UNIVERSITÀ DI PISA

DIPARTIMENTO DI INFORMATICA

DATA MINING 1 PROJECT

Autori:

Lorenzo Cascone (581465)

Emiliano Marrale (545552)

Salvatore Puccio (547518)

ANNO ACCADEMICO 2023/2024

Contents

1 Data understanding and preparation

1.1 Data semantics

We dispose of two datasets, one for training and one for test. Both contains a list of Spotify songs of 20 different music genres. The train dataset contains 15000 instances, with 24 different columns.

The following list describes each variable:

- **Nominal/Categorical variables**:
  - **key**: key the track is in. Integers map to pitches using standard Pitch Class notation;
  - **name**: name of the track;
  - **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;
  - **genre**: the genre in which the track belong.
  - **time_signature**: estimated time signature;
- **Discrete symmetric binary variable**:
  - **explicit**: whether or not the track has explicit lyrics;
  - **mode**: mode indicates the modality (major or minor) of a track;
- **Continuos numerical attributes**:
  - **duration_ms**: track lenght in milliseconds;
  - **loudness**: overall loudness of a track in decibels;
  - **tempo**: estimated tempo of a track in beats per minute (BPM);
  - **features_duration_ms**: duration of the track in milliseconds;
  - **n_beats**: total number of time intervals of beats throughout the track;
  - **n_bars**: total number of time intervals of the bars throughout the track;
- **Numeric Ratio_scaled**:
  - **valence**: measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track;
  - **instrumentalness**: whether a track contains no vocals the closer the instrumentalness value is to 1.0 (continuous measure);
  - **acousticness**: confidence measure from 0.0 to 1.0 of whether the track is acoustic;
  - **speechiness**: presence of spoken words in a track;
  - **liveness**: predicts the presence of an audience in the recordings;
  - **Danceability**: describes how suitable a track is for dancing. A value of 0.0 is least danceable and 1.0 is most danceable;
  - **Energy**: is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity;
  - **popularity_confidence**: the confidence, from 0.0 to 1.0, of the popularity of the song;
  - **popularity**: popularity of a track is a value between 0 and 100, with 100 being the most popular;

1.1.1 Distribution of variables

There are 750 songs for each genre, meaning the training dataset is balanced between the different genres. There are no duplicates among the records and songs. By looking at some plots and statistics like mean, minimum and maximum values for each feature, we noticed that **loudness** feature has 14979 values smaller than 0, this is a strange result but dB values also depends on the setting/context of the situation. For instance, when dealing with professional sound equipment, 0 dB usually refers to the loudest level before distortion begins. The bar chart below shows the popularity of each genre present in the dataset.

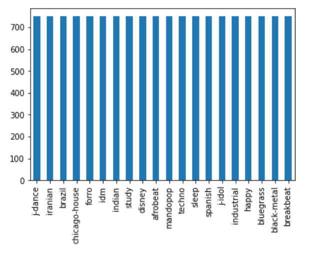


Figure 1: Songs for each genre

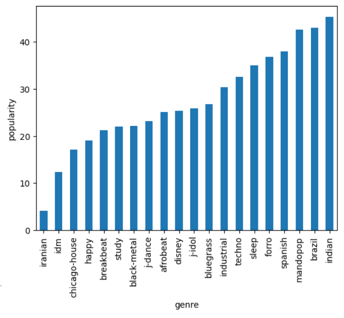


Figure 2: Most popular genres

The most energetic genres according to the dataset are 'happy', 'black-metal' and 'industrial' and the least are 'disney', 'sleep' and 'study', that goes in contrast with the most danceable where 'study' stands as second. The distribution of numerical features is as follows:

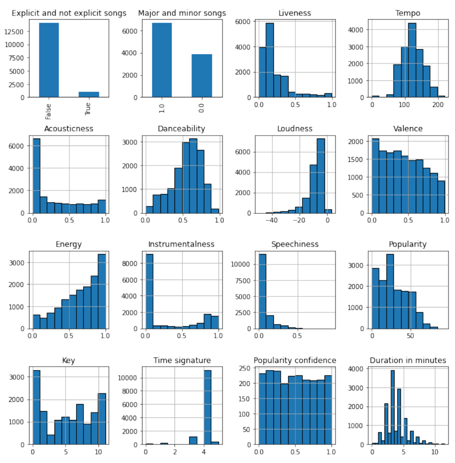


Figure 3: Data distribution

As expected the majority of the songs are not instrumental or acoustic, we can point this at the music industry trends. It's clear that most songs are not explicit and in major mode. Also there is a prevalence of low values in both acousticness and liveness. The **normal distribution** of **energy**, **liveness** and **loudness** is *negatively skewed* (Mean < Median < Mode) while **popularity** and **valence** are both *positively skewed* (Mean > Median > Mode).

1.1.2 Assessing Data Quality

After a quick glimpse at the data, it has been decided to check for columns containing NaN values and identified that the columns **mode**, **time_signature** and **popularity_confidence** contains NaN values, precisely: **mode**: Has 4450 NaN values; **popularity_confidence**: Has 12783 NaN values. That's 85.22% of the total number of entries; **time_signature**: Has 2062 Nan values. Handling missing values The time signature feature can be estimated by the ratio of the number of beats in a bar to the number of bars. The formula to calculate the time signature is as follows: $TimeSignature = n\_beats \ / \ n\_bars$. For example, if a song has 96 beats and 24 bars, it is possible to calculate the time signature as: $TimeSignature = 96/24 = 4$ In this case, the time signature would be 4/4, which is a common time signature in music, indicating four beats in each

bar.
So we proceeded to substitute missing values for the feature **time_signature** using the formula described above.

The **popularity_confidence** feature has been dropped because of its 12783 NaN values.

For the feature **mode** we had a different approach, we used the RandomForest algorithm to predict its values, with an accuracy of 0.70 and using 500 estimators. Before choosing the 500 estimators we did some tests and found out that even if we used 1000 estimator or 2000 the accuracy would basically remain the same.

1.1.3 Outliers

The presence of outliers was detected through boxplots, it was decided to leave them in the dataset and remove them during centroid-based clustering in order to make a comparison.

1.2 Variable transformation

Before using clustering algorithms all the continuos numerical values have been normalized using the MinMax normalization technique. We transformed the feature 'explicit' values from Boolean to Integer and mapped the feature 'genre' with a number for each genre.

1.3 Pairwise correlations and eventual elimination of variables

It's important to note that correlations do not imply causation. Just because two variables are correlated doesn't mean that one causes the other. Additionally, the strength and direction of correlations can vary, and sometimes correlations may be weak or not significant. After reading the description.txt file, it's easy to notice that the features **duration_ms** and **features_duration_ms** are almost the same; we then noticed that they have only 5832 different values, where 5756 of them differed by 1. Also there is a high correlation as we can see from the scatter plot 4.

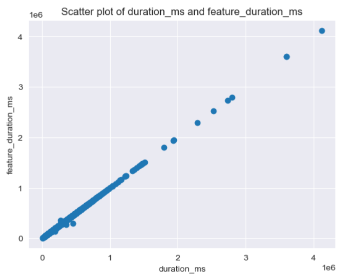


Figure 4: Correlation of duration_ms and feature_duration_ms

We decided to leave the "processing" feature, which while not described, by doing some research we thought was a variable related to audio processing. In order to identify correlation between features, we are going to calculate **Pearson's correlation coefficient** for each possible pair of features.

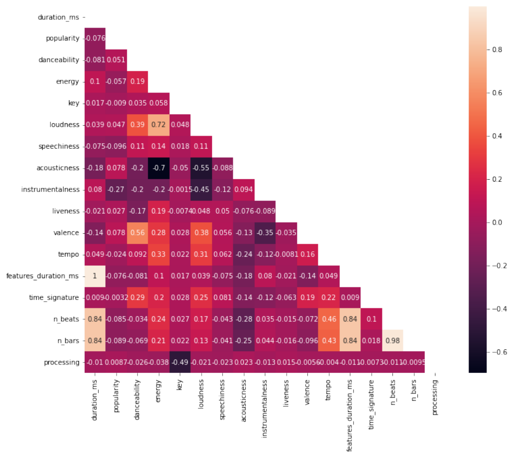


Figure 5: Pearson correlation heatmap

Looking at pearson we can say that there is:

Strong correlation

- acousticness and energy
- loudness and energy
- n_beats and n_bars
- acousticness and loudness
- valence and danceability

Moderate Correlation

- loudness and danceability
- tempo and energy
- valence and loudness
- tempo and loudness
- loudness and instrumentalness
- tempo and n_beats
- tempo and n_bars
- key and processing
- instrumentalness and loudness

As expected valence and danceability are highly correlated, dance songs are usually happier. Also acousticness is highly negatively correlated with energy and loudness, that's because it's very rare that a song played by an orchestra or acoustic band is energetic or loud. We can also see that acousticness, for the same reasons, is uncorrelated with popularity.

2 Clustering

2.1 Analysis by centroid-based methods

2.1.1 K-Means

The kind of dataset we have has a particular distribution of data that doesen't allow us to find well defined clusters with a low SSE value and a high silhouette. The choosen continuous variables for the clustering are danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo, n_beats, duration_ms and processing. The features n_bars and feature_duration_ms were removed for their high correlation with n_beats and duration_ms. We first remove the outliers with the IQR method dropping the most of them, the result is a better cluster creation, given the sensitivity of k-means to outliers. To choose the right number of clusters K we check the SSE curve, the Silhouette and the Elbow Method.

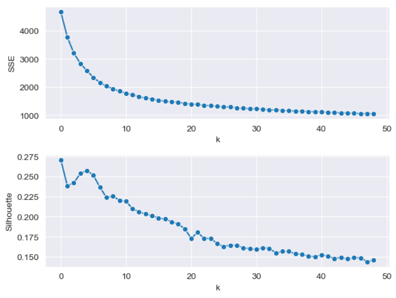


Figure 6: SSE score and Silhouette score

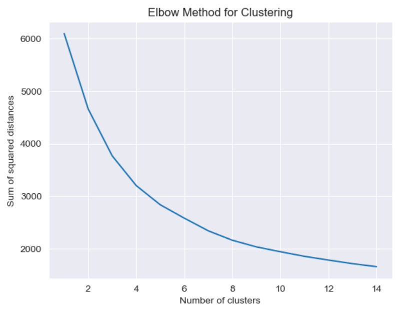


Figure 7: Elbow method

In the end k=8 has been decided, with a SSE score of 2158 and a Silhouette score of 0.23. Looking at the clusters through scatterplots wasn't satisfying, with the clusters almost always overlapping eachothers or being very close. The best two good plots we got are 8 and 9 where we can see the overlap of cluster 1 and 2, 0 and 4 in the 8 image and of cluster 0, 4 and 7 in the other image; but overall a good separation.

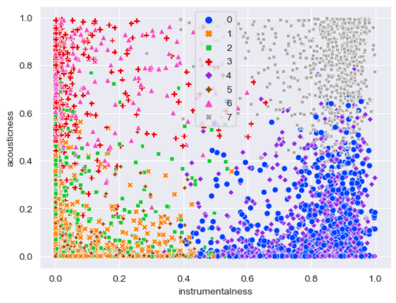


Figure 8: Instrumentalness and Acousticness

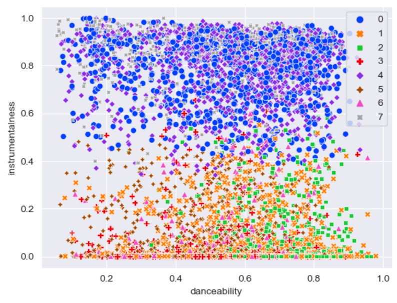


Figure 9: Instrumentalness and Danceability

To have a better a look at the clusters we used the PCA dimensionality reduction technique used to transform high-dimensional data into a lower-dimensional space while preserving as much variance as possibile. We used it to visualize, reduce the data in two dimensions and visualize the clusters in a lower-dimensional space to provide insights into how well the clustering algorithm separated the data points into distinct groups. Note that PCA is a tool for understanding the structure of the data but may not be optimized for visualizing clusters specifically.

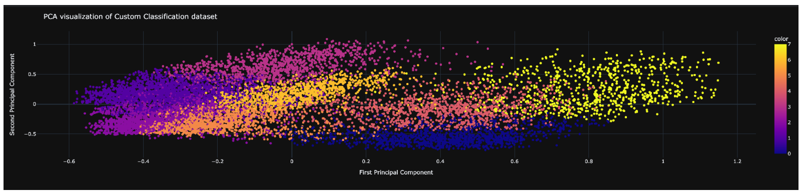


Figure 10: PCA Visualization

We also used the T-distributed Stochastic Neighbor Embedding (t-SNE) for our analysis to emphasize the structure of clusters. After testing with many perplexities values we settled on a perplexity of 7.

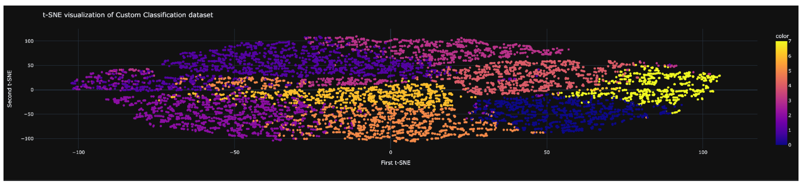


Figure 11: t-SNE Visualization

In another analysis we did, after the clustering, we crossed the original dataset with the dataset used for k-means and added in this last one the columns popularity and genre. In the case of 'popularity' the songs were distributed almost identically among the various clusters. In the case of 'genre', however, we noticed that clusters 1 and 5 contained more songs from the genres: j-dance, Iranian, black-metat, bluegrass, happy, industrial, and j-idol. Clusters 0 and 4, 3 and 6 contains almost the same genres of songs. Cluster 7, on the other hand, contains almost half of songs belonging to the genre spanish.

Also we tried to cluster without the feature 'processing' resulting in a much better defined clustering, the elbow method suggested a k=6 and the silhouette score and SSE reached were of 0.26 for the silhouette and 1563 for the SSE. Also this time we noticed an high distance between values in the case of danceability and instrumentalness and in the case of instrumentalness and acousticness.

2.1.2 Bisecting K-Means

Bisecting K-Means is also been applied. With the same dataset (without outliers and categorical values) but this time with $k = 5$ because of the elbow method. The silhouette score and SSE were also very similar and have been studied with k up to 50.

While studying the scatterplots we noticed that the two most significant ones we had chosen to show in the case of K-Means are very similar, but the clusters looks way better and are more divided. We also tought about joining cluster 2 and cluster 1, but in the end we dropped this hypothesis because it would create a cluster too large. Also by looking at the PCA we could see a better division of the clusters and less overlapping.

2.2 Analysis by density-based methods

In this subsection, we will conduct a thorough analysis of the dataset employing Density-Based Clustering algorithms, specifically DBSCAN, Optics, and Hierarchical DBSCAN. The selected features previously employed in K-Means clustering will be utilized, and outliers will not be removed, as the listed methods are expected to exhibit robustness in handling them.

2.2.1 DBSCAN

The subsequent visuals depict a graphical representation of the distances from the k-th neighbor within the range of [4, 8, 16, 32, 64, 128, 256, 512], offering valuable insights into identifying the optimal pair of parameters, namely **eps** and **min_samples**.

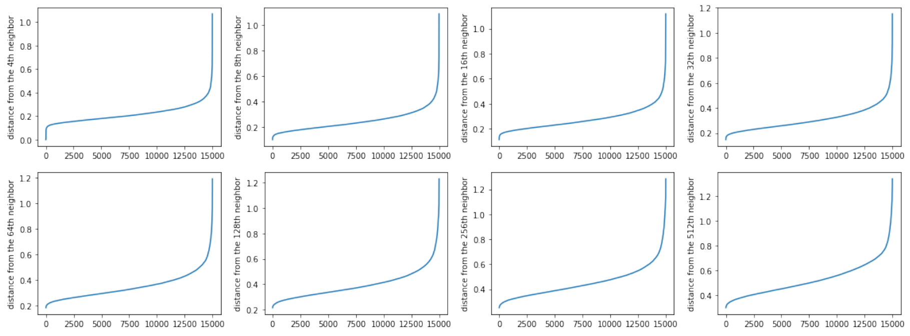


Figure 12: Distance from the k-th neighbor

Subsequently, a manual grid search is initiated to ascertain the optimal silhouette score and further fine-tune the parameter selection. The initial grid search spans from eps=0.2 to eps=0.5, with a step size of 0.1, while k varies from 4 to 32 incrementing by 1. The most favorable silhouette score attained from this search amounts to 0.234. It corresponds to an eps value of 0.5 and min_samples of 16, resulting in a clustering pattern featuring a major cluster comprising 14917 elements, a secondary cluster containing 9 elements, and 74 noise elements.
Continuing the exploration, a subsequent grid search was conducted, extending the eps range from 0.3 to 0.9, while adjusting k within the range of 32 to 512, incrementing by 8. This search resulted in a silhouette score of 0.238, identifying the optimal parameters as eps=0.5 and min_samples=384. The clustering structure revealed 921 noise elements, a major cluster comprising 13052 elements, and an additional cluster containing 1027 elements [13]. These findings underscore the dominance of a large cluster alongside smaller groupings, potentially indicating a need for additional refinement in the clustering process to attain more balanced and meaningful clusters.

As we aimed for a higher silhouette score, which notably led to the formation of a singular large cluster, we extended the exploration beyond this by scrutinizing the grid search outcomes in search of potentially intriguing clustering configurations. We specifically filtered the results to focus on scenarios where the clusters exceeded three. From this examination, we observed the following: setting eps=0.3 and min_samples=64 resulted in the identification of multiple clusters, including one comprising 8347 elements, another with 339 elements, one containing 1807 elements, and an additional cluster housing 287 elements. The remaining 4220 elements were categorized as noise [14].

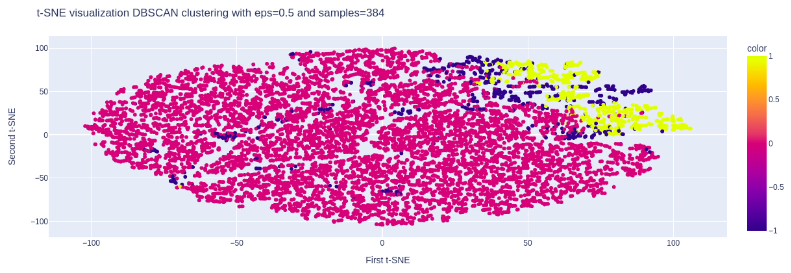


Figure 13: t-SNE DBSCAN with eps=0.5 min_samples=384 silhouette=0.238

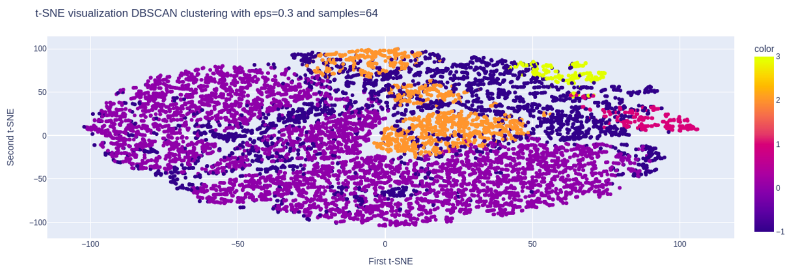


Figure 14: t-SNE DBSCAN with eps=0.3 min_samples=64 silhouette=0.124

The presence of a single prominent cluster encompassing the majority of the data points raises concerns regarding the meaningfulness of the clustering output. With this in mind, we also tried OPTICS and HDBSCAN to check if they might perform better.

2.2.2 OPTICS

Similarly to DBSCAN, we conducted a manual grid search to pinpoint the optimal value for the min_samples parameter. This involved examining silhouette scores for each run, varying min_samples from 4 to 513 with an increment of 8. The most favorable outcome emerged with eps=0.5 and min_samples=44, delivering a silhouette score of 0.266. While this result marginally surpasses DBSCAN, the clustering structure proves to be nonsensical as it marks 14951 elements as noise, with merely 49 elements categorized within a single cluster.

2.2.3 HDBSCAN

Also here, we executed a manual grid search by ranging eps from 0.3 to 0.8 with a step of 0.1, varying min_samples within the set [4,8,16,32,64,128], and exploring min_cluster_size within [50, 100, 300, 500, 750, 1500]. However, the resulting higher silhouette score did not exceed 0.064. Despite these efforts, the clustering structure remained nonsensical across nearly all experiments, consistently displaying one predominant large cluster, a smaller one, and categorizing the majority of elements as noise.

2.3 Hierarchical Clustering (Dendrogram)

We performed a hierarchical clustering using the 'dendrogram', 'linkage', 'fcluster' libraries from 'scipy.cluster.hierarchy' with all the combinations of the linkage methods seen in class ('single','complete' ,'average' ,'ward') and thresholds: 0,5,10,15,20,25,30. We had the best result of silhouette with ward method and threshold of 20.

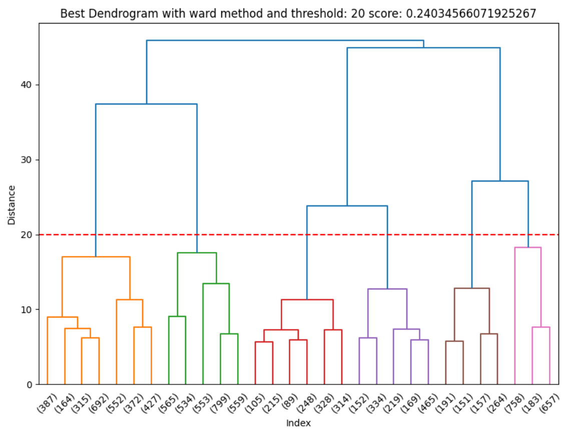


Figure 15: Dendrogram with 6 clusters. (Sizes: C1: 2909, C2: 3010, C3: 1299, C4: 1339, C5: 763, C6: 1598)

We can see that we have 6 clusters and now we can make an analysis about them.

2.3.1 Results Analysis

We can generate box plots for the various columns we've analyzed during clustering. The most strange looking box-plot is the one of the instrumentalness feature. As we can see clusters 1,2 and 6 contain songs with majority of instrumentalness values near zero. That should mean that those songs with a lot of lyrics.
In the other hand clusters 3,4 and 5 have majority of instrumentalness values near to 0. So it is likely that in these clusters there are much more songs with (almost) no lyrics.

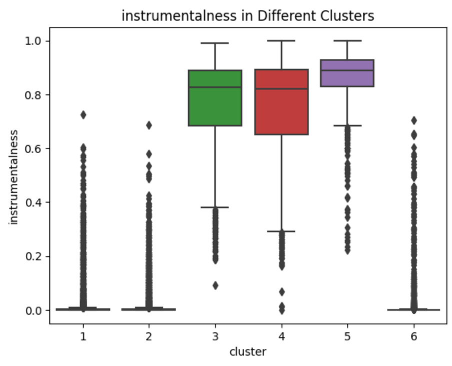


Figure 16: Instrumentalness in clusters

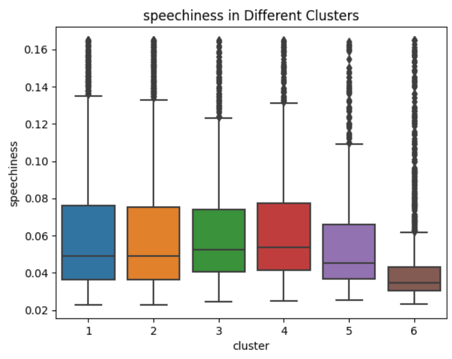


Figure 17: Speechiness in clusters

We initially thought that speechiness was the opposite of instrumentalness. If this was the case we should have seen in speechiness box-plot clusters 1,2 and 6 with really high average values and clusters 3,4,5 with really low values. But in fact we can see that the maximum value in speechiness is 0.165 after outliers removal. Maybe that "feature" in Spotify system does not work well, or simply does not mean what we though.

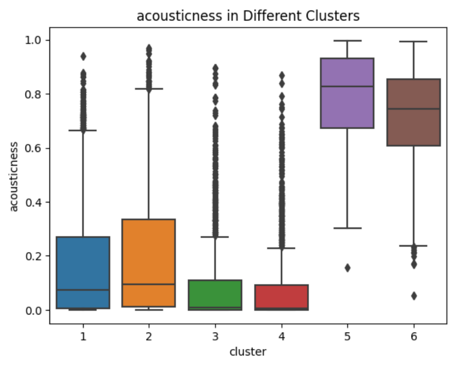


Figure 18: Acousticness in clusters

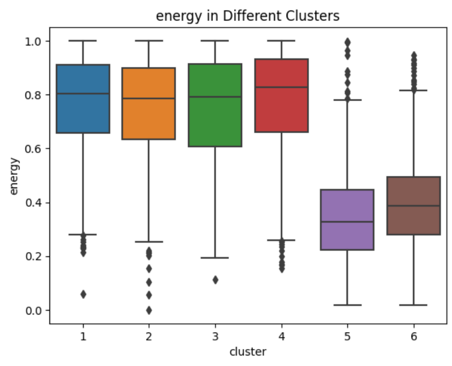


Figure 19: Energy in clusters

Another interesting feature is the 'acousticness':
The box plots indicate that clusters 5 and 6 have higher average acousticness values, suggesting that the songs within these clusters are more acoustic than those in the other clusters. To corroborate this observation, we can examine the energy box plot, which indeed shows that the average energy values for clusters 5 and 6 are significantly lower compared to the others. This aligns with the general perception that acoustic songs tend to be slower and less energetic compared to other types of music.

2.4 Final thoughts

We can say that Density Based clustering (in all its applications) is not a good option for this data-set. One big cluster is detected and nothing interesting shows up. K-Means showed interesting results just like bisecting K-Means, but the visualization wasn't that great for understanding new things about the data-set. Hierarchical clustering was the most interesting. In fact it showed some correlation between the features of the data-set. Also the silhouette score was close to the one of K-means and the number of cluster appears to be more significant in comparison the others.

3 Classification

3.1 KNN

3.1.1 Target variable ’explicit’

Before performing the classification task, we normalized the dataset and removed the categorical variables.

The feature ’explicit’ in our dataset is highly imbalanced, with 93% of the instances labeled as ’false’. To address this, we sought to balance the class distribution before training our classifier. We opted for an oversampling approach, employing the SMOTE to generate artificial examples of the minority class. This will reduce the risk of classifier bias towards the majority class.

In our experiments with the k-Nearest Neighbors (KNN) algorithm, we explored a range of hyperparameters. Specifically, we varied the number of neighbors k from 1 to 50 and tested both ’uniform’ and ’distance’ weighting strategies. We also evaluated the performance using different distance metrics, including ’euclidean’, ’manhattan’ and ’minkowski’.

An interesting trend was observed during the hyperparameter tuning process: as k increased, the accuracy of the classifier decreased. This inverse relationship might be attributed to the characteristics of the synthetic instances generated by SMOTE. With a lower k, the classifier’s decision is heavily influenced by the nearest few instances, which may be more representative of the minority class due to the SMOTE algorithm’s resampling strategy. As k increases, the classifier considers a broader set of neighbors, potentially diminishing the relative influence of these synthetic instances and incorporating more of the majority class characteristics, which could lead to a decrease in accuracy. To further assess the performance of our classifier, we made the confusion matrix, classification_reports and ROC curve at different values of k: 2, 10, and 15. These points were selected to represent a low, moderate, and slightly higher neighborhood size in the KNN algorithm.

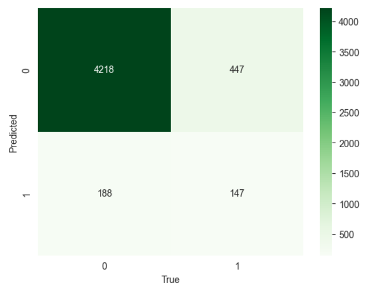


Figure 20: K=2

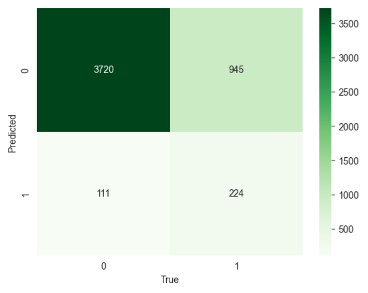


Figure 21: K=10

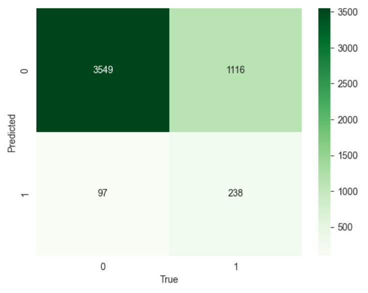


Figure 22: K=15

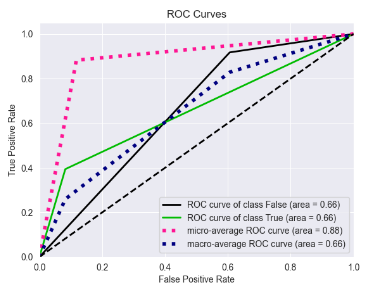


Figure 23: K=2

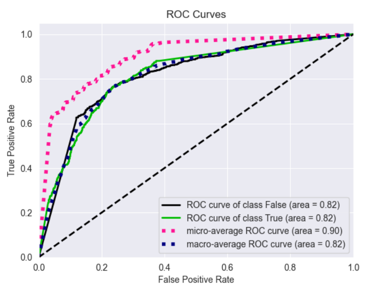


Figure 24: K=10

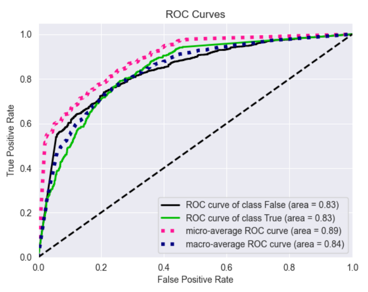


Figure 25: K=15

We then tried to understand how varying the neighborhood size impacts model performance, especially with regards to the minority class in our imbalanced dataset.

- $K == 2$: The confusion matrix shows a large number of true negatives (4218) and a relatively small number of true positives (147), which could indicate that the model is biased towards predicting the majority class, so it's overfitting. The precision for the minority class ('True') was relatively low at 0.25, but the recall was substantially higher at 0.44 compared to higher values of k. This suggests that while the classifier correctly identified nearly half of the actual positive instances, it also misclassified a considerable number of negative instances as positive. The overall accuracy at this point was 0.87, indicating a reasonably high number of correct predictions overall, but with a significant number of false positives;

- $K == 10$ and $K == 15$: we noticed a decrease in precision for the minority class, dropping to 0.19 and 0.18 respectively. This reduction in precision was accompanied by a significant increase in recall, reaching 0.67 for k=10 and maintaining a similar level for k=15 (0.71). The high recall indicates a stronger capability of the classifier to detect the minority class, which is a priority in our case. However, this comes at the cost of incorrectly labeling many negative instances as positive, as reflected by the higher false positive rate.

Also the F1-Value suggests that with increasing k we have a diminishing ability to balance precision and recall, in particular for the majority class. The ROC curve, as expected, suggests that we have a better overall performance when all classes are considered together. We can see that the ROC curve of the positive class performs worse than the negative class, exactly as we expected and probably due to the strong imbalance. In summary, our analysis reveals that lower k values lead to a more balanced precision-recall trade-off, while higher k values, despite improving recall for the minority class, result in lower precision and overall accuracy.

3.1.2 Target variable 'genre'

In order to predict the genre, a grid search cross-validation with a Repeated Stratified K-fold was conducted. The search encompassed k-values ranging from 1 to 123 in increments of 10. Both uniform and distance weights were considered, along with Euclidean and Cityblock distance metrics. Normalization was applied to both the training and test instances using two scaling methods: standard scaler and min-max scaler in order to determine which scaling method produced the most optimal results. The outcomes of this analysis are summarized below:

- **StandardScaler**:
  - Metric: CityBlock
  - K: 31
  - Weights: Distance
  - Accuracy: 0.5132
- **Min-Max Scaler**:
  - Metric: CityBlock
  - K: 31
  - Weights: Distance
  - Accuracy: 0.4542

Looking at the precision, recall, and F1-score for each class, we can observe varying performance across different classes. For instance:

- Class 8 has relatively high precision, recall, and F1-score (0.87, 0.75, 0.81), indicating better performance in predicting this class.
- Classes 13, 2, 16 also have reasonably high precision, recall, and F1-scores.
- Classes like 7, 15, 18, 19 have relatively lower precision, recall, and F1-scores, suggesting poorer model performance in predicting these classes.

Figure 26: Precision, Recall and F1-Score - StandardScaler

N°	**Class**	**Precision**	**Recall**	**F1-Score**
0	j-dance	0.51	0.63	0.56
1	iranian	0.63	0.56	0.59
2	black-metal	0.64	0.66	0.65
3	bluegrass	0.49	0.65	0.56
4	happy	0.42	0.49	0.45
5	industrial	0.39	0.36	0.37
6	j-idol	0.43	0.69	0.53
7	spanish	0.31	0.22	0.25
8	sleep	0.87	0.75	0.81
9	techno	0.64	0.49	0.55
10	mandopop	0.34	0.67	0.45
11	afrobeat	0.38	0.23	0.29
12	disney	0.59	0.50	0.54
13	study	0.77	0.86	0.81
14	indian	0.48	0.31	0.38
15	idm	0.56	0.27	0.36
16	forro	0.57	0.71	0.63
17	chicago-house	0.53	0.68	0.60
18	brazil	0.46	0.29	0.35
19	breakbeat	0.39	0.27	0.32
Accuracy		0.51		
Macro Avg		0.52	0.51	0.50
Weighted Avg		0.52	0.51	0.50

While precision, recall, and accuracy, in general, demonstrate poor performance, a closer examination of the ROC Curve reveals an AUC exceeding 0.82 for every class included. This implies that the model's predictions are well-discriminated for all classes. It suggests that the model is making good distinctions between positive and negative instances for each class across different thresholds.

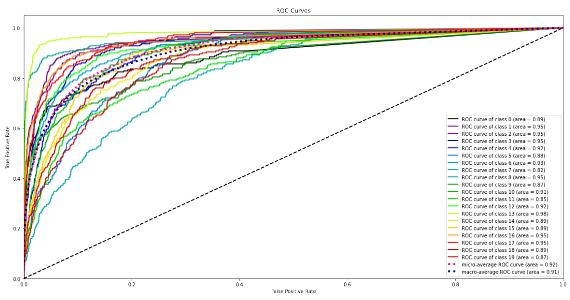


Figure 27: ROC Curve for KNN - Genre

With a small number of samples per class (750 for each), the evaluation metrics' estimates may not be statistically significant or reliable since random fluctuations in a small dataset can significantly impact performance metrics. Also models trained on small datasets per class might overfit to the limited patterns present in those samples, resulting in poor generalization to unseen data. With this in mind we decided to join classes into broader classes, in order to understand if we might get better results. We created the following broader classes:

Dance and Electronic	**Regional and Cultural**
j-dance	Iranian
Techno	Spanish
Idm	Mandopop
Chicago-house	Indian
Breakbeat	Afrobeat
Industrial	Brazil
	Forro
Mood or Theme-Based	**Unique or Specific Genres**
Happy	Black-metal
Sleep	Bluegrass
Disney	
Study	

The results of the classification are worse, since we have a 0.76 accuracy on the train set (as the grid_best_score), and a 0.12 accuracy on the test set. The substantial difference between the training and test accuracies suggests that the model might not generalize well to unseen data. This difference can be an indication of overfitting.

3.2 Naive Bayes

3.2.1 Target variable ’explicit’

To evaluate our model’s performance we analyzed several metrics beyond accuracy, given the disproportionate class sizes before the application of SMOTE.

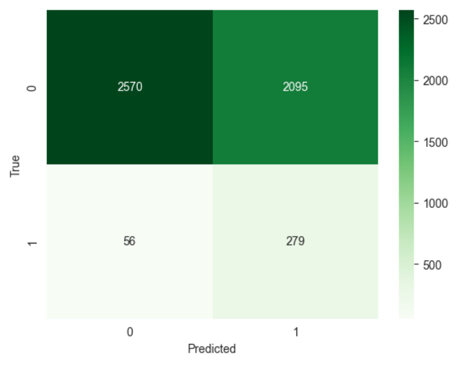


Figure 28: Confusion Matrix

	Classification$_r$			
	precision	recall	f1-score	support
Class 0	0.98	0.55	0.70	4665
Class 1	0.12	0.83	0.21	335
accuracy			0.57	5000
macro avg	0.55	0.69	0.46	5000
weighted avg	0.92	0.57	0.67	5000

Figure 29: Classification report

The confusion matrix revealed a big disparity between the model’s ability to identify non-explicit and explicit content. With a precision of 0.98 for the non-explicit class (0), the model demonstrated high reliability in its positive predictions for this class. However, this high precision came with a recall of only 0.55, suggesting that while our model was accurate when it identified a song as non-explicit, it still missed a substantial portion of this class. Conversely, the explicit class (1) exhibited a precision of 0.12, indicating a high rate of false positives, but the recall was notably higher at 0.83, meaning the model was quite effective at capturing most of the explicit content. The precision-recall curve underscores this trade-off, showing a high area under the curve (AUC)

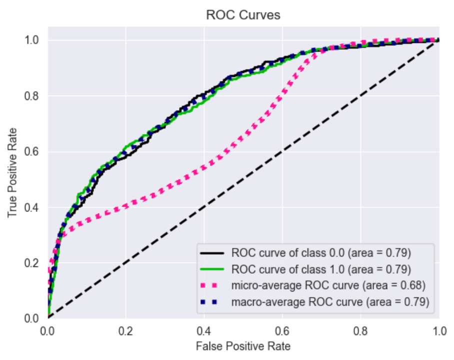


Figure 30: ROC Curve

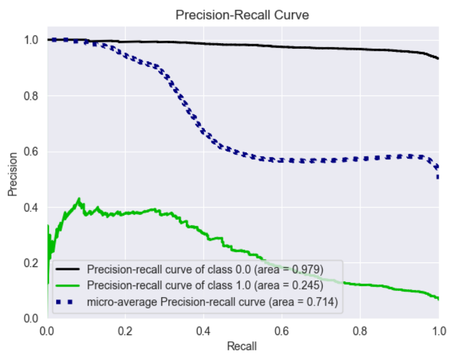


Figure 31: Precision Recall Curve

of 0.979 for the non-explicit class, indicative of the model’s performance in correctly identifying non-explicit songs. The curve for the explicit class, however, have a much lower AUC of 0.245, reflecting the model’s challenges in balancing the precision-recall trade-off for this minority class. The ROC curve shows a quite optimistic view with an AUC of 0.79, suggesting that the model has a comparable capability to discriminate between explicit and non-explicit content.

3.2.2 Target variable ’genre’

Naive bayes performs much worse than KNN, with an accuracy score of 0.26, having overall very low precision and recall for every class.
Also here the ROC curve is not that bad, but still performing worse than KNN.

Genre	Precision	Recall	F1-score
j-dance	0.12	0.38	0.18
iranian	0.60	0.33	0.42
black-metal	0.26	0.08	0.12
bluegrass	0.26	0.08	0.13
happy	0.24	0.36	0.28
industrial	0.06	0.00	0.01
j-idol	0.25	0.60	0.35
spanish	0.21	0.17	0.19
sleep	0.82	0.44	0.58
techno	0.22	0.10	0.13
mandopop	0.21	0.52	0.30
afrobeat	0.20	0.03	0.06
disney	0.36	0.31	0.33
study	0.26	0.75	0.39
indian	0.25	0.05	0.08
idm	0.35	0.18	0.24
forro	0.58	0.24	0.34
chicago-house	0.34	0.55	0.42
brazil	0.21	0.05	0.08
breakbeat	0.15	0.05	0.08

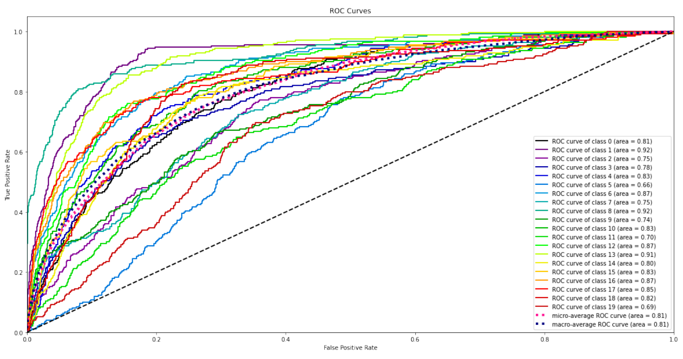


Figure 32: ROC Curve for KNN - Genre

3.3 Decision Tree Classifier

3.3.1 Target variable 'explicit'

As before the feature presented a significant class imbalance. To address this, we implemented the Synthetic Minority Over-sampling Technique (SMOTE), enhancing the representation of the minority class to create a more balanced training environment for our models.

For hyperparameter tuning we used GridSearchCV. The final model was selected based on its ability to balance overall accuracy with recall for the explicit class. Our chosen Decision Tree Classifier was configured with entropy as the criterion, a maximum depth of 36, and a minimum sample split of 0.0001. Recognizing the disproportionate misclassification costs, we employed a class weight strategy, giving ten times more weight to the minority class (explicit content).

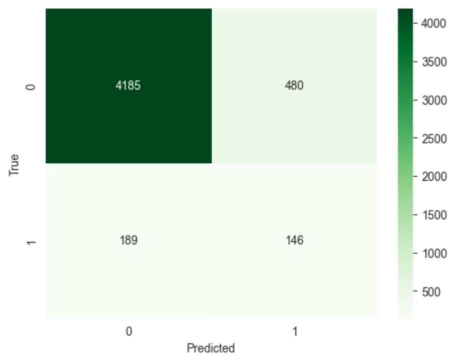
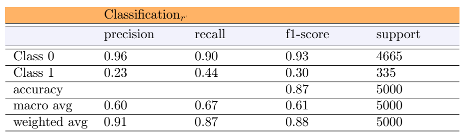


Figure 33: Confusion Matrix



Classification_r

	precision	recall	f1-score	support
Class 0	0.96	0.90	0.93	4665
Class 1	0.23	0.44	0.30	335
accuracy			0.87	5000
macro avg	0.60	0.67	0.61	5000
weighted avg	0.91	0.87	0.88	5000

Figure 34: Classification report

The performance of our Decision Tree Classifier, as summarized by the classification report, indicates a model that effectively recognizes non-explicit songs with a precision of 0.96 and a recall of 0.90. However, the recall for explicit songs stood at 0.44, a very bad result. Ensuring explicit content is correctly flagged in our application of the algorithm is the most important thing, even at the expense of misclassifying some non-explicit songs. The confusion matrix offers a visual representation of our model's performance, echoing the classification report's findings with more false positives then true positives given the recall of 0.44. One of the things we noticed was the importance of features like 'speechiness', 'loudness' and 'genre' in the prediction of explicit and non-explicit songs as highlighted by the feature_importance_ attributes.

3.3.2 Target variable 'genre'

Here aswell a gridsearch was conducted with the following parameters:

- 'max_depth': [2, 3, 5, 10, 20, 30 , 40, 50, 60]
- 'min_samples_leaf': [5, 10, 20, 50, 100, 125, 150, 200]
- 'criterion': ["gini", "entropy"]
- Cross validation: 5

Identifying the best setup with

- criterion: gini
- max_depth: 10
- min_samples_leaf: 10
- accuracy: 0.461

The classification report produced for the Decision Tree classifier is very similar to the one produced with KNN. We will talk about it more in the conclusion subsection. We can see the feature importance computed by the classifier.

- **Popularity (0.229)**: This feature seems to be the most influential according to the decision tree model, suggesting that it has the highest predictive power among all the features.
- **Danceability (0.126) and Duration_ms (0.125) Acousticness (0.119), Loudness (0.083), Instrumentalness (0.078), and Valence (0.076)**: These features are the next most important after "Popularity." They have moderate importance but are not as influential as "Popularity."
- **Speechiness (0.052), Energy (0.047), Tempo (0.037), n_beats (0.011), and Liveness (0.011)**: These features show a lower level of importance compared to the preceding ones but still contribute to the model's predictive ability.
- **Explicit (0.003), Key (0.001), Mode (0.001), Processing (0.0005), and Time_Signature (0.0)**: These features seem to have minimal or no impact on the model's decision-making process, suggesting that they might not significantly contribute to predicting the target variable based on the tree's split criteria.

3.4 Conclusions

3.4.1 Discussion of 'explicit' variable

The **Decision Tree Classifier**, despite its interpretability, yielded a precision of 0.23 and recall of 0.44 for the explicit class, resulting in an F1-score of 0.30, hinting at the fact that the ability of decision tree of correctly classifying explicit content is very poor, as the confusion matrix confirms. While its ROC curve indicated a fair discriminative ability, thanks to the correct predictions for the non-explicit class, with an AUC of 0.67, the model worked pretty well but failed in the intent of classifying the minority class.

The **Naive Bayes** algorithm demonstrated a similar challenge, with a an higher recall for the explicit class but at the cost of misclassifying almost half of the non-explicit songs due to the low precision of the explicit class. As highlighted by the f1-score the model is relatively good at identifying non-explicit content but on the other hand struggle at identifying the explicit content. That's why we have 2095 false negative out of 4665 values in the confusion matrix, and 279 true positives out of 335 values even with low accuracy.

The **KNN** algorithm, particularly with $k = 15$ stood out as the superior model. It achieved a robust balance between precision and recall, reflected in a higher F1-score and a more favorable position on the precision-recall curve, with a score of 0.978 for the class 0 (non-explicit), means that when the model predicts a song is not explicit, it is correct most of the time. The precision-recall curve score for class 1 (explicit) is very low, at 0.245, indicating that the model tries to capture more of the actual explicit songs (increasing recall), the precision drops significantly, meaning that there are many non-explicit songs that are incorrectly labeled as explicit. Given the importance of accurately identifying the minority class (explicit) KNN with k=15 emerged as the most suitable model for our needs.

3.4.2 Discussion of 'genre' variable

Regarding the prediction of the target variable 'genre', the algorithm that performed worse is definitely the naive Bayes, with an accuracy of 0.26 accompanied by a discouraging classification report, and it had the worst ROC curve among the three algorithms. Therefore, for the classification of genre, we would opt for either KNN or Decision Tree, as they both show very similar results in terms of accuracy, classification report, and ROC curve. In the table below, the classification reports of KNN and Decision Tree are compared, and the similarity between the two is quite evident.

Genre	Decision Tree Precision	KNN Precision	Decision Tree Recall	KNN Recall	Decision Tree F1-Score	KNN F1-Score
j-dance	0.50	0.51	0.51	0.63	0.50	0.56
iranian	0.53	0.63	0.63	0.56	0.58	0.59
black-metal	0.66	0.64	0.71	0.66	0.68	0.65
bluegrass	0.48	0.49	0.40	0.65	0.44	0.56
happy	0.39	0.42	0.50	0.49	0.44	0.45
industrial	0.27	0.39	0.26	0.36	0.27	0.37
j-idol	0.45	0.43	0.46	0.69	0.46	0.53
spanish	0.28	0.31	0.30	0.22	0.29	0.25
sleep	0.83	0.87	0.72	0.75	0.77	0.81
techno	0.54	0.64	0.43	0.49	0.48	0.55
mandopop	0.34	0.34	0.54	0.67	0.42	0.45
afrobeat	0.28	0.38	0.24	0.23	0.26	0.29
disney	0.50	0.59	0.47	0.50	0.48	0.54
study	0.73	0.77	0.78	0.86	0.75	0.81
indian	0.31	0.48	0.25	0.31	0.28	0.38
idm	0.36	0.56	0.31	0.27	0.34	0.36
forro	0.57	0.57	0.57	0.71	0.57	0.63
chicago-house	0.56	0.53	0.57	0.68	0.57	0.60
brazil	0.30	0.46	0.31	0.29	0.30	0.35
breakbeat	0.35	0.39	0.25	0.27	0.29	0.32

It's interesting to note that for the genres **sleep** and **study**, relatively high metrics are observed. This could imply that the data available in the dataset are particularly significant for predicting these two classes and that the two algorithms used perform quite well in predicting them.

4 Pattern Mining

4.1 Data preprocessing

Before performing the analysis it was necessary to manage the data. To do this we dropped useless variables such as name, album, artists, feature_duration_ms, time_signature and n_bars. We then transformed mode feature from [0,1] to [minor,major] and mapped to each number of key his respective as string. In the end to transform the continuous variables we used the binning technique and converted them into bins of different intervals.

4.2 Frequent Pattern Extraction

We performed frequent patter extraction using the **apriori** and **FP-growth** algorithms. We extracted the frequent itemsets, with a support $>=$ 15 and of minimum 3 items per set. The apriori and FP-growth algorithms mined the same itemsets, with 49 unique values and with support ranging from 15 to 24.7.

4.2.1 Closed itemsets

We then proceeded to extract closed itemsets, basically itemsets that have no frequent supersets with the same frequency. To do that we used the same support and number of items per set.

- with **apriori** algorithm: the resulting dataset contains 49 elements and these elements are the same of the frequent dataset;
- with **FP-growth** algorithm: also in this case no differences were spotted.

4.2.2 Maximal itemsets

To extract the maximal dataset we changed the support to 20 but keep the same the number of items per set.

- with **apriori** algorithm: the resulting dataset contains 11 elements;
- with **FP-growth** agorithm: the resulting dataset contains the same 11 elements of the apriori resulting dataset.

To find the optimal support we relied on Figure 35 and tried many values to see the ones that fit the most in our dataset. The plot shows that the best support should be around 6 but, with our amount of data, such support would not be descriptive. Also we examined the support 36 to obtain itemsets with a particular item, in this case 'major' or 'minor' mode.

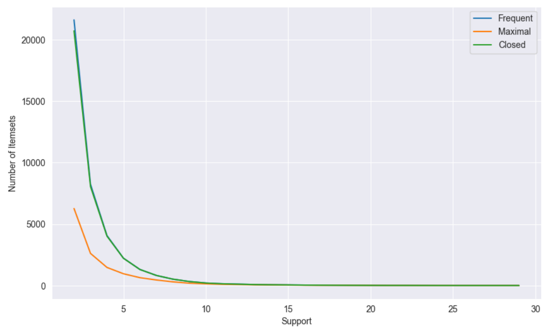


Figure 35: Apriori support vs itemsets plot

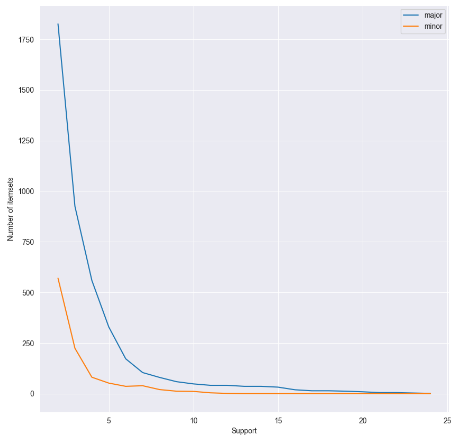


Figure 36: Support for mode

4.3 Rules extraction

We performed rules extraction with both apriori and FP-growth algoritms.

4.3.1 Rules Extraction with apriori

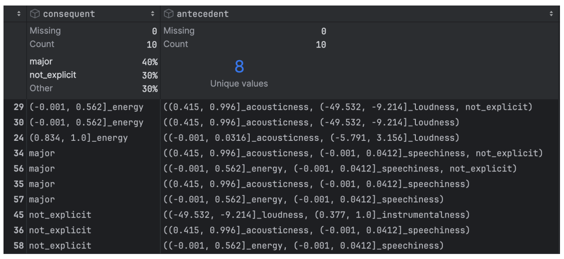
We applied apriori with 10 minimum support and a minimum of 3 items per set. Discovering 246 rules.

4.3.2 Rules Extraction with FP-growth

With FP-growth, with the same parameters as apriori, we came up with a dataset of only 17 rules, all of them with 'not_explicit' as consequent and with lower lift than apriori.

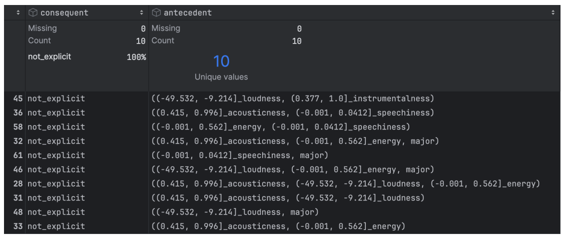
4.3.3 Quantitative and qualitative analysis

The rules with the highest lift and confidence are:



	consequent	antecedent
	Missing 0	Missing 0
	Count 10	Count 10
	major 40% not_explicit 30% Other 30%	8 Unique values
29	(-0.001, 0.562]_energy	((0.415, 0.996]_acousticness, (-49.532, -9.214]_loudness, not_explicit)
30	(-0.001, 0.562]_energy	((0.415, 0.996]_acousticness, (-49.532, -9.214]_loudness)
24	(0.834, 1.0]_energy	((-0.001, 0.0316]_acousticness, (-5.791, 3.156]_loudness)
34	major	((0.415, 0.996]_acousticness, (-0.001, 0.0412]_speechiness, not_explicit)
56	major	((-0.001, 0.562]_energy, (-0.001, 0.0412]_speechiness, not_explicit)
35	major	((0.415, 0.996]_acousticness, (-0.001, 0.0412]_speechiness)
57	major	((-0.001, 0.562]_energy, (-0.001, 0.0412]_speechiness)
45	not_explicit	((-49.532, -9.214]_loudness, (0.377, 1.0]_instrumentalness)
36	not_explicit	((0.415, 0.996]_acousticness, (-0.001, 0.0412]_speechiness)
58	not_explicit	((-0.001, 0.562]_energy, (-0.001, 0.0412]_speechiness)

Figure 37: Rules with highest lift



	consequent	antecedent
	Missing 0	Missing 0
	Count 10	Count 10
	not_explicit 100%	10 Unique values
45	not_explicit	((-49.532, -9.214]_loudness, (0.377, 1.0]_instrumentalness)
36	not_explicit	((0.415, 0.996]_acousticness, (-0.001, 0.0412]_speechiness)
58	not_explicit	((-0.001, 0.562]_energy, (-0.001, 0.0412]_speechiness)
32	not_explicit	((0.415, 0.996]_acousticness, (-0.001, 0.562]_energy, major)
61	not_explicit	((-0.001, 0.0412]_speechiness, major)
46	not_explicit	((-49.532, -9.214]_loudness, (-0.001, 0.562]_energy, major)
28	not_explicit	((0.415, 0.996]_acousticness, (-49.532, -9.214]_loudness, (-0.001, 0.562]_energy)
31	not_explicit	((0.415, 0.996]_acousticness, (-49.532, -9.214]_loudness)
48	not_explicit	((-49.532, -9.214]_loudness, major)
33	not_explicit	((0.415, 0.996]_acousticness, (-0.001, 0.562]_energy)

Figure 38: Rules with highest confidence

	abs_support	%_support	confidence	lift
count	62.000000	62.000000	62.000000	62.000000
mean	2584.096774	17.227312	0.929457	1.097338
std	464.845284	3.098969	0.053060	0.350659
min	1865.000000	12.433333	0.801274	0.876593
25%	2290.000000	15.266667	0.903144	0.992551
50%	2489.000000	16.593333	0.938078	1.013076
75%	2919.750000	19.465000	0.969618	1.053253
max	3706.000000	24.706667	0.994275	2.682481

Table 1: Descriptive Statistics

We can see in the table 1 that the absolute support, with an average of 2584.10 and a standard deviation of 464.85, reflects the frequency of certain data combinations, this might highlight diverse musical genres and tastes of listeners by showing off how often certain combination occurs. The relative support average of 17.23% further affirms that the rules found represents the occurrence of these combinations in the overall dataset. A high average confidence level of 92.9% and an average lift of 1.097 suggest that the association rules derived are not only common but also meaningful. The maximum lift of 2.682 indicates that the items in the dataset that are frequently found together, as indicated by the high lift value, are especially correlated.

4.3.4 Optimization of confidence and support for rules

To optimize the parameters for our association rule mining, we conducted a systematic analysis by varying the support and confidence thresholds. Utilizing a heatmap visualization, we were able to observe the impact of different threshold settings on the number of generated rules. The support threshold was incrementally adjusted from 1% to 30%, ensuring that the rules derived were significant enough to be considered frequent. Simultaneously, the confidence threshold was altered between 50% to 90% to ensure the reliability of the rules.

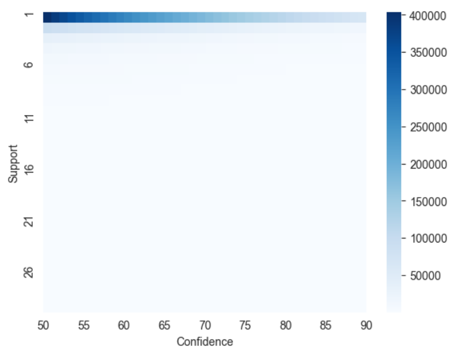


Figure 39: Heatmap of confidence and support

4.4 Target Variable Prediction

We have considered as a rule: consequent: *major* and antecedent: *((0.415, 0.996]_acousticness, (-0.001, 0.0412]_speechiness, not_explicit)* with a support of 12.78, a lift of 1.20 and a confidence of 0.80 from the Apriori rules set. Our initial approach involved a direct comparison in the test dataset of these antecedent conditions against corresponding columns in the dataset, with the expectation that a match would trigger the application of the rule's consequent as our prediction. In the end with 825 matches of antecedents conditions the rules proved its self right in 678 cases so the 82% of the total.

5 Regression

5.1 Univariate Regression

We tried to understand the relationship between the loudness of music tracks (independent variable) and their energy levels (dependent variable). We employed five different regression algorithms to model this relationship: Linear Regression, Ridge, Lasso, Decision Tree Regressor, and K-Nearest Neighbors Regressor. Let's analyze them in detail:

Model	MSE	MAE	R2	Coefficients	Intercept
Linear Regression	0.03	0.15	0.52	0.03173381	0.9385
Ridge Regression	0.03	0.15	0.52	0.03173376	0.9385
Lasso	0.06	0.21	0.12	0.00400781	0.6918
Decision Tree Regressor	0.03	0.13	0.57	N/A	
KNN Regressor	0.04	0.14	0.48	N/A	

Table 2: Model comparison

- **Linear and Ridge**: Both algorithms performed similarly, the R2 indicates that approximately 52% of the variance in energy can be explained by the loudness. The coefficients were nearly identical indicating a small positive relationship between the two variables;
- **Lasso Regression**: The Lasso model suggested a much weaker relationship between loudness and energy. The R2 dropped to 0.12 with an increase in MSE and MAE, this, as shown by the plots indicates a less accurate prediction compared to Linear and Ridge;
- **Decision Tree Regressor**: Improved our results with an R2 of 0.57. The errors were reduced to an MSE of 0.03 and an MAE of 0.13, suggesting better fit and prediction accuracy. We reached this result after applying gridsearchCV method and using as hyperparameters max_depth=5, min_samples_split=2, min_samples_leaf=101. This was the best algorithm for this regression task;

- **KNN Regressor**: Achieved an R2 of 0.48, close to Linear and Ridge Regression, and had the second lowest MAE of 0.14 among all the models, but with a slightly higher MSE of 0.04.

5.2 Multivariate Regression

We tried to predict values of 2 target variables 'danceability' and 'energy' training on independent variables 'valence','popularity','speechiness','acousticness','processing'.

Model	MSE	MAE	R2
Linear Regression	0.029	0.135	0.441
Ridge Regression	0.029	0.135	0.441
Lasso	0.055	0.189	0
Decision Tree Regressor	0.043	0.156	0.169
KNN Regressor	0.028	0.129	0.463

Table 3: Model comparison (multivariate)

We looked at the Pearson correlation heat map and choose those independent variables. As we can see Linear and Ridge regressor performs exactly the same. The MSE and MAE are all under the 1% so let's look just at the R2 score. KNN Regressor seems to be the best one followed by Linear Regressor and Ridge. Looking at the plots we can see that Lasso cannot draw a line that follows the shape of data, particularly for 'danceability' variable. For the first time we tried with just two independent variables in the train set, but the scores were really bad. R2 score was a negative number. So we increased the number of independent variables and we reached much better scores.