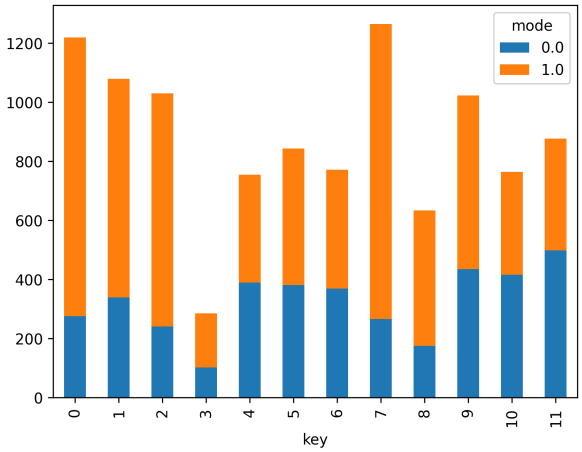

Figure 6: Relationship between node and key.

As for the remaining 15 continuous features, we computed their respective histograms, from which we inferred that for the most part the distribution they follow is either left or right skewed, spare for *popularity_confidence*.

We analyzed the distribution of variables with scatter plots as well. Some interesting findings are reported in the two figures below (figure 7 and 8). The scatter plots show that the tracks with the lowest values of *loudness* also have low danceability, energy, speechiness, and valence. As the loudness increases, tracks tend to have more varying values of danceability and speechiness. It is also evident that the variables *loudness* and *energy* are highly positively correlated; there's also some degree of positive correlation between *loudness* and *valence* (observations confirmed by looking at the heatmap for the dataset).

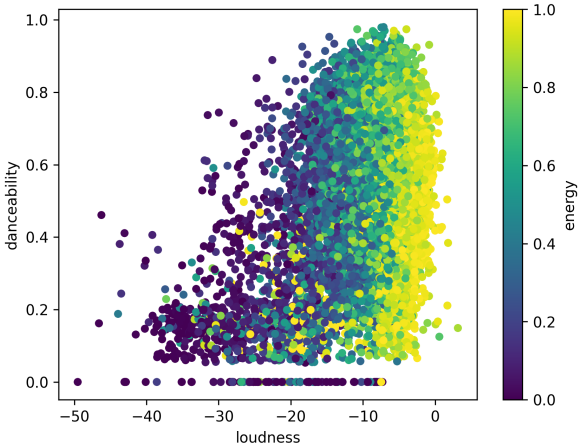

Figure 7: Scatter plot showing the relationship between loudness, danceability, and energy.

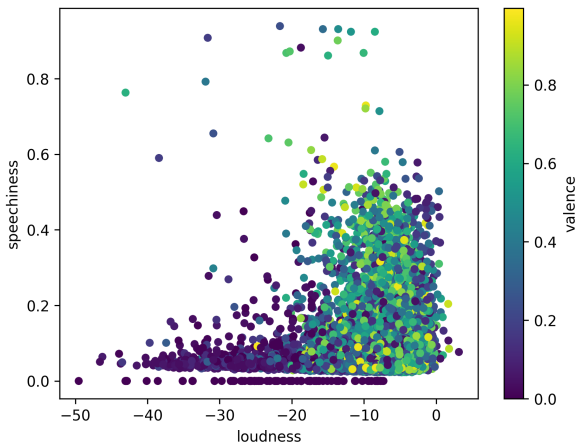

Figure 8: Scatter plot showing the relationship between loudness, speechiness, and valence.

1.3 Handling Missing Values

The task of identifying any missing values was performed by computing a bar chart counting the number of null values for all the features (figure 9). Observing the bar chart highlighted the presence of missing values for the features *mode*, *time_signature* and *popularity_confidence*.

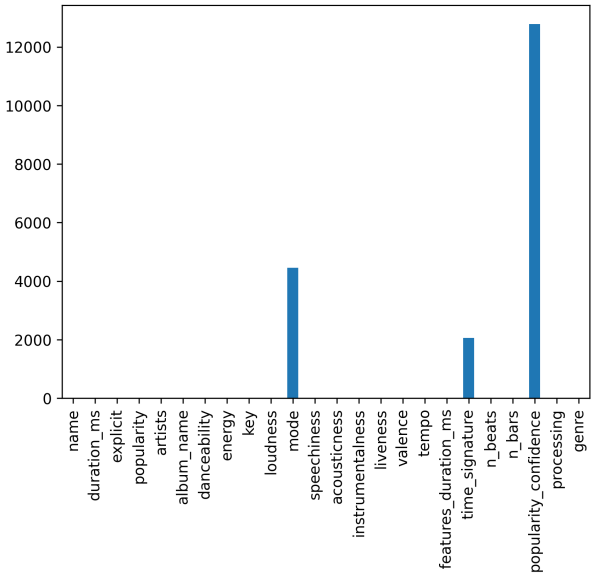

Figure 9: Missing values found.

mode	4450
time_signature	2062
popularity_confidence	12783

Figure 10: Amount of missing values for the variables *mode*, *time_signature* and *popularity_confidence*.

We chose to completely remove the attribute *popularity_confidence*, since its values were mostly missing with no apparent way to recover them from other attribute values.

We previously observed that the attribute *time_signature* could be extrapolated by dividing the variable *n_beats* by *n_bars*, so for each row containing a missing value, we calculated the result, which was then approximated to the nearest integer and used to substitute the *NaN* value.

To fill the attribute *mode*, we observed that the per the provided description of the feature, it indicates whether a track is performed on a major or a minor mode. Since certain scale-mode pairs tend to be more popular than others, and the scale of the song is indicated by the attribute *key*, we filled the missing values by first calculating the most popular mode for each key (see figure 6), and then filling each row by choosing the corresponding value based on the *key* attribute of that row (e.g., for each row with key 7, the missing value was filled in as 1.0, while for rows with key 11, the missing value was filled in with 0.0).

1.4 Pairwise Correlations and Eventual Elimination of Variables

We observed pairwise correlation between variables by computing their correlation heatmap (figure 11), displaying Pearson's correlation coefficient for each pair of features. We decided to drop the features *n_beats*, *n_bars*, and *features_duration_ms* as they are strongly correlated with *duration_ms*, with a coefficient greater than 0.80. The pairs *loudness-energy* (already observed before) and *acousticness-energy* show, respectively, a strong positive correlation and a strong negative correlation, with a significant coefficient (0.72 and -0.70). We decided to drop both *loudness* and *acousticness* from the dataset.

We also removed the feature *processing*, after producing a cross tabulation between *processing* and *key*, and noticing that each value of the former could be matched to a specific value of the latter in every single record of the dataset.

Overall, the number of features was reduced from 24 to 16.

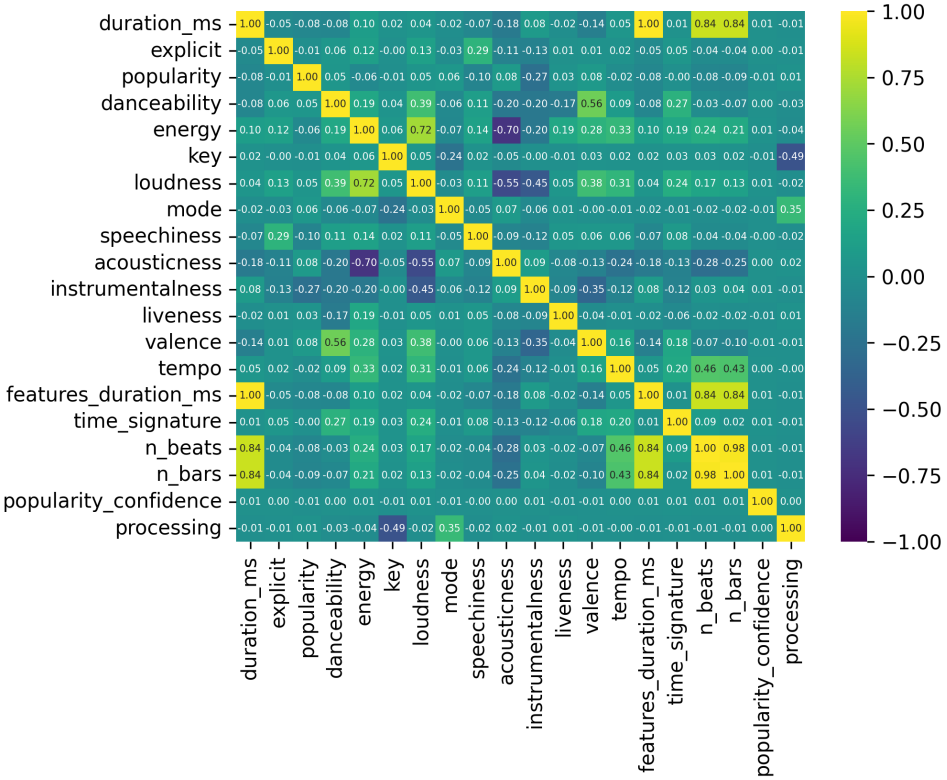

Figure 11: Correlation matrix for the dataset before removing any feature.

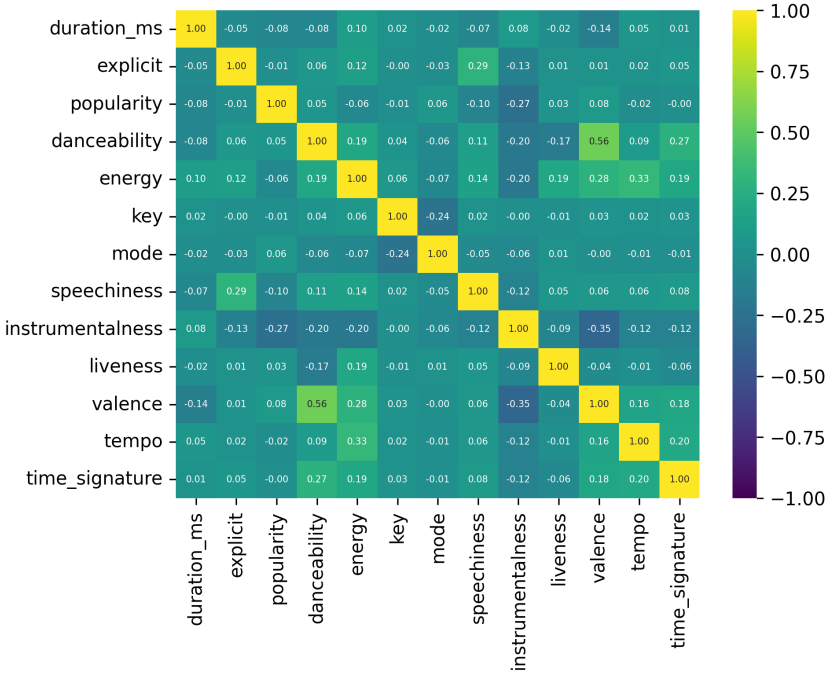

Figure 12: Correlation matrix for the dataset after removing highly correlated features, as well as *popularity_confidence*, and *processing*.

1.5 Handling Outliers

To identify whether there were outliers, we produced the boxplots for the 15 continuous variables. We detected outliers for the features *duration_ms* and *speechiness* (Figure 13).

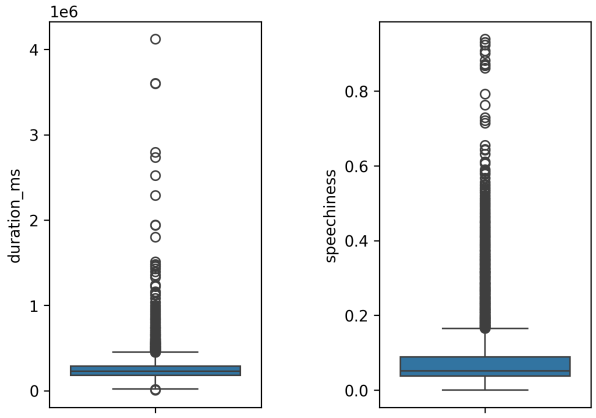

Figure 13: Outliers for features duration_ms, speechiness, and loudness.

In order to identify outliers, we considered all points whose value is greater or equal than the 2^{nd} quartile plus 10 times the inter-quartile range for both variables. We detected 31 outliers for *speechiness* and 19 for *duration_ms*, which were removed from the dataset to prevent them from influencing further analysis. The final number of records has been lowered to 14950.

2 Clustering

The dataset was analyzed using three classes of clustering algorithms: centroid-based, density-based, and hierarchical. For this task, we considered all of the numeric attributes, which were normalized using rescaling (min-max normalization).

2.1 Analysis by centroid-based methods

2.1.1 K-means

The first step we took was setting the value of the hyperparameter k. We calculated both the Sum of Squared Error (SSE) and Silhouette Score for k values in the range [2,50], and plotted their values (figure 14).

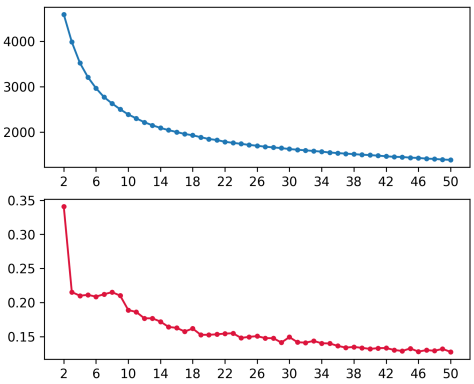

Figure 14: SSE and Silhouette Score for k-means.

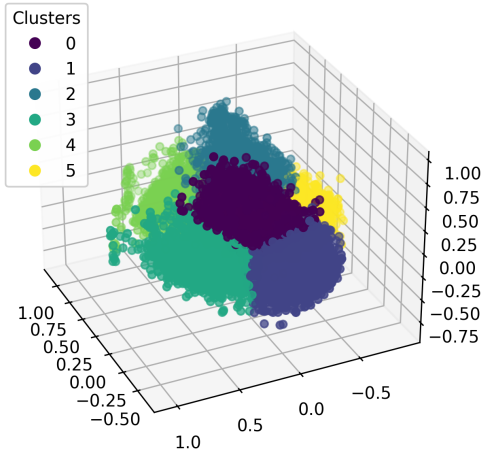

Figure 15: Clusters produced by 6-means, visualized via PCA.

Based on the trend of the two curves, we have selected the parameter k = 6 as the best trade-off between SSE and Silhouette Score, and executed K-means accordingly. We chose not to use an higher value of k as it would produce a less meaningful clustering with little improvement over SSE and Silhouette Score. The 6 clusters produced are visualized in figure 15. They are composed respectively of 2676, 4438, 1602, 2923, 1484 and 1827 points, and notably Cluster 1 is the largest one. The SSE value for this clustering is 2967.046, and the Silhouette Score obtained is 0.209.

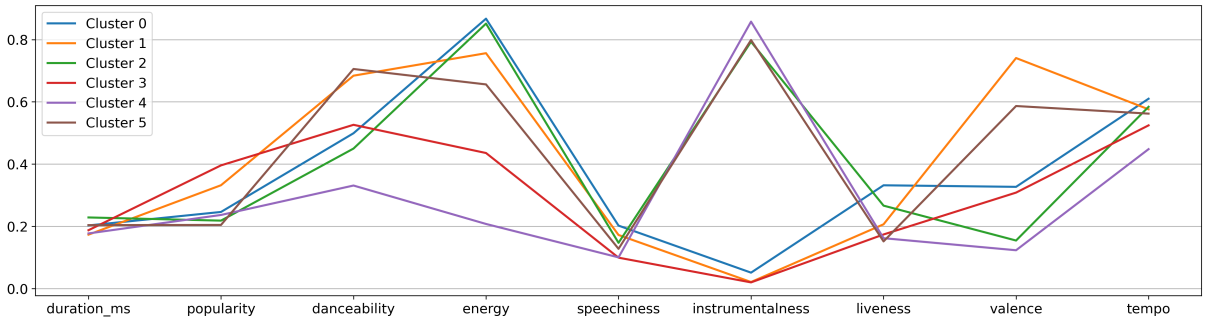

Figure 16: Parallel coordinates plot for the centroids found by 6-means.

By looking at figure 16, we can notice how the clusters tend to separate points mostly based on danceability, energy, instrumentalness, and valence.

2.1.2 Bisecting K-means

The second centroid-based algorithm tested was Bisecting K-means. We again calculated both the SSE and Silhouette Score for k values in the range [2,50]. The results are shown in figure 17. For this algorithm we chose k = 5, through further observation of the coefficients and careful measure of the trade off between SSE and Silhouette Coefficient: a value of k = 6 would reduce the SSE, but also significantly lower the Silhouette Score as well.

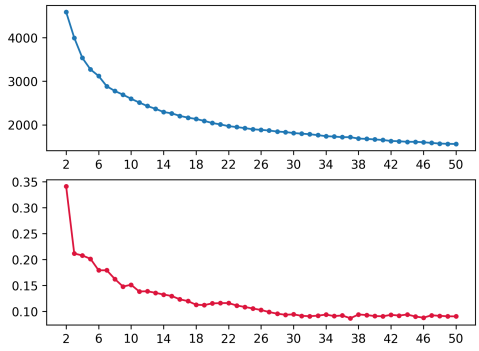

Figure 17: SSE and Silhouette Score for Bisecting k-means.

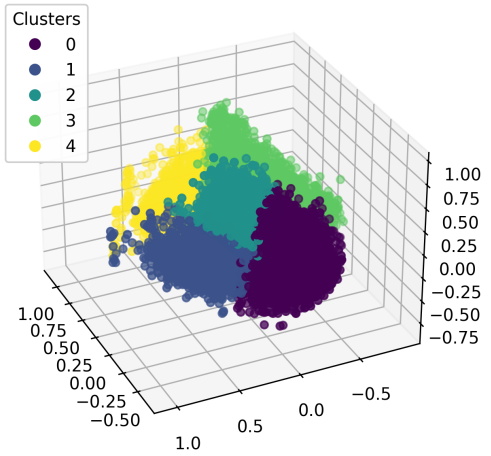

Figure 18: Clusters produced by Bisecting 5-means, visualized via PCA.

The clusters found for k = 5 were adequately balanced, spare for Cluster 0 which was the largest and contained 5388 points, while the others accounted for 2728, 1958, 3246 and 1630 points. The Silhouette Score and SSE for this clustering are 0.2011 and 3274.719, respectively. Figure 19 shows the parallel coordinates of the centroids of the clusters. The division of the points across clusters is similar to the one found in the clustering produced via 6-means.

Since Cluster 3 found by Bisecting 5-means is approximately the fusion between Clusters 2 and 5 in 6-means, this is reflected in the new coordinates of the centroids of the cluster: for

each feature, they are approximately the mean between the coordinates of the centroids found for Clusters 2 and 5 in 6-means.

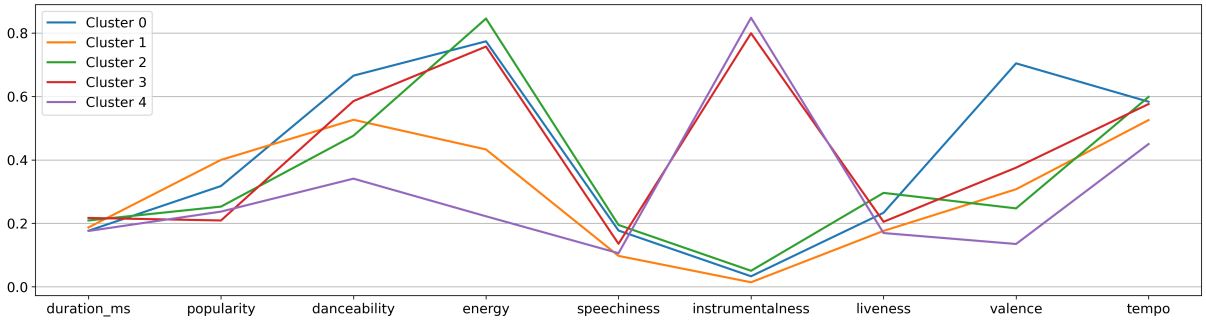

Figure 19: Parallel coordinates plot for the centroids found by Bisecting 5-means.

2.1.3 X-means

X-means is the last centroid-based clustering method we used to analyze the dataset. We started testing the algorithm without setting any limit to the execution; however, the output was a clustering containing 20 different clusters, which proved difficult to analyze. We repeated the execution, this time by imposing a limit on the number of final clusters equal to 6, which produced the clustering in figure 20.

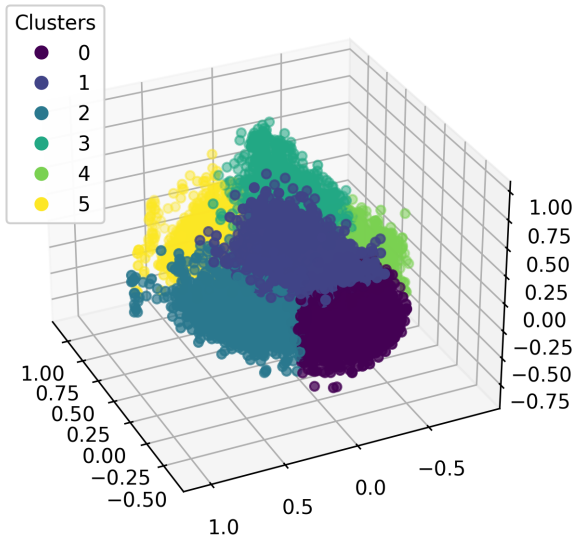

Figure 20: Clusters produced by X means (limiting the maximum number of clusters to 6), visualized via PCA

The results of the procedure were overall similar to those obtained by using 6-means, as the 6 clusters obtained were, yet again, a larger cluster (Cluster 0) comprising of 4514 points, and the remaining 5 of respectively 2520, 2981, 1762, 1553 and 1620 points. The Silhouette Score resulted in 0.21.

2.2 Analysis by density-based clustering

2.2.1 DBSCAN

For this algorithm we had to set two hyperparameters, *Eps*, which is the radius to consider for each point, and *MinPts*, which is the minimum number of points that must be within *Eps* distance to a point p in order to consider it a core point of a cluster. Using the K-NN algorithm,

we plotted, for each value of k equal to powers of 2 in the range $[2, 512]$, the sorted distances from the k^{th} point, and then checked the presence of an elbow in the curve that could suggest an ideal value for the hyperparameter *Eps*.

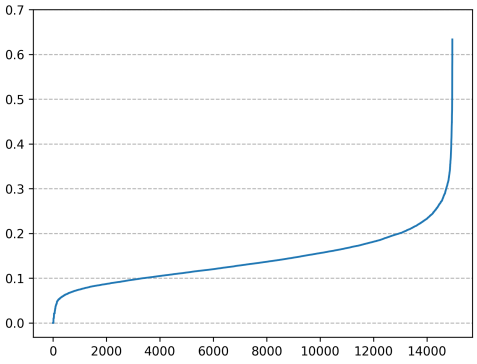

Figure 21: Sorted distances from k^{th} neighbor (here, k = 64).

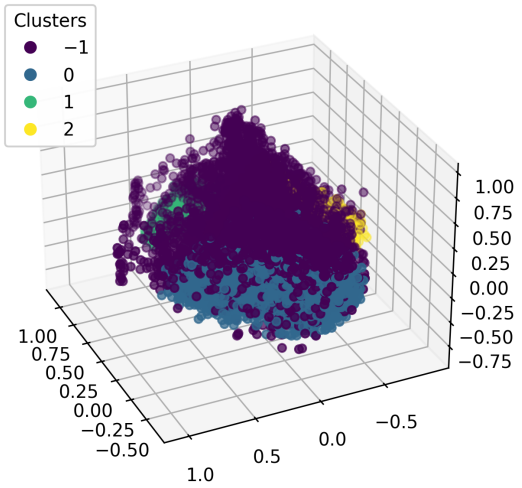

Figure 22: Clusters produced by DBSCAN, visualized via PCA.

From figure 21 it can be seen that the elbow of the curve is between 0.2 and 0.3. We set the value of *Eps* to 0.24, and tested different values of *MinPts* by setting it to powers of 2 in the range $[2, 512]$. We then chose 64 as the final value of the hyperparameter since it produced a sufficiently high Silhouette Score. By using these hyperparameters, DBSCAN produced three clusters (figure 22) of size 7071, 605, and 1428, respectively. 5846 points (a little over than $1/3^{rd}$ of the total) were classified as noise points. The Silhouette Score of the clustering is 0.42.

2.2.2 OPTICS

The analysis was done by using the same values of the hyperparameters used for DBSCAN, since they yielded similar values of Silhouette Score. The result produced by the algorithm was similar to that of DBSCAN, with again three clusters (Figures 24 and 23), of size 1411, 587, and 7067, respectively, and 5885 points classified as noise points. The Silhouette Score amounts, again, to 0.42. In the reachability plot, the two smaller valleys (on the far left) correspond to the first two clusters, while the wider valley on the right corresponds to the bigger cluster.

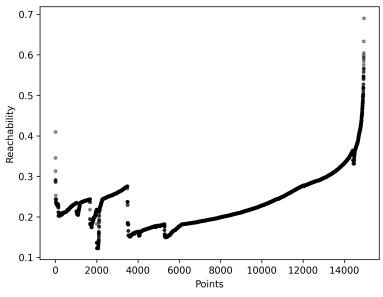

Figure 23: Reachability plot produced by OPTICS.

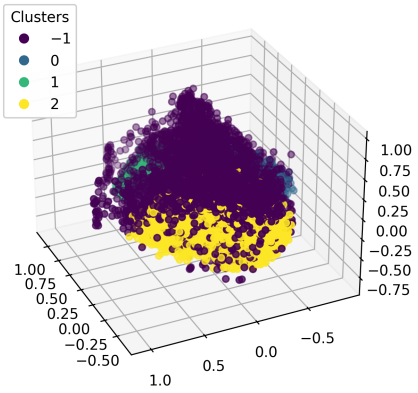

Figure 24: Clusters produced by OPTICS, visualized via PCA.

2.3 Analysis by hierarchical clustering

Hierarchical (agglomerative) clustering was repeated on the dataset using different measures, namely complete link (MAX), single link (MIN), group average, and Ward's method. The dendrograms obtained from the executions of the algorithm are shown, respectively, in figures 25, 26, 27, and 28.

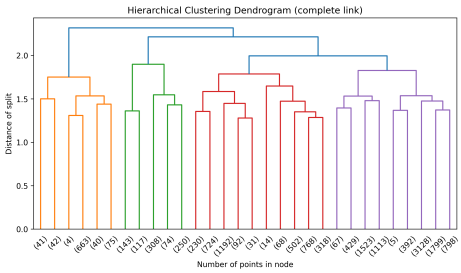

Figure 25: Dendrogram with complete link.

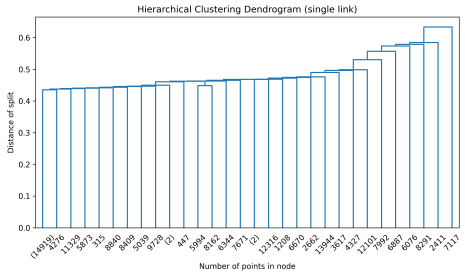

Figure 26: Dendrogram with single link.

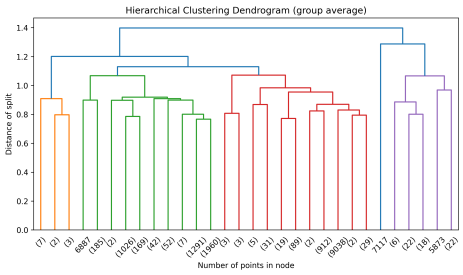

Figure 27: Dendrogram with group average.

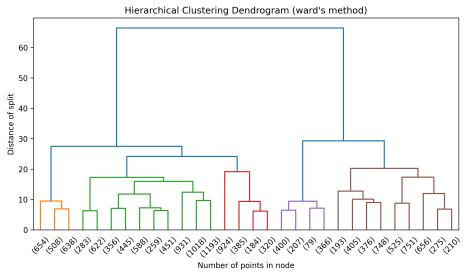

Figure 28: Dendrogram with Ward's method.

Single link produced the worst result overall as it isolates several single points into singleton clusters at any cutting height. Group average performed slightly better, but still produced a very unbalanced clustering, with clusters containing few points and a couple clusters containing most of the remaining ones.

Complete link (figure 29) produced better but still unbalanced clusters, with a bigger cluster containing almost two thirds of the data (Cluster 4), a medium-sized cluster (Cluster 3), and two very small clusters of less than 1000 points each (Clusters 1 and 2). Lastly, Ward's Method produced the most well balanced clustering (figure 30), although its Silhouette Score is the lowest recorded among the other attempts (0.17 against 0.30 for group average and 0.27 for complete link).

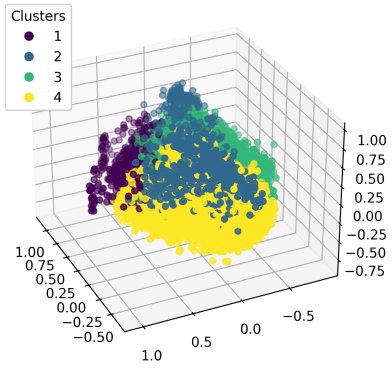

Figure 29: Clusters produced by hierarchical clustering using complete link, visualized via PCA.

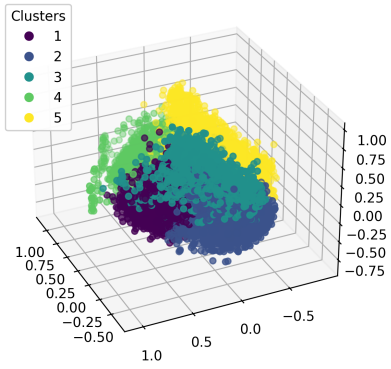

Figure 30: Clusters produced by hierarchical clustering using Ward's method, visualized via PCA.

2.4 Final discussion

We analyzed the dataset with several clustering techniques, obtaining different results. Overall, we could observe that:

- Centroid-based algorithms produced clusterings with clusters of different sizes. They managed to separate points on the basis of four main features by characterizing them as having high/medium/low values of them.
- Density based algorithms, although being characterized by the highest Silhouette Score of all techniques employed, do not separate points as well as centroid-based ones; the output only seems to capture differences among points in values of *energy* and *instrumentalness*.
- Hierarchical clusterings done using single link and group average measures are very unbalanced; specifically, they isolate several single points as clusters. Complete link also produces a better but still slightly unbalanced result; finally, Ward's method produced the most balanced clustering, but also had the lowest Silhouette Score.

In conclusion, the best clustering was obtained by using centroid-based algorithms, while the second best clustering was the one obtained by using hierarchical clustering with Ward's method.

3 Classification

In this section, we examined various classification methods, including K-Nearest Neighbors, Decision Tree, and Naive Bayes classifier. To train and assess the performance of the models, we used the two provided datasets *train* (which is the one used in the past sections) and *test*, respectively. The latter dataset was also preprocessed with the same procedures described in section 1. We chose the attribute *genre* as our target variable: we first considered a multiclass problem, where each genre corresponds to a different class, and later a binary problem, in which we isolated one of the values (*"techno"*) as the positive class against all other values grouped as the negative class. The end of the section will compare the different models and discuss which one is the best among them.

3.1 K-Nearest Neighbors

We began our investigation by using the K-Nearest Neighbor (K-NN) model for multiclass classification. First, the input was normalized via standardization to prepare the data for the model. Afterwards, we performed a grid search to find the optimal values for the number of neighbors k, varying between 10 and 100, whether to use distances as weights during prediction or not, and the choice of distance function between euclidean or Manhattan (*cityblock*). The grid search evaluated the quality of each hyperparameters combination by performing a 5-Fold Cross Validation with shuffling. The optimal hyperparameters found are:

- $k = 18$
- $metric = cityblock$
- $weight = distance$

The accuracy evaluated on the test set is equal to 0.52. Figure 31 shows the confusion matrix of the model. The three classes with the highest precision and recall scores are *study*, *sleep*, and *forro*. The three ones with the lowest values are *spanish*, *afrobeat*, and *industrial*.

3.1.1 K-NN for Binary Classification Task

We repeated the previous analysis, but considering two class labels, *techno* and *not techno*. We again performed a grid search with 5-Fold Cross Validation to identify the best hyperparameters, which are

- $k = 22$
- *metric*= *cityblock*
- *weight* = *distance*

The accuracy of the model for these hyperparameters is 0.96. Table 3 summarizes precision, recall, and F1 score for the two classes. The positive class has lower scores than the negative class; this is affected by the fact that the training dataset is highly unbalanced, with only 749 records belonging to the class *techno*, and 14201 belonging to the class *not techno*. Specifically, the positive class shows an high precision and a low recall, meaning that the model does not detect the class too well and produces a lot of false negatives, but the times it does identify a member of the positive class, it is reasonably reliable.

Class	**Precision**	**Recall**	**F1-Score**	**AUROC**
techno	0.83	0.36	0.50	0.87
not techno	0.97	1.00	0.98	0.87

Table 3: Summary of the main classification measures for the binary problem for K-NN.

3.2 Decision Tree

The analysis with the Decision Tree classifier started with a grid search to determine the optimal values for the following hyperparameters:

- *max_depth*, the maximum limit for the depth of the tree, varying between 1 and 100, plus the default value *None* (as in no maximum limit);
- *min_samples_split*, the minimum number of samples required to split a node, varying between 2 and 512 in power of 2 increments;
- *min_samples_leaf*, the minimum number of samples required to be at a leaf node, varying between 2 and 512 in power of 2 increments;
- *criterion*, the function used to measure the quality of the split, chosen between *gini* and *entropy*.

The grid search again used a 5-Fold Cross Validation to estimate the performance of the model. The best hyperparameters found are the following:

- *max_depth*: 11
- *min_samples_leaf*: 8
- *min_samples_split*: 32
- *criterion*: *gini*

The accuracy of the model is 0.46, so worse than the K-NN model. From the confusion matrix, shown in Figure 32, we can see how performance degrades for most classes, with only some minor improvements for a handful classes. Figure 33 shows the decision tree found with the optimal hyperparameters. By looking at the splits done in the first few levels, it can be observed how the first split divides the records into two subsets based on *danceability*, with $\approx$ 13000 records

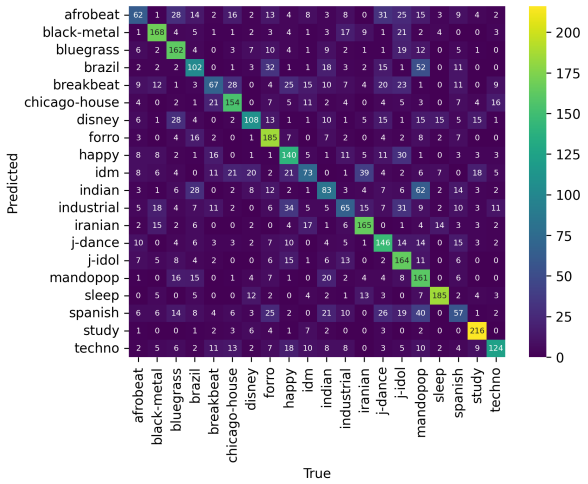

Figure 31: Confusion matrix of K-NN.

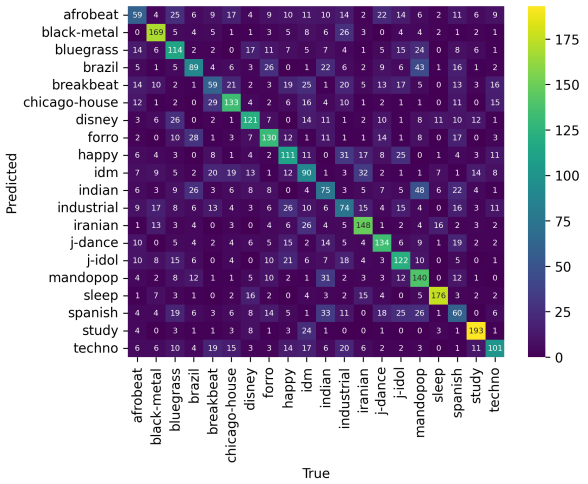

Figure 32: Confusion matrix of Decision Tree.

in the right child, and ≈ 2000 records in the left child. The first splits done in the left subtree immediately produce relatively pure nodes, containing many of the classes with the highest precision and recall (e.g., *sleep*, *iranian*, *black-metal*). The subtree on the right, instead, shows less pure nodes, with the exception of the ones that contain tracks belonging to the *study* genre, which is also the one that corresponds to the highest precision/recall scores. In general, all classes that correspond to better scores are immediately isolated into leaves at low depth in the tree, while instances that belong to other classes can be found into leaves at higher depths. This tree also suggests interesting ways in which different genres relate to each other: *iranian* and *black-metal* differ in terms of *popularity*, *j-idol* and *j-dance* are understandably close to each other with different levels of *speechiness*, and the *study* genre is close to instrumental ones such as *techno*, *industrial*, and *chicago-house*.

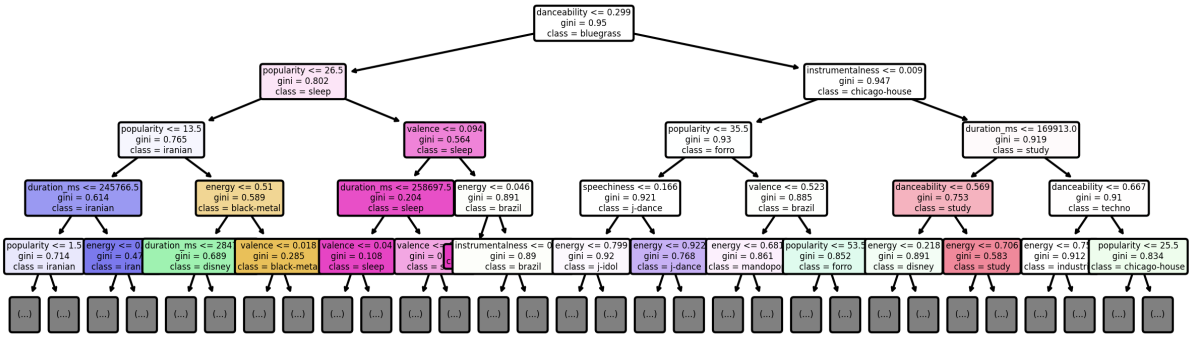

Figure 33: Decision tree constructed for the multiclass problem.

3.2.1 Decision Tree for Binary Classification Task

The search for the best hyperparameters was performed as above, with the following values found as the best:

- *max_depth*: 8
- *min_samples_leaf*: 16
- *min_samples_split*: 32
- *criterion*: *gini*

The accuracy of the model is 0.96, so the same as the one obtained with K-Nearest Neighbors. Precision, recall and F1-score are also similar (slightly lower) than the ones obtained with K-Nearest Neighbors.

Class	Precision	Recall	F1-Score	AUROC
techno	0.74	0.33	0.45	0.82
not techno	0.97	0.99	0.98	0.82

Table 4: Summary of the main classification measures for the binary problem for the Decision Tree model.

Figure 34 shows the decision tree constructed for the multiclass task. The tree exhibits a certain level of complexity, as it doesn't immediately partition instances into the two classes. Many nodes, characterised by a predominant presence of the *non-techno* class, remain enough to be classified as leaves. One subtree on the right side of the tree appears to include mainly instances of the positive class. This particular subtree is discerned by specific feature conditions, namely high *instrumentalness*, high *popularity*, high *duration_ms*, and high *danceability*.

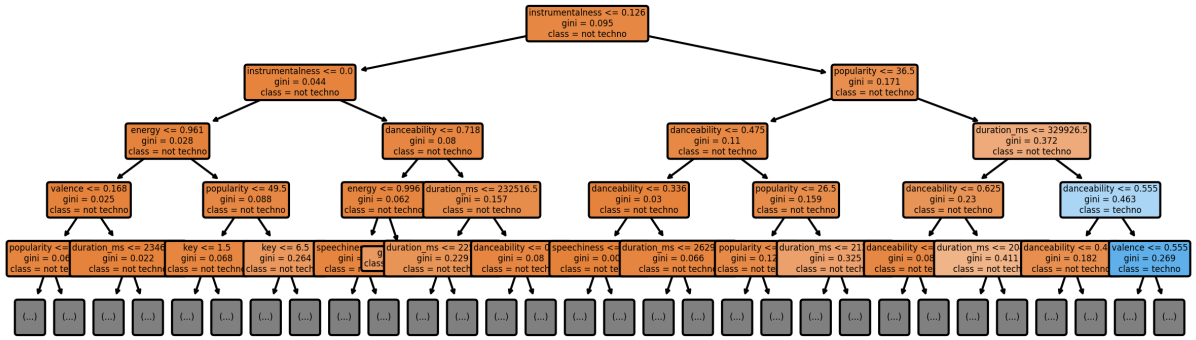

Figure 34: Decision tree constructed for the binary problem.

3.3 Naive Bayes

The third model used to analyze the dataset is the Naive Bayes classifier. The analysis initially used a Gaussian Naive Bayes model, which assumes that the continuous data is drawn from a Gaussian distribution, but due to the emergence of null conditional probabilities, we subsequently utilized the Multinomial classifier in order to better estimate our values. We then proceeded to compare the outcomes obtained from both approaches. Figures 35 and 36 show the confusion matrices for the two variants of the model.

The results obtained are significantly less accurate than the previous models. This discrepancy is particularly noticeable for classes with minimal or zero precision and recall, leading to challenges in accurate predictions. The results achieved using Gaussian Naive Bayes are better than the other, possibly because our data exhibits characteristics more aligned with a continuous distribution rather than a discrete one. The Gaussian distribution, being more flexible in this context, yields superior accuracy (0.25 against 0.19 obtained with Multinomial Naive Bayes).

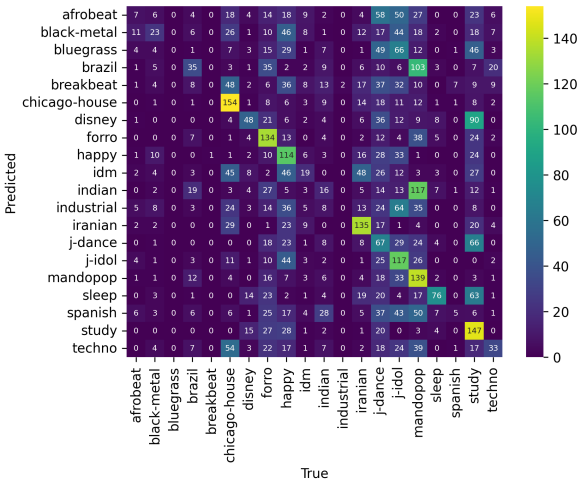

Figure 35: Confusion matrix of Gaussian Naive Bayes.

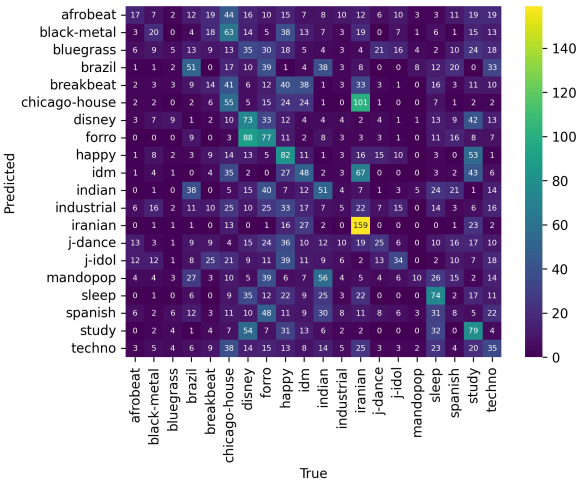

Figure 36: Confusion matrix of Multinomial Naive Bayes.

3.3.1 Naive Bayes for Binary Classification Task

For the binary classification task, both Naive Bayesian variants of the model report low values of both precision, recall, and F1-score for the positive class, as demonstrated in Tables 37a and 37b. Their accuracy is 0.95 and 0.73 for the Gaussian and Multinomial models, respectively. The Gaussian model, specifically, has both very low precision and 0 recall, meaning that it cannot identify instances of the positive class at all. The Multinomial model has higher precision and a non-null recall, so it produces less false negatives, but still struggles with false positives, predicting many *non-techno* tracks as *techno* ones.

Class	Prec.	Rec.	F1	AUROC
techno	0.04	0.00	0.01	0.74
not techno	0.95	1.00	0.97	0.74

(a) Summary of the main classification measures for the binary problem with Gaussian distribution.

Class	Prec.	Rec.	F1	AUROC
techno	0.08	0.43	0.14	0.74
not techno	0.96	0.75	0.84	0.74

(b) Summary of the main classification measures for the binary problem with Gaussian distribution.

3.4 Comparison Between the ROC Curves

To better compare the performance of the classifiers applied to the dataset, we computed the ROC curves for both the multiclass and binary tasks, since they offer valuable insights on performance evaluation: in particular, the Area Under the ROC Curve (AUROC) assesses the overall discriminative capacity of the classification model. Regarding the multiclass task, through examinating the ROC curves for K-Nearest Neighbors (Figure 38) we can see that highlighted the genres *study*, *sleep* tied with *iranian* and *forro* as the most recognizable, and genre *afrobeat* as the least distinguishable overall (the areas under the curve are respectively: 0.97, 0.95, 0.94 and 0.78); the ROC curves for the Decision Tree (Figure 39) showed instead genres *study*, *sleep*, tied again with *iranian*, and *black-metal* as the most distinguishable and genre *afrobeat* as the worst distinguishable, as it was for the K-NN instance, here the areas under the curve are respectively: 0.95, 0.93, 0.91 and 0.76. In our cases, where the values are close to 1, it indicates that our models demonstrate strong performance.

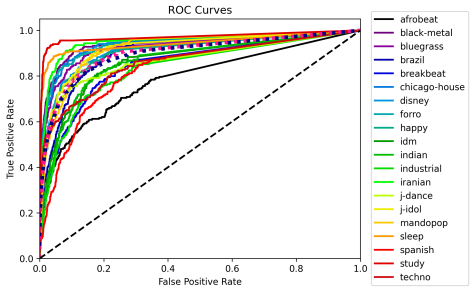

Figure 38: ROC curves for K-NN.

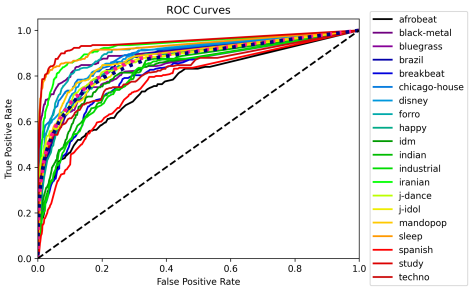

Figure 39: ROC curves of Decision Tree.

Both the Gaussian (Figure 40) and Multinomial Naive Bayes Classifier (Figure 41) ROC curves paint a noticeably different picture: for the Naive Gaussian case, genres *iranian* and *study* are the most recognizable, while *afrobeat*, *breakbeat* and *industrial* were poorly distinguishable, due to the presence of the already mentioned null-valued conditional probabilities (respectively AUROC: 0.92, 0.90, 0.70, 0.61, 0.63).
The ROC curves for the Multinomial Naive Bayes showed, instead, that genre *iranian* was the most recognizable one overall, while genres *breakbeat* and *techno* tied up with *industrial* were the least distinguishable ones with AUROC: 0.92, 0.61, 0.58.
This curves yielded noticeably inferior results compared to previous observations, particularly in the multinomial case. Notably, the area under the ROC curve for the *techno* genre is very close to 0.5: this suggests that the model, when classifying instances within this genre, tends to perform similarly to random chance.

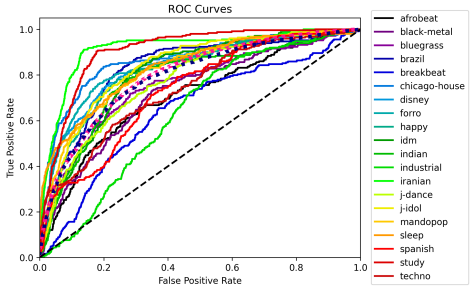

Figure 40: ROC curves for Naive Gaussian.

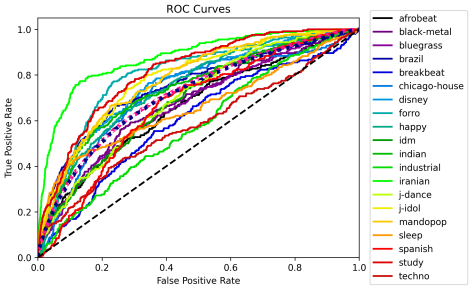

Figure 41: ROC curves for Naive Multinomial.

In the binary case, we observe that KNN returns again a slightly higher value than decision trees, indicating once more a consistent and good model performance across both scenarios.

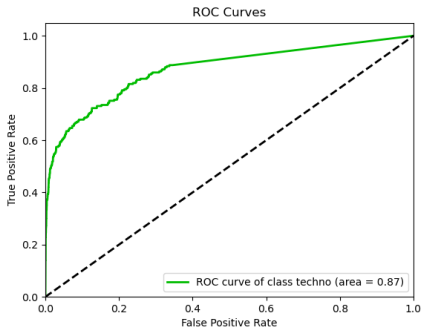

Figure 42: ROC curves for K-NN.

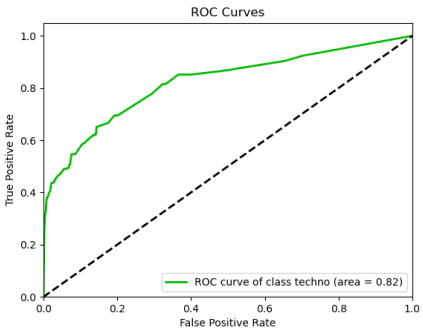

Figure 43: ROC curves for Decision Tree.

Regarding Naive Bayes, whether considering the Gaussian or Multinomial distribution, we obtain satisfying results, although clearly worse than the other models described above.

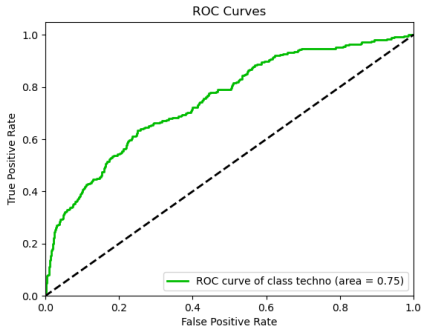

Figure 44: ROC curves for Naive Gaussian.

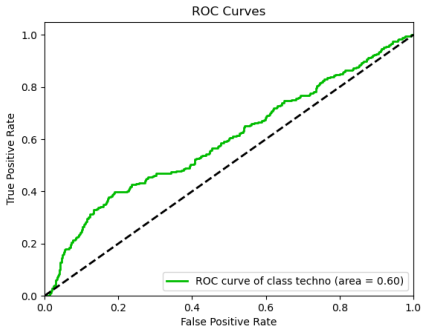

Figure 45: ROC curves for Naive Multinomial.

3.5 Conclusions

The outcomes of our analysis reveal a notable discrepancy among the three models, with the Naive Bayes classifier showcasing the least favorable performance, both for the multiclass and the binary classification tasks. The reported values for precision, recall, and F1-score pertaining to the positive class are close to or equal to 0. This implies that the Naive Bayes classifier predominantly predicts instances belonging to the negative class. The K-Nearest Neighbors and Decision Tree models exhibit superior performance in both multiclass and binary classification task. The precision, recall, and F1-score metrics for these models indicate a more balanced and effective classification across positive and negative classes, affirming their efficacy in capturing the underlying patterns in the data.

4 Regression

In this section, we illustrate how we analyzed the dataset using different regression models. We first considered a univariate regression task, choosing the variable *danceability* as the target, and later a multivariate task, using the variables *danceability* and *instrumentalness* as target variables. All other continuous variables were used as input. We used both linear models (with and without regularization), and non-linear models (Decision Tree and K-Nearest Neighbors); for the latter, we also performed a grid search to identify the best hyperparameters, exploring the same hyperparameter space defined for the classification task (the only difference here being the splitting criterion for the decision tree, chosen between either *squared_error* or *absolute_error*). The best hyperparameters found for the Decision Tree regressor are:

- *max_depth*: 21 (both tasks)
- *min_samples_leaf*: 64 (univariate), 32 (multivariate)
- *min_samples_split*: 2 (both tasks)
- *criterion*: *squared_error* (both tasks)

The bets hyperparameters found for the K-NN regressor are:

- $k = 20$ (univariate), 30 (multivariate)
- *distance*= *cityblock* (both tasks)
- *weight* = *uniform* (both tasks)

Tables 5 and 6 summarize the different test scores we found for each model used.

As evidenced by the reported scores, the K-Nearest Neighbors model performs best, followed by the Decision Tree regressor. Linear models produce higher errors and lower R^2 scores (especially when regularized with L_1 norm); when comparing test and training errors, they were

Model	R^2 score	MSE	MAE
Linear	0.356	0.024	0.125
Linear with ridge regression	0.356	0.024	0.125
Linear with lasso regularization	0.002	0.037	0.153
Decision Tree	0.541	0.017	0.101
K-NN	0.590	0.015	0.095

Table 5: Scores calculated for the univariate regression task.

Model	R^2 score	MSE	MAE
Linear	0.290	0.069	0.205
Linear with ridge regression	0.290	0.069	0.205
Linear with lasso regularization	0.039	0.086	0.243
Decision Tree	0.424	0.056	0.164
K-NN	0.503	0.050	0.158

Table 6: Scores calculated for the multivariate regression task.

found to be really close, suggesting underfitting caused by the model's inflexibility. Indeed, the other two models are capable of adapting to the data better, since they can actually approximate non-linear relationships between the input and the output.

5 Pattern Mining

This section discusses how we analyzed the dataset via Association Analysis techniques. The *train* dataset was preprocessed, discretizing all numeric variables into quartiles, and appending the name of each variable to the corresponding values in each record. We extracted frequent itemsets and association rules, and then used some of those rules to predict the values of two different target variables, *genre* and *mode*, using the dataset *test* to calculate their accuracy. The algorithm chosen for both itemset and rule extraction is FP-growth.

5.1 Frequent Itemsets Extraction

We executed the FP-growth algorithm multiple times with different values of the *minsup* threshold, between 1% and 25%, in order to study the change in the number of frequent maximal, closed, and all itemsets found, visualized in Figure 46.

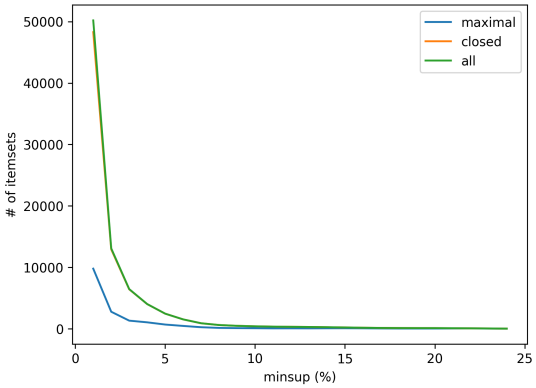

Figure 46: Number of frequent itemsets found for different values of *minsup*.

The total number of frequent itemsets and the number of closed frequent itemsets is very close for all *minsup* thresholds we tested for. The number of maximal frequent itemsets is much

lower. As it can be seen in Figure 46, a threshold higher than or equal to 10 produce very few ($<$ 100) maximal itemsets. We ran the algorithm with a *minsup* threshold equal to 1%, obtaining a decently sized output that also contains some useful patterns (used in the next section to generate rules). The maximal itemsets with the highest support are reported in table 7. A very high number of itemsets contain the same items: explicit=False, time_signature=4.0, and mode=1.0; this is expected, since those values are the most frequent for the corresponding variables and appear in most records.

Itemset	**Support (%)**
{instrumentalness=(0.003, 0.7], liveness=(0.098, 0.13], mode=1.0, time_signature=4.0, explicit=False}	3.00
{instrumentalness=(0.003, 0.7], popularity=(24.0, 42.0], mode=1.0, time_signature=4.0, explicit=False}	2.85
{popularity=(42.0, 94.0], tempo=(142.0, 220.5], instrumentalness=(-0.1, 0.003], mode=1.0, time_signature=4.0, explicit=False}	2.76
{duration_ms=(227826.0, 288557.5], popularity=(-0.1, 14.0], mode=1.0, time_signature=4.0, explicit=False}	2.72
{valence=(0.4, 0.7], energy=(0.7, 0.9], instrumentalness=(-0.1, 0.003], mode=1.0, time_signature=4.0, explicit=False}	2.66
{danceability=(0.4, 0.6], popularity=(-0.1, 14.0], mode=1.0, time_signature=4.0, explicit=False}	2.60

Table 7: Maximal itemsets with the highest support, for *minsup* = 1%.

5.2 Association Rules Extraction

We executed FP-growth to extract association rules, fixing the *minsup* threshold at 1% and varying the value of the *minconf* threshold between 65% and 95%. The number of rules found is shown in Figure 47. At higher levels of confidence, most of the rules found were not interesting, since they all involved the same few items. More interesting and varied rules can be found at lower levels of *minconf*. We extracted rules setting a minimum threshold equal to 75%; the rules with the highest lift found for this threshold are reported in Table 8. The first 90 rules with highest lift have a different value of the variable *genre* as the only item in the consequent. The distribution of these rules across genres is shown in Figure 48. Almost all rules at lower values of lift (≤ 1) have either explicit=False or time_signature=4.0 as their consequent.

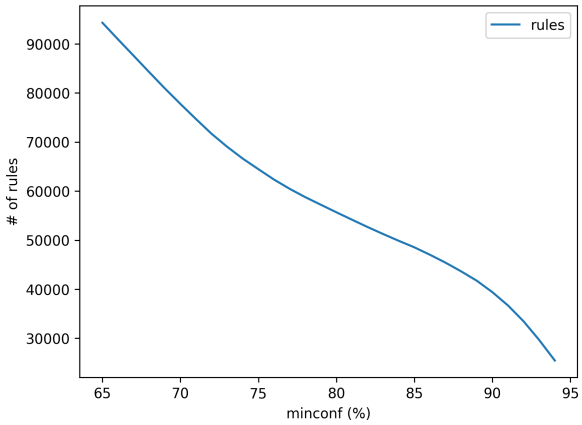

Figure 47: Number of association rules found for different values of *minconf*.

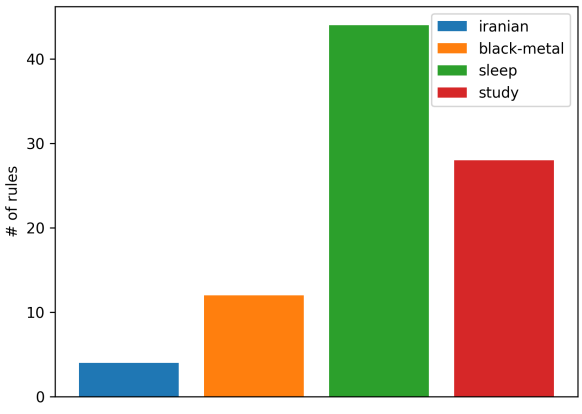

Figure 48: Number of association rules that contain a genre as a consequent.

Antecedent	Consequent	Confidence (%)	Lift
{popularity=(42.0, 94.0], instrumentalness=(0.7, 1.0], valence=(-0.1, 0.2], danceability=(-0.1, 0.4], energy=(-0.1, 0.5]}	genre=sleep	92.1	18.46
{popularity=(42.0, 94.0], instrumentalness=(0.7, 1.0], valence=(-0.1, 0.2], danceability=(-0.1, 0.4], energy=(-0.1, 0.5], explicit=False}	genre=sleep	92.1	18.46
{popularity=(42.0, 94.0], instrumentalness=(0.7, 1.0], danceability=(-0.1, 0.4], energy=(-0.1, 0.5]}	genre=sleep	91.9	18.42
{popularity=(42.0, 94.0], instrumentalness=(0.7, 1.0], danceability=(-0.1, 0.4], energy=(-0.1, 0.5], explicit=False}	genre=sleep	91.9	18.42
{danceability=(0.7, 1.0], instrumentalness=(0.7, 1.0], duration_ms=(8585.9, 180111.8], tempo=(-0.1, 100.0], time_signature=4.0}	genre=study	91.5	18.29
{danceability=(0.7, 1.0], instrumentalness=(0.7, 1.0], duration_ms=(8585.9, 180111.8], tempo=(-0.1, 100.0], time_signature=4.0, explicit=False}	genre=study	91.5	18.29

Table 8: Rules with the highest lift found for $minconf = 75\%$.

5.3 Using Association Rules to Predict a Target Variable

We isolated a few interesting rules from the high support and high confidence ones found in the previous section, and assessed their accuracy in predicting their consequent. We chose the rules with the highest lift which have as their consequent the variables *mode* or *genre*. These rules and their accuracy are reported in Table 9.

Rule	Confidence (%)	Lift	Accuracy
(key=10, popularity=(-0.1, 14.0], time_signature=4.0) → mode=0.0	82.5	2.55	0.86
(key=10, popularity=(-0.1, 14.0]) → mode=0.0	79.1	2.45	0.89
(key=0, liveness=(0.098, 0.13], instrumentalness=(-0.1, 0.003], time_signature=4.0, explicit=False) → mode=1.0	92.9	1.37	0.84
(key=0, popularity=(42.0, 94.0], speechiness=(-0.1, 0.04]) → mode=1.0	91.7	1.35	0.83
(popularity=(42.0, 94.0], instrumentalness=(0.7, 1.0], valence=(-0.1, 0.2], danceability=(-0.1, 0.4], energy=(-0.1, 0.5]) → genre=sleep	92.1	18.46	0.87
(danceability=(0.7, 1.0], instrumentalness=(0.7, 1.0], duration_ms=(8585.9, 180111.8], tempo=(-0.1, 100.0], time_signature=4.0) → genre=study	91.5	18.29	0.79
(popularity=(14.0, 24.0], energy=(0.9, 1.0], speechiness=(0.09, 0.6], valence=(-0.1, 0.2], danceability=(-0.1, 0.4], explicit=False) → genre=black-metal	85.2	17.07	0.84

Table 9: Association rules tested for classification and relative measures.

All rules have high accuracy in predicting the specified class. It's interesting to draw parallels between the rules that classify different genres, and the decision tree obtained with an actual Decision Tree model shown in Figure 33. Following the path that goes from the root of the tree to the nodes that contain the tracks that belong to the genre *sleep*, it is classified as a low *danceability*, high *popularity*, and low *valence* genre. Indeed, by looking at the corresponding rule on the table above, we can see how the genre is associated with danceability=(-0.1, 0.4], popularity=(42.0, 94.0], and valence=(-0.1, 0.2]. The same thing can be observed for *study* (high *danceability*, high *instrumentalness*, low-medium *duration_ms*), and *black-metal* (low *danceability*, low-medium *popularity*, high *energy*).