University of Pisa
Data Mining I
A.Y 2023-2024

Data Mining I Project

Submitted to:
Prof. Dino Pedreschi
Prof. Riccardo Guidotti

Submitted by:
Alessandro Carella
Sara Hoxha
Rafael Urbina

December 31, 2023

Contents

1 Data Understanding & Preparation

1.1 Data semantics

In this section, we have done a comprehensive analysis of the semantics pertaining to the attributes within the Spotify Tracks dataset. This analysis is depicted in Table 1.

Attribute	Type	Description
Name	Nominal	The track's name
Duration_ms	Continuous	The track's length in milliseconds
Explicit	Binary	Whether or not the track has explicit lyrics
Popularity	Ratio	The popularity of the artist
Artists	Nominal	The names of the track's artists
Album_name	Nominal	The name of the album
Danceability	Continuous	How suitable a track is for dancing based on a combination of tempo, rhythm stability, beat strength, and regularity
Energy	Continuous	Measure of intensity and activity where 0.0 is least danceable and 1.0 is most danceable
Key	Ordinal	The key the track is in
Loudness	Continuous	Overall loudness of a track in decibels
Mode	Binary	Modality (major or minor) of a track
Speechiness	Continuous	Presence of spoken words in a track, above 0.66 describe tracks made entirely of spoken words, between 0.33 and 0.66 describe tracks that may contain both music and speech, and below 0.33 most likely represent non-speech-like tracks.
Acousticness	Continuous	Overall accousticness of track.
Instrumentalness	Continuous	Whether a track contains no vocals, where "Ooh" and "aah" sounds are treated as instrumental and spoken word tracks are "vocal"
Liveness	Continuous	The presence of an audience in the recording
Valence	Continuous	The musical emotion in two main categories, where high valence means that the song is happy, vibrant etc, and lower values mean that the song is sad, depressing, etc
Tempo	Continuous	The BPM tempo of the song, it derives from the average duration so it's not consistent for a track with variations in tempo, also musically the measurements tend to be discrete while this variable is continuous.
Features_duration_ms	Continuous	The duration of each song in milliseconds
Time_signature	Ordinal	The time signature in a reduced way, so from "3/4" to "7/4" values from 3 to 7. This metric doesn't allow the full scope representation of the time signatures and can be used just as a proxy of time structure in songs.
N_beats	Discrete	The number of beats the song has. This value is subjective to the time signature of the song and cannot be analysed as the true length of the song, it should be analysed in conjunction with the time_signature.
N_bars	Discrete	The number of bars the song has. This variable should be analysed taking into consideration the time_signature variable, as they are directly dependable.
Popularity_confidence	Discrete	Level of confidence of popularity of the song.
Genre	Nominal	Musical genre of the song
Processing	Discrete	There is no information about the variable but there is a relation with the variable "key" and also although the variable is of float type, it only has 12 unique values

Table 1: *Data Description*

1.2 Distribution of the variables and statistics

We examined fundamental statistical properties of the dataset, focusing on the measurement of central tendency and dispersion of attributes. These properties are shown in Table 2 and Table 3.

The dataset seems to be well-populated in most columns. However, the attributes **Processing**, **Mode** and **Popularity_confidence** show significant number of rows with missing values.

Another important consideration is the difference in mean and median found in attributes like **Duration_ms**, **N_beats**, and **N_bars**. This offset between mean, and median could imply skewness in the distribution of the attributes, which we will further investigate using histograms in the next subsection.

1.2.1 Measurement of Central Tendency

Attribute	**Mean**	**Mode**	**Median**
Duration_ms	246807.480133	180000.0	227826
Popularity	27.423667	0.0	24
Danceability	0.551063	0.0	0.58
Energy	0.656231	0.961	0.709
Key	5.287867	7	5
Valence	0.436853	0	0.416
Tempo	123.116544	0	124.188
Features_duration _ms	246794.6835	180000	227818.5
Time_signature	3.876179	4	4
N_beats	501.862333	0	461
N_bars	128.39340	97	117
Popularity_confidence	0.490479	0.083,0.211,0.707	0.48

Table 2: *Central Tendency of Data*

In Figure 1 we can observe that some attributes, such as **Danceability**, exhibit well-behaved distributions close to a normal distribution and are unimodal. Other attributes, such as **Popularity**, and **Energy**, display different distribution shapes. **Popularity** is non-symmetric, with a mild concentration of points on the left side of the distributions and a mild skewness to the right. **Energy** has a long tail and a high concentration of points close to 1. One of the attributes with the highest skewness in the plots is **Duration_ms**. It has a wide range and is skewed to the right. The presence of extreme outliers can be inferred from the histograms, as a large portion of the plots remain almost empty. This suggests that there are likely only a few data points in the extreme sections of the x-axis.

1.2.2 Measurement of Data Dispersion

In Table 3 it's possible to observe numerically the dispersion of the data by looking at the Standard deviation, Variance, and the min-max values. These affirm the dispersion seen in Figure 1.

Time_signature is an attribute where all 25th, 50th, and 75th percentiles of the data are identified with a value of 4. This, combined with the categorical nature of the attribute, suggests that the variable predominantly concentrates in the value of 4.

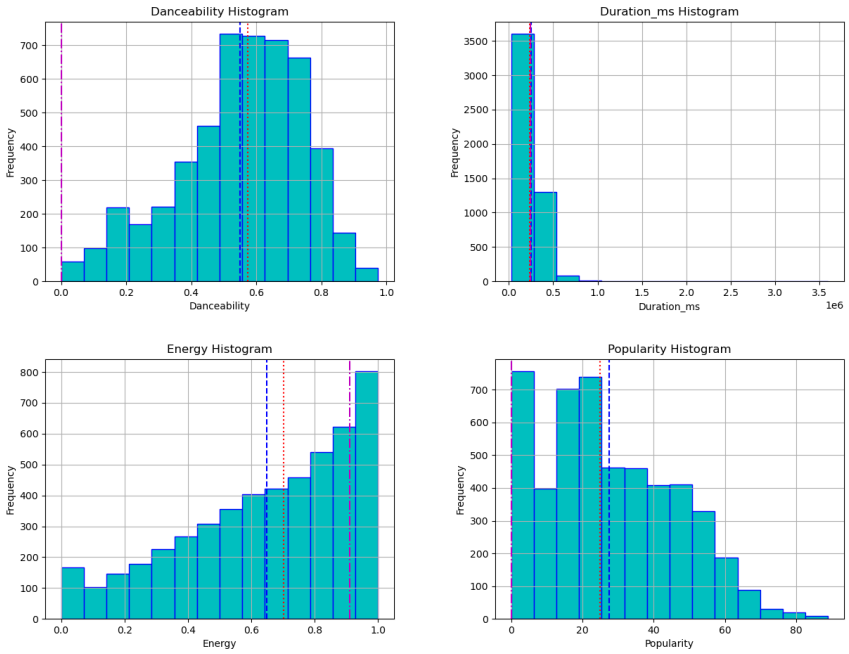


Figure 1: Matrix of most relevant attributes histograms

Attribute	**Min**	**25%; 50%; 75%**	**Max**	**Range**	**Variance**	**Std**
Duration_ms	8586	180000; 227826; 288903	4120258	4111672	1.638×10^{10}	127994.05
Popularity	0	14; 24; 42	94	94	345.49	18.58
Danceability	0	0.441; 0.58; 0.69	0.98	0.98	0.03	0.19
Energy	0	0.48; 0.70; 0.884	1	1	0.06	0.26
Key	0	2; 5; 8	11	11	12.85	3.58
Valence	0	0.19; 0.41; 0.66	0.99	0.99	0.07	0.27
Tempo	0	99.93; 124.18; 141.98	220.52	220.52	1019.58	31.93
Features_duration _ms	8587	180000; 227818.5; 288903	4120258	3940258	1.63×10^{10}	127984.99
Time_signature	0	4; 4; 4	5	**5**	0.32	0.56

N_beats	0	327; 461; 625	7348	7348	78786.76	280.68
N_bars	0	83; 117; 159	2170	2170	5642.09	75.11
Popularity_confidence	0	0.23; 0.48; 0.73	1	1	0.08	0.29

Table 3: *Dispersion of Data*

1.3 Assessing data quality

1.3.1 Errors

During the project's data evaluation for errors, we identified a single issue with the **Time_signature** attribute. The attribute is defined as categorical showing the time signature of the song, i.e.: 3/4 = 3; in the official Spotify definition we find the range to be from 3 to 7. However, the actual values in the dataset range from 0 to 5. This discrepancy made us question how the real values align with the defined categories, especially since both the Spotify definition and the dataset exhibit 5 categories. The absence of the number 2 in the dataset's categories raises further suspicion.

1.3.2 Outliers

Regarding outliers, we conducted a visual and statistical analysis on the attributes, with the help of box plots, the Z-score and the Inter-quartile range. We initially examined outlier values for each attribute independently, testing various Z-score ranges (2 to 5) and corresponding scalar IQR values (1.5, 3.0, 4.5, and 7.0).

This approach revealed that the majority of attributes exhibit data points beyond the ($1.5 \times IQR$) threshold, but exceptions include **Acousticness**, **Instrumentalness**, and **Valence**. However, it's important to note that these attributes have a range of 0.0 to 1.0, contributing to a lower occurrence of extreme outlier points.

After this initial approach, we explored adding an additional criterion for handling outliers. Given our dataset's extensive attributes (24 initially), removing observations with a single outlier attribute would lead to a substantial loss of data. Thus, we adopted a criterion where an observation requires two or more attributes to be classified as outliers to be labelled as a potential outlier. The final metric employed for this criterion is the IQR with a scale of 1.5, resulting in the following formula for defining outliers:

> If two or more attributes for each observation fall outside of this Lower or Upper bound, the observation is considered an outlier.
>
> Inter-quartile range: $IQR = Q_3 - Q_1$
> Lower bound of the criteria: $L = Q_1 - 1.5 \times \text{IQR}$
> Upper bound of the criteria: $U = Q_3 + 1.5 \times \text{IQR}$

As such, we classify observations: 10,913 instances with no outliers, 3,324 with one outlier, 602 with two outliers, 141 with three outliers, 19 with four outliers, 6 with five outliers, and

1 with six outliers. Based on these results, at least 66% of the observations show no outliers in any attribute. All the observations with more than 2 attributes classified as outliers were removed from the dataset. This allowed us to clean extreme observations while maintaining a dataset size close to the original. In fact, the final dataset consists of 14237 rows, rather than the initial 15000 rows.

Figure 2 illustrates the difference between the original version of **n_beats** attribute and the cleaned version. In conclusion, this outlier removal approach effectively addresses extreme values while minimizing information loss.

1.3.3 Missing values

The columns with missing values in the original dataset are three: **time_signature**, **mode**, **popularity_confidence**. The presence of missing data poses a significant challenge in data analysis. Incomplete datasets can lead to biased results and hinder the ability to extract meaningful insights from the data. To address this challenge, we explored various imputation methods to replace missing values with realistic estimates.

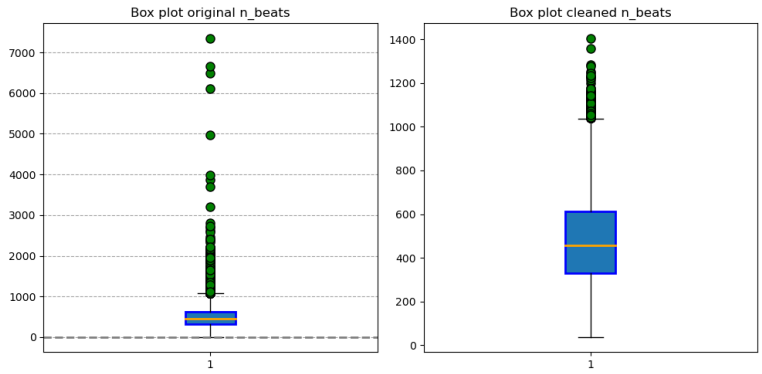


Figure 2: Box plot of N_beats original and cleaned data

Our initial focus was on filling methods that did not yield good results for our scope such as: Hot-deck imputation, classification methods and regression. While these methods appeared promising, we opted against using them since Hot-deck imputation filling has limitations when dealing with largely spread data and classification is a separate section of the project and fell outside the scope of the current section.

After careful consideration, we decided to employ mean and mode replacement methods due to their simplicity and reliability. These methods, though basic, were deemed suitable for handling the missing data in our dataset.

For what regards `time_signature` and `mode`, we decided to fill the missing values with the mode value for the column, where this translated to using the values 4 and 0, respectively. For what regards `popularity_confidence`, we decided to delete the column since there are too many missing values, in fact 12783 samples have this feature value missing. Furthermore, the `popularity` feature seems to be an alternative for `popularity_confidence` values.

1.3.4 Semantic Inconsistencies

In regard to semantic inconsistencies, which means any strange or incorrect values due to human mistake in the dataset, we looked into two cases: albums with similar name from the same artists, as well albums with the same name from similar artists. For the first case, we set a rule that if the album names were 80% similar, we'd flag them as potential inconsistencies. After our research, there were a few cases where albums by the same artists had the exact same name, except for some small changes like one letter being uppercase or lowercase: "Voz d'Amor" and "Voz D'Amor". Then, we made sure the inconsistent values were replaced with the correct album names.

For the second case, we used the same threshold so that if the artist names were 80% similar, we'd flag them as potential inconsistencies. However, we didn't find any such instances.

1.4 Variable transformations

In the initial stages of data analysis, we focused on variable transformations, specifically exploring log, x^2, and square root transformations to refine raw data for meaningful insights. After weighing the advantages and disadvantages of each transformation, we considered using log transformation and square root transformation to improve visualization readability and mitigate outliers. However, in the current phase of preparing data for clustering analysis, these transformations were intentionally omitted due to prior normalization efforts aimed at maintaining equitable variable contribution. While acknowledging the benefits of variable transformations in specific contexts, our immediate priority is preserving normalized data integrity for clustering insights. Looking ahead to regression and classification analyses, we plan to reassess the potential of log and square root transformations to extract meaningful patterns. As a concluding note, we highlight improved attributes (**Speechiness**) through transformation, showcasing enhanced distribution shapes and reduced outlier presence as we can see in figure 3.

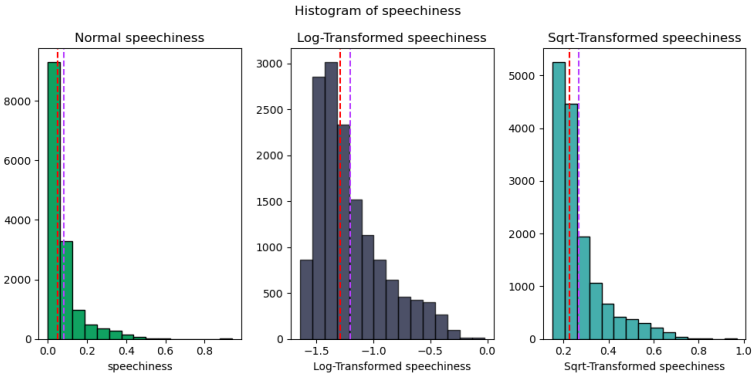


Figure 3: Speechiness with variable transformations

1.5 Pairwise correlations and elimination of variables

To have a comprehensive and visual analysis of the inter attribute correlations in the dataset, we employed the clustermap method depicted in Figure 4. Our analysis yielded the following observations:

1.The attributes **duration_ms** and **features_ duration_ ms** show a perfect positive linear relationship with a correlation coefficient of 1, indicating they are nearly identical, as a correlation of 1 is expected when correlating a variable with itself. This strong correlation is visually evident in the heatmap's prominent red colour.

2.Similarly, a strong positive linear correlation (correlation coefficient: 0.98) is evident between the attributes **n_beats** and **n_bars**. This association is visually represented by the intensified red colour in the heatmap.

Beyond correlations, our analysis revealed these key insights on specific dataset attributes:

1.The attribute **popularity_confidence** has 12783

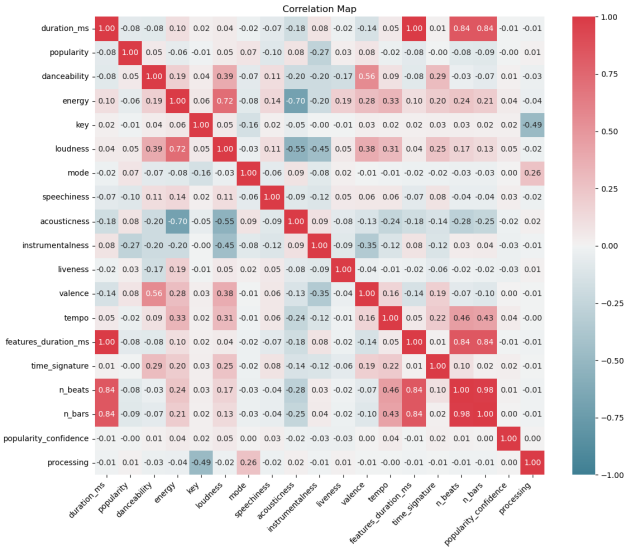


Figure 4: Correlation matrix before feature selection

missing values, raising a significant data integrity concern.

2. For **processing** attribute, we found no substantial information in the dataset and its absence in the Spotify API documentation raises uncertainties about its relevance.

3. The **processing** attribute, despite observing no direct correlation with the **key** attribute through the clustermap, consistently mirrored (with different but consistent values) the values found in **key**.

Based on these findings, we decided to remove these specific attributes: `features_duration_ms`, `n_bars`, `popularity_confidence`, and `processing`, thus achieving a more standardized dataset.

2 Clustering

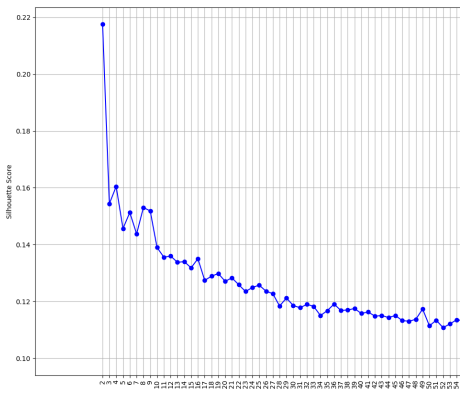
2.1 Analysis by kMeans and alike clustering

2.1.1 kMeans

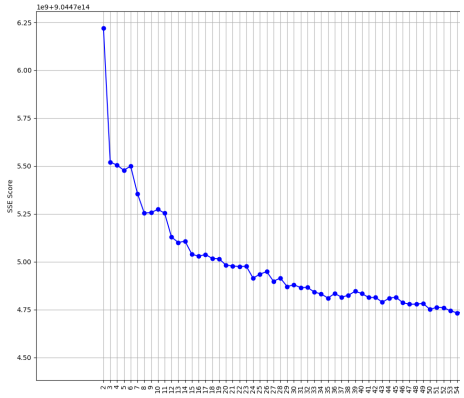
For kMeans, we explored a range of k values, ranging from 2 to 200, to ascertain the optimal clustering parameter. For this, we conducted an analysis involving both the Sum of Squared Errors (SSE) and Silhouette scores for each k value.

In Figure 5(a), it is evident that the Silhouette score remains consistently low, even when the cluster number is as low as 2, registering a Silhouette score of **0.22**. Furthermore, the Silhouette score tends to converge to 0 as it stabilizes around k equals to 30.

On the other hand, as illustrated in Figure 5(b), the SSE steadily decreases until k is approximately 50 and subsequently stabilizes at an approximate value of **4.6**. To have a reasonable balance between the two metrics, we have chosen a value of $k = 11$, where the sum of squared error (SSE) is ˜ 5.1 and the Silhouette score is ˜ 0.13.



((a)) Silhouette score plot on different k values



((b)) SSE plot on different k values

Figure 5: Silhouette score and SSE plots on different k values

2.1.2 Bisecting kMeans

Just like in kMeans, for Bisecting kMeans we iterated over a range of k values from 2 to 200, to ascertain the optimal number of clusters. As before, both the Sum of Squared Errors (SSE) and Silhouette scores were calculated.

The obtained results are quite similar to the kMeans algorithm, with the best Silhouette score value of **0.225**, using a k equal to 2 and a stabilization of the score around a value of k ˜ 15.

Same goes for the SSE where it decreases until a value of $k = 36$ and stabilizes around a value of **4.8**. We think that the mostly optimal k value for this algorithm, to have a balance between the two metrics, is $k = 10$, where the sum of squared errors is ˜ 5.3 and the silhouette value is ˜ 0.11.

2.1.3 kModes

For the K-modes algorithm, we conducted an iterative exploration across a range of K values to ascertain the optimal choice for clustering our dataset. In this evaluation, we employed the *Entropy* metric as a criterion for determining the most effective K value. Our findings revealed that the optimal clustering was achieved when $K = 2$, resulting in an entropy value of approximately **0.0102**. In Figure 6, it is clear that as K increases, the entropy value also rises. This increase is undesirable in our context, as we aim for the lowest possible entropy value.

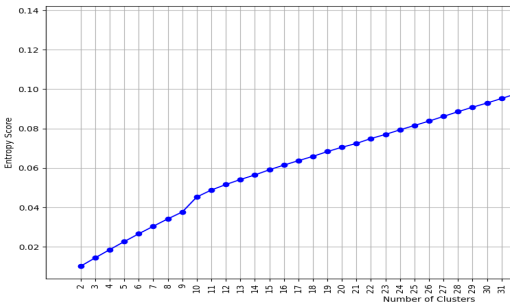


Figure 6: Entropy scores for kModes

2.2 Analysis by hierarchical clustering

In our hierarchical clustering analysis, we systematically investigated clustering outcomes using various linkage methods, namely *Centroid* Linkage, *Complete* Linkage, *Single* Linkage, and *Group Average* Linkage. The exploration involved employing threshold values ranging from 5 to 50 and considering two distinct criteria: *distance* and *maxclust*. Upon calculating Silhouette scores for each configuration, we identified that the most optimal result was applying Single Linkage with a threshold of 5 and the maxclust criterion, yielding a silhouette score of approximately 0.34. Despite this favourable outcome, we exercised caution in adopting Single Linkage due to the possible presence of outliers and noise in our dataset, which might compromise the robustness of the clustering method. Our attention turned to the second-highest silhouette score of approximately 0.268, achieved with Group Average Linkage with a threshold of 5 and the maxclust criterion, but its dendrogram resulted in two unbalanced clusters, which was undesirable. Lastly, we decided the most suitable to be the third-highest score of 0.11 obtained with Group Average Linkage with a threshold of 10 and the maxclust criterion. This Group Average method strikes a balance between Complete and Single Linkage. Moreover, recognizing that Group Average Linkage is less sensitive to noise and outliers, we deemed it more suitable for our dataset. In Figure 7 one can observe the dendrogram corresponding to this group average hierarchical clustering.

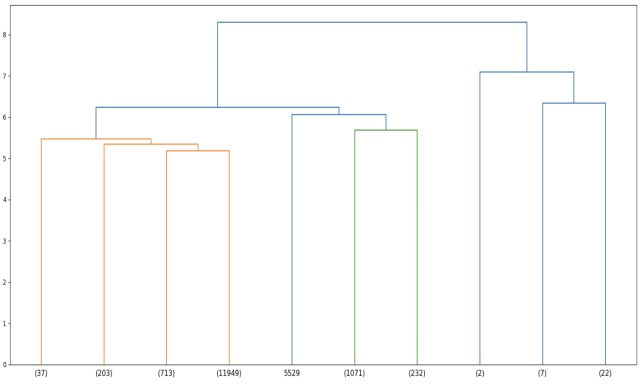


Figure 7: Group Average Linkage with threshold 10 & maxclust

2.3 Analysis by density based clusterings

For density-based clustering, we used DBSCAN, HDBSCAN, and OPTICS algorithms, ensuring consistency by employing the same StandardScaler throughout.

2.3.1 DBSCAN

We ran the DBSCAN algorithm, systematically varying the *eps* across [0.1, 0.5, 1, 1.5, 2, 2.5, 3] and *min_samples* in the range 1 to 20. After running the algorithm, we computed the Silhouette score. Promising results were obtained with an *eps* value of 3 and a *min_samples* of 5 or more, as shown in Figure 8. For *eps* values exceeding 3 and a constant *min_samples*, it showed favourable results when most samples were positioned within two clusters.

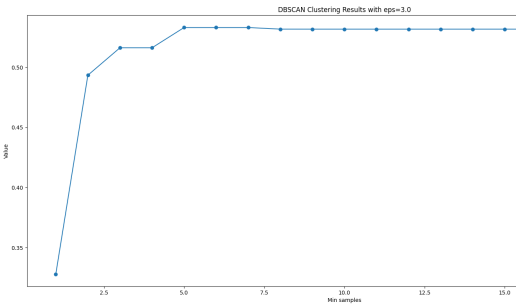


Figure 8: DBSCAN silhouette plot

2.3.2 OPTICS

For OPTICS algorithm, we utilized minimum samples in the range of 1 to 20, the damping factor (Xi) across values [0.01, 0.05, 0.1], and minimum cluster size across values [0.01, 0.05, 0.1]. As OPTICS is quite computationally expensive, we decided to reduce the sample size to 5000, as it was still deemed enough to give us a clear picture of the data behaviour. After assessing the results using Silhouette scores, promising outcomes were limited, except when considering 2 clusters and Xi of 0.05 or 0.1. However, even with the most favourable Silhouette score of 0.57, the outcome was still considered suboptimal.

2.3.3 HDBSCAN

In HDBSCAN, we set ‘min_cluster_size‘ from 2 to 50 and ‘min_samples‘ from 2 to 100. The clustering results followed similar trends as observed in other density-based algorithms, with the optimal Silhouette score consistently achieved when the number of clusters was set to 2.

2.4 Conclusion

In conclusion, after evaluating various clustering algorithms, Hierarchical Group Average Linkage with a **10** threshold and **maxclust** criterion emerged as the optimal choice for the dataset, achieving a Silhouette score of 0.11, and leading to a larger number of more balanced cluster than the other algorithms.

3 Classification

3.1 Encoded Genres

For the classification models the aim of our study is to correctly identify the genre of each song in the test set, this approach will have a multiclass target variable with 20 possible classes, which are the possible genre's currently in the data set. The repercussion of this approach is that some of the models can have problems with high multiclass target variables.

Since this issue could arise with the models, we decided to split the work in two: at first, we considered as targets all the genres, then we took as targets a variable of clustered genres. These clusters of genres were generated by the Kmeans model specified in our clustering section (with a k value of 5). We can see the results of this clustering in Figure 9.

With this approach we came up with this grouping setting:
Group0: afrobeat, bluegrass, brazil, forro, indian, industrial, j-dance, mandopop, spanish
Group1: j-idol
Group2: black-metal, breakbeat, chicago-house, happy, iranian, techno
Group3: disney, idm, sleep, study

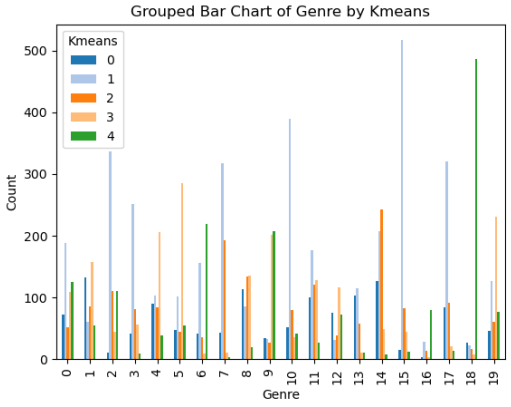


Figure 9: Barchart of clustered genres [1]

3.2 K-nearest neighbours classifier on genres

3.2.1 Implementation

For the implementation of the K-nearest neighbours classifier we decided to implement a mix of various techniques, such as: varying sample sizes (10%, 20%, 30%, ..., 90%, 100% of the dataset), k fold cross validation (with a k value of 33), a range of weights (uniform and distance), a range of k, determined by finding a good approximation for the value of k (square root of the sample size) and using a range of values near it, composed by the first 10 odd numbers preceding and succeeding the previously determined approximation.

3.2.2 Results on train data

Once all the models have been created by the iterative process, we find the accuracy, precision, recall and f1 scores of all of them and make a comparison between the mean value of each k-fold and the other mix of parameters to determine the best model to use on the test dataset. The models were compared based on the dataset sample size resulting in the learning curve represented in Figure 10.

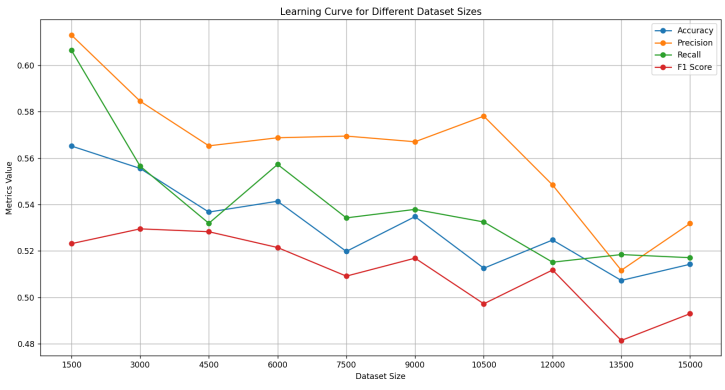


Figure 10: Learning curve K Nearest Neighbours Classifier: Target genre

In image 10, we can observe that the highest score results were obtained with a sample size of 1500,

[1]The mapped genres are: 'j-idol':14, 'afrobeat':0, 'spanish':17, 'study':18, 'breakbeat':4, 'forro':7, 'idm':9, 'mandopop':15, 'chicago-house':5, 'bluegrass':2,'black-metal':1, 'brazil':3, 'iranian':12, 'happy':8,'indian':10, 'j-dance':13, 'industrial':11,'techno':19,'disney':6, 'sleep':16

or in other words 10% of the full
dataset. Those results use a k value of 35 and a uniform weight. The scores obtained by this model on the test dataset were: Accuracy of 56% Precision of 61% Recall of 60% F1 score of 52%

3.2.3 Results on test data

After determining the best model, we trained it using the downsampled dataset of size 1500 as a whole (without the k fold cross validation split) and we processed the same metrics as before on the provided test dataset (scaled with the same scaler used for the 1500 samples dataset during the first part) and we found the following metrics: Accuracy of 41% Precision of 44% Recall 41% F1 score 39%

Even if the results are better than expected for a 20 multiclass classification problem we were not satisfied with the obtained results. Therefore, our group decided to find clusters of the available genres and to use this new column of the dataset as a target variable.

3.3 K-nearest neighbours classifier on clustered genres

3.3.1 Implementation

Since the classification algorithm is identical to the one previously used, the process for determining the best model using the new feature as target kept the same parameters and iterations expect for the target values.

3.3.2 Results on train data

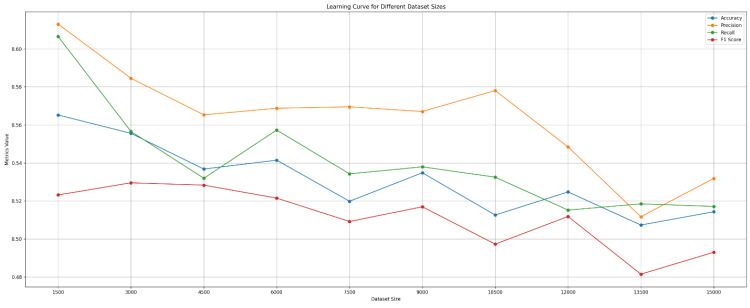


Figure 11: Learning curve K Nearest Neighbours Classifier: Target **grouped_genres**

In Figure 11 one can see that the learning curve is somewhat similar to the previous with a descending trend toward bigger samples dataset. In fact, the best model is only using 10% of the total sample size, as the previous model.

As previously stated, we kept the size of the dataset used at 1500 samples and the best results were obtained with the same k value of the K-nearest neighbours classifier on genre, which was 35 and a uniform weight. The metrics obtained by this model on the training data are: Accuracy of 56% Precision of 61% Recall of 60% F1 score of 52%

3.3.3 Results on test data

We repeated the same process as before and obtained this, surprising, scores: Accuracy of 74% Precision of 74% Recall of 67% F1 score of 73%

3.4 Naive Bayes Classifier

We applied the Naive Bayes classifier to categorize songs into genres using the test set. Target labels are the song genres, with similar results for individual and grouped genres, indicating model robustness. For Naive Bayes, handling multiple classes is challenging, but reducing from 20 genres to 4 alleviates this issue.

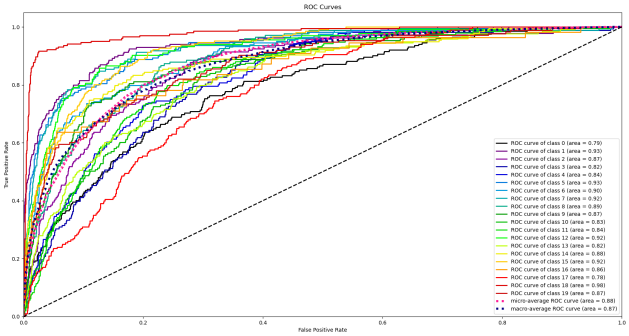


Figure 12: ROC Curve for Gaussian NB: Target genre

We evaluated two models: **Gaussian NB** for **continuous attributes** and **Multinomial NB** for **categorical attributes**. Gaussian NB outperformed Multinomial NB due to better results and more applicable continuous attributes.

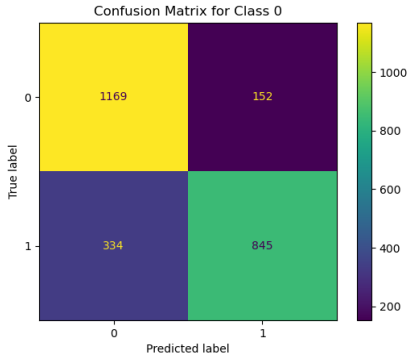


Figure 13: Confusion matrix for Grouped NB

Results for Gaussian NB ROC curve are depicted in Figure 12. Additionally, the confusion matrix for grouped genres is presented in figure 13.

The AUC for most genres is high, possibly due to numerous true negatives in the 20-class target. Grouped and non-grouped Gaussian models exhibit similar total accuracy (**0.88** and **0.87**, respectively). Metrics like precision, recall, and f1-score were assessed for further clarity:

Grouped Gaussian: Precision = 0.64, Recall = 0.70, F1-score = 0.75

Non-grouped Gaussian: Precision = 0.39, Recall = 0.37, F1-score = 0.34

Considering the class count difference (4 vs 20), these results are impressive, emphasizing the need to interpret metrics cautiously, especially with varied class sizes.

3.5 Decision Set Trees

3.5.1 Multiclass Classification

Training with no hyperparameter tuning

We used genre for multiclass classification by categorizing it into 20 classes. Initially, we trained the model with 17 variables without any hyperparameter tuning. Despite achieving 0.97 accuracy on the training set, the test set accuracy dropped to 43%. See Figure 15 for the classification report on the test set.

	precision	recall	f1-score	support
0	0.79	0.80	0.80	2250
1	0.75	0.70	0.73	500
2	0.69	0.71	0.70	1500
3	0.54	0.53	0.53	750
accuracy			0.72	5000
macro avg	0.70	0.69	0.69	5000
weighted avg	0.72	0.72	0.72	5000

Figure 14: Classification Report Grouped_Genres

Dissatisfied with prior results, we targeted **grouped_genres** for classification using the same approach. We attained 98% training accuracy and 72% test accuracy, a notable improvement

over the previous genre model. However, both models exhibited overfitting, indicating suboptimal choices. See Figure 14 for the classification report on the test set for **grouped_genres**.

Randomized Search Cross Validation

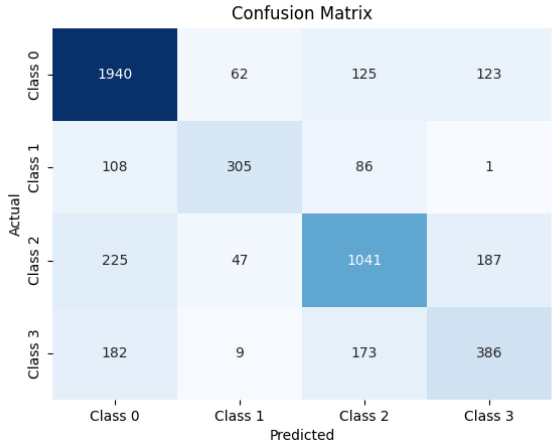
	precision	recall	f1-score	support
afrobeat	0.23	0.21	0.22	250
black-metal	0.62	0.65	0.64	250
bluegrass	0.45	0.42	0.43	250
brazil	0.29	0.30	0.30	250
breakbeat	0.29	0.32	0.30	250
chicago-house	0.55	0.57	0.56	250
disney	0.46	0.48	0.47	250
forro	0.51	0.54	0.53	250
happy	0.39	0.37	0.38	250
idm	0.32	0.32	0.32	250
indian	0.30	0.32	0.31	250
industrial	0.28	0.27	0.28	250
iranian	0.52	0.52	0.52	250
j-dance	0.43	0.37	0.40	250
j-idol	0.46	0.44	0.45	250
mandopop	0.34	0.39	0.36	250
sleep	0.74	0.71	0.73	250
spanish	0.22	0.20	0.21	250
study	0.77	0.74	0.75	250
techno	0.40	0.41	0.40	250
accuracy			0.43	5000
macro avg	0.43	0.43	0.43	5000
weighted avg	0.43	0.43	0.43	5000

Figure 15: Classification Report Genre

We used these hyperparameters for randomized search cross-validation: `min_ samples_ split` [0.002, 0.01, 0.05, 0.1, 0.2], `min_ samples_ leaf` [0.01, 0.05, 0.1, 0.2, 1, 2], `max_ depth` [2, 3, 4, 5, 6, 7, 8, None], `criterion` ["gini", "entropy"], `ccp_ alpha` [0.0, 0.01,..., 0.1].

Two target variables were tested: **genre** with train accuracy 0.48, test accuracy 0.45, and **grouped_genres** with train accuracy 0.74, test accuracy 0.73. Confusion matrix in Figure 16(a) for **grouped_genres** shows correct predictions for 3672 out of 5000 songs. Figure 16(b) displays the classification report for test set predictions of **grouped_genres**.

Undoubtedly, the **grouped_genres** model outperformed the **genre** model, both in randomized model and without hyperparameter tuning, make it our optimal one for Decision Trees.



((a)) Confusion Matrix : Random Model Grouped_Genres Target

	precision	recall	f1-score	support
0	0.79	0.86	0.82	2250
1	0.72	0.61	0.66	500
2	0.73	0.69	0.71	1500
3	0.55	0.51	0.53	750
accuracy			0.73	5000
macro avg	0.70	0.67	0.68	5000
weighted avg	0.73	0.73	0.73	5000

((b)) Classification Report : Random Model Grouped_Genres Target

Grid Search Cross Validation

For grid search cross-validation, we used the same parameters of the randomized search to compare the outcomes between the two methods for the **grouped_genres**, whereas for **genre** we adjusted the values: **min_samples_split** up to 4 and **min_samples_leaf** up to 6. During the grid search, we observed that it consumed more time than the randomized approach, which is expected since grid search explores all possible parameter combinations. For the **genre** target, the optimal values for the parameters were `min_samples_split`: 4, `min_samples_leaf`:

6, `max_depth`: None, `criterion`: gini , and `ccp_alpha`: 0.01. This model yielded a 68% train and a 46% test accuracy, however due to the possible overfitting, as well as precaution of not creating a too simple model, we were not pleased with these results.

On the other hand, for the **grouped_genres** target, the optimal values for the parameters were `min_samples_split`: 0.05, `min_samples_leaf`: 0.1, `max_depth`: 2, `criterion`: entropy , and `ccp_alpha`: 0.01. This model yielded a 66% train and a 65% test accuracy and the tree is depicted in figure 17.

3.5.2 Binary Classification

Training with no hyperparameter tuning

For binary classification, we utilized **mode** as the target variable, which is categorized into two classes : 1.0 & 0.0. We trained the model without hyperparameter tuning utilizing 17 variables. For the train set, we obtained a near-perfect accuracy of 98%, however for the test set we got an accuracy of 66% which exhibits signs of overfitting.

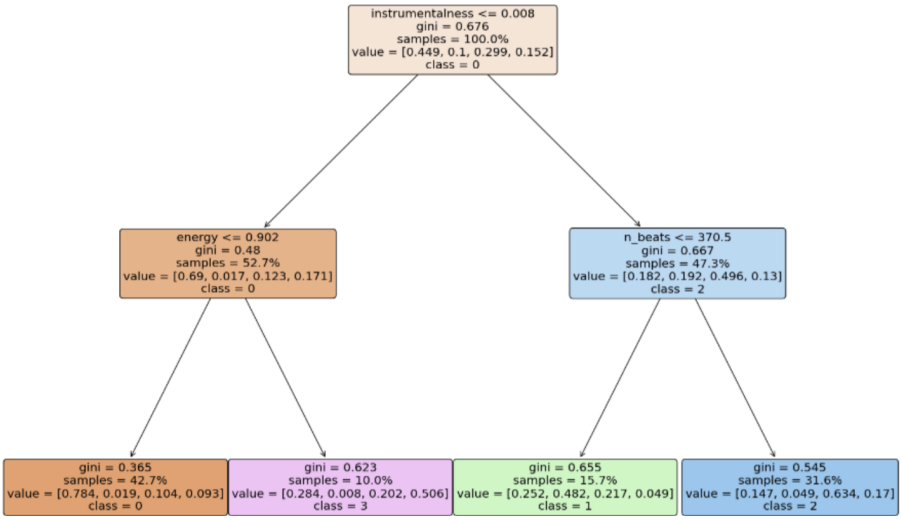


Figure 17: Decision Tree - Grid Search Grouped_Genres Target

See Figure 18 for classification report on test set.

	precision	recall	f1-score	support
0.0	0.35	0.36	0.35	1282
1.0	0.78	0.76	0.77	3718
accuracy			0.66	5000
macro avg	0.56	0.56	0.56	5000
weighted avg	0.67	0.66	0.66	5000

Figure 18: Classification Report Mode

Randomized Search Cross Validation

For randomized search cross-validation we used the same hyperparameters ranges as we did before in the multiclass classification, however we got very unsatisfactory results. According to the Randomized Search, the model obtained produced a more stable test and train accuracy both at 74%, but it was not able to correctly identify any instances of class 0. This can even be seen in the confusion matrix in the Figure 19. Even though the results varied slightly because they were random, we didn't see any change in terms of the recall score for class 0, which remained at 0.00. The reason for this poor result, was due to a bias towards the majority class prompted by the imbalanced train dataset, where there are 11111 instances of class 1, but only 3889 instances of class 0, causing a poor generalization on the minority class.

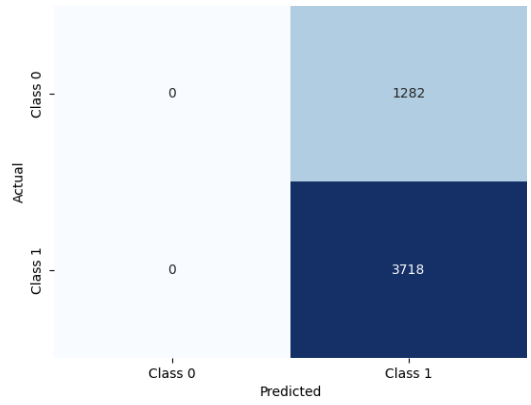


Figure 19: Confusion Matrix Mode

Grid Search Cross Validation

For grid search cross validation, we have decided to use the same parameters as we used in the randomized search to compare them. The model we got was same as the one from Randomized Search with 74% accuracy on both train and test sets, as well as a recall score of 0.0 for class 0. The parameters selected by the iterative approach were: `min_samples_split`: 0.002, `min_samples_leaf`: 0.01, `max_depth`: 2, `criterion`: entropy , and `ccp_alpha`: 0.0. The model produced such poor results for the same reason we indicated in the Randomized Search. As such, none of these models were seen as optimal.

3.6 Conclusion

In our analysis of classification models, a clear trend emerged: grouping the `genre` attribute consistently gave better results than using its original values. Given these considerations, we concluded that the optimal classification model is KNN, primarily for two compelling reasons. Firstly, KNN exhibited satisfactory performance metrics, boasting an accuracy of **74%**, precision of 74%, recall of 67%, and an F1 score of 73% on the test set. Furthermore, the K-nearest neighbors approach aligns with our human-like thought process when categorizing songs into genres, as this method calculates the distance between songs to determine their similarity, mirroring the way we naturally classify songs based on their similarities.

4 Regression

In the regression section, we aimed to predict a song's **duration**, **popularity**, and **danceability**. Employing various models, we discuss their implementations, conclusions, and compare results on test data in section 4.6.

4.1 Simple Regression

Implementation & Conclussion

As simple regression assesses one regressor at a time, we conducted an exhaustive analysis involving 12 regressors, three targets, and three setups **(Standard, Ridge, Lasso)**, totalling **108** models. Due to the extensive model count, our emphasis is on spotlighting the most significant findings.

For **Ridge regression**, we applied the standard procedure while incorporating a regularization term, which penalizes the model complexity. This approach controls the model's complexity by introducing a penalty term λ: minimize $J(\beta) = \sum(\mathbf{y} - \mathbf{X}\beta)^2 + \lambda \sum \beta^2$

For **Lasso regularization**, we followed the same procedure but introduced a regularization term that encourages the model to omit irrelevant features. It aims to regularize the model by shrinking the coefficients of unimportant features towards zero: minimize $J(\beta) = \sum(\mathbf{y} - \mathbf{X}\beta)^2 + \lambda \sum |\beta|$

A notable finding is the strong correlation **duratio_ms** and **n_beats**, initially evident from the correlation heatmap in Figure 4. The **Standard** and **Ridge** models both show the number of beats as a robust predictor for song duration (R^2 = **0.691**, **MSE** = **0.309**, **MAE** = **0.443**). See Figure 20 for an example.

Another notable model explores the connection between **danceability** and **valence**, revealing a less accurate prediction (R^2 of 0.248) with a slight positive relationship—suggesting 'happy' songs tend to be 'danceable', aligning with common musical experiences. This trend is consistent with Ridge results.

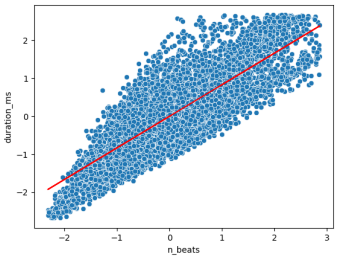


Figure 20: Simple regression: duration_ms against n_beats

Similarly, a less powerful prediction (R^2 = **0.128**) indicates that instrumental songs are generally less popular than vocal ones, reflecting the typical preference for vocal-driven tracks in popular rankings.

It's crucial to highlight that Lasso consistently performed worse than Ridge and Standard models. This can be explained as this regularization tries to lower the complexity, often finding and removing useless features. Yet, in simple regression there is only one feature, low complexity, and one regressor.

4.2 Multiple regression

Implementation & Conclussion

In this section, we delve into multiple regressors and modify our approach by considering three targets (**Duration_ms**, **Popularity**, **Danceability**) and using three setups (**Standard**, **Ridge**, **Lasso**). This leads to nine models, with each model employing multiple specifications.

The **Ridge regression** implementation was used similar to in simple regression, we iterated through alphas(Penalty term) from 0.1 to 0.00001.

For **Lasso regression**. We expected that Lasso would be compelling for this model section, given its effectiveness in handling irrelevant features. We experimented with alpha values ranging from 0.1 to 0.00001.

After experimenting with the penalty term 'alpha' in both **Ridge** and **Lasso** regressions, surprisingly, the results of simple regression outperformed the regularized models. This suggests that the model's complexity might not be significant enough to warrant the penalty term, as it fits the data well without it. Lower alpha values yielded better results in the regularized models.

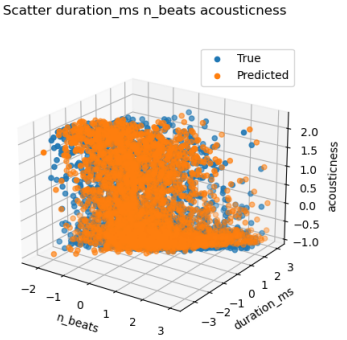


Figure 21: Multiple regression fitted results: duration_ms against n_beats and acousticness

Among the notable findings, for the target **duration_ms**, the number of beats and acousticness emerged as the most influential regressors, with respective weights: **W_nbeats** = 1.17484205 and **W_acousticness** =

-1.13964060e-02. This aligns with the simple regression results, indicating that both attributes significantly impact duration, but in opposing ways—higher acousticness correlates with a lower expected duration.

Figure 21 shows the relationship of the 3 variables together, how close was the prediction made and connected relationships i.e. an acoustic song tends to have less beats and the duration of the song tends to be shorter.

Target Variable	Sample size	K value	Weight
popularity	1500	35	Uniform
duration_ms	1500	33	Uniform
danceability	1500	47	Uniform

Table 4: K-nearest neighbours based regression models parameters table

4.3 Logistic regression

Implementation & Conclussion

Logistic regression was briefly explored since we wanted to check the predictive power of the data set on the **explicit** attribute, this can be useful to predict if a song would contain explicit word just by knowing it's general attributes, in terms of sensoring.

Logistic regression is used in categorical targets like binary or multiclass attributes, for our proposed model we used the already specified for regression continuous attributes, in terms of data specification we tried to delete all the songs that had no vocals from the data set, since we were working with explicit as target, the only songs to meet this criteria are songs which contain any verbal language, and so we used speechiness as a proxy for songs that contain language and songs that don't, exploring three different ranges, data with speechines > 0.6, 0.5, 0.4.

The best results were obtained with the specification of speechiness > 0.6, but still the data was still unbalanced, with Non-explicit = 1898 obs and Explicit = 242 obs.

Unfortunately, after setting up the model, it was still performing poorly on the data, after looking at potential reasons, we tried to remove all the songs that were not containing voices but still the results were not the desired. We still think it's worth to note the possibilities of this model.

4.4 K-nearest neighbours based regression

Implementation & Conclussion

The implementation of the K nearest neighbours regressor resamples the implementation of the K-nearest neighbours classifier model. As before we implemented the following techniques: varying sample sizes (10%, 20%, 30%, ..., 90%, 100% of the dataset), k fold cross validation (with a k value of 33), a range of weights (uniform and distance), a range of k, determined by finding a good approximation for the value of k (square root of the sample size) and using a range of values near it, composed by the first 10 odd numbers preceding and succeeding the previously determined approximation.

The biggest difference is just the model in itself, as a regressor instead of a classification model.

As we can see from 4.6 the regression model performed best in predicting **duration_ms**, while **popularity** and **danceability** showed moderate predictive capabilities since, when predicting popularity, the R2 score is very low, and, when predicting **duration_ms**, the Mean Squared Error and Mean Absolute Deviation are very high. So, the models are not as reliable as they are for **danceability**.

The results for this regressor reported in section 4.6 are obtained with the parameters in Table 4.

Target Variable	Criterion	Max depth	Min samples per leaf	Min samples per split
popularity	squared_error	10	100	2
duration_ms	squared_error	None	1	20
danceability	squared_error	None	20	2

Table 5: Decision tree-based regression models parameters table

4.5 Decision tree based regression

Implementation & Conclussion

For the implementation of the decision tree regressor we decided to iterate through various parameter to find the best model for our purposes: a range of criterions (squared error and friedman mse)[2], a range of max depths for the tree (None, 1, 5, 10, 20, 30, 50, 100), a range of minimum samples per leaf (1, 5, 10, 20, 30, 50, 100), a range of minimum samples per split (2, 5, 10, 20, 30, 50, 100) k fold cross validation (with a k value of 33),

As one can notice from Table 4.6 the results were consistent in both the tables, with the model performing its best with the danceability parameters.

The results reported in section 4.6 are obtained with the parameters in the table 5.

4.6 Results on test data

Target Variable	Model	R2 Score	MSE	MAD
popularity	multi reg	0.170	0.830	0.747
duration_ms	multi reg	0.947	0.053	0.155
danceability	multi reg	0.321	0.679	0.654
popularity	multi reg ridge	0.170	0.830	0.747
duration_ms	multi reg ridge	0.947	0.053	0.155
danceability	multi reg ridge	0.321	0.679	0.654
popularity	multi reg lasso	0.114	0.885	0.784
duration_ms	multi reg lasso	0.900	0.100	0.240
danceability	multi reg lasso	0.244	0.756	0.694
popularity	knn regressor	0.17	287.76	13.81
duration_ms	knn regressor	0.49	7906201932.14	19600.05
danceability	knn regressor	0.46	0.019	0.11
popularity	decision tree regressor	0.19	281.61	13.23
duration_ms	decision tree regressor	0.78	3372462023.22	10679.48
danceability	decision tree regressor	0.46	0.02	0.1

Table 6: Regression Results for Predicting Target Variables

[2]We also tried to implement absolute error and poisson but they resulted in a very computational heavy process, to the extent of more than 24 hours run of the program that resulted in a crash of visual studio code. We tried some handpicked combinations of the previously reported values but did not find any meaningful differences with the completely implemented criterions; therefore, we decided not to report on them.

5 Pattern Mining

5.1 Data Preprocessing

To conduct a meaningful association analysis, we employed data preprocessing and decided to use the following columns: `duration_ms`, `explicit`, `popularity`, `mode`, `genre`, `danceability`, `energy`, `speechiness`, `acousticness`, and `liveness`. The reason we chose these variables is because we deemed them as the most valuable ones to provide information on song characteristics and be useful when using patterns for predictions. Furthermore, we transformed continuous variables into categories using a binning technique.

5.2 Frequent Pattern Extraction

The Apriori technique was utilized on the preprocessed dataset, with parameters of a minimum number of items in an itemset to 2 and 25% support. The count of frequent item sets obtained equalled 104, as shown in Figure 22. The same parameters and dataset were employed with FP-Growth technique, and we obtained 81 item sets, as shown in Figure 23.

100	(veryLowLiveness, notExplicit)	61.653333
101	(Major, veryLowSpeechiness)	68.113333
102	(Major, veryLowSpeechiness, notExplicit)	65.300000
103	(Major, notExplicit)	69.653333
104	(veryLowSpeechiness, notExplicit)	87.313333

Figure 22: Apriori frequent itemset

..	...	...
77	(moderateDanceability, veryLowSpeechiness)	31.733333
78	(moderateDanceability, Major)	26.206667
79	(longDuration, notExplicit)	28.273333
80	(longDuration, veryLowSpeechiness, notExplicit)	26.133333
81	(longDuration, veryLowSpeechiness)	27.640000

Figure 23: FP Growth frequent itemset

5.3 Closed Itemsets Extraction

In terms of closed itemsets, employing the Apriori algorithm with a minimum support of 25% and a minimum of 2 items per set, we obtained a number of 104 closed itemsets. Applying the same parameters to FP Growth, the count of closed itemsets was 81. For both algorithms, the count of rules remained the same as for the frequent itemsets.

5.4 Maximal Itemsets Extraction

To extract maximal item sets, we employed the same parameters of 25% support and 2 minimum items per item set for both Apriori, which amounted to 25 itemsets, and FP Growth, which resulted into 20 itemsets.

5.5 Finding the Optimal Support

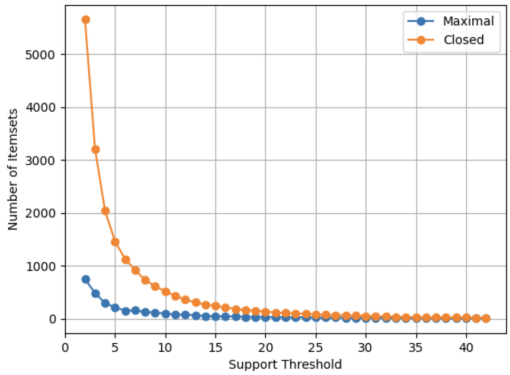


Figure 24: FP Growth support vs itemsets plot

Initially, we opted for a fixed support of 25% for each extraction type to analyse itemsets using various techniques. To determine the optimal support, we employed an iterative approach coupled with visualization to identify the most suitable support for each technique. Figure 24 illustrates the number of itemsets across a support range of 2% to the highest common support between maximal and closed item sets,

which was 43%, for the FP Growth algorithm. In this analysis, we set the minimum number of items per set to 2. The plot indicates that the optimal minimum support is around 5%, but such a low support may not be ideal given the substantial amount of data involved.

5.6 Rules Extraction

To uncover association rules, we employed the two algorithms on the dataset with a confidence level of 70%. The extraction criteria included a minimum of two items and a support level of 25%. For Apriori, it yielded 236 rules, where the top rules are visualized in Figure 25. For FP Growth, it extracted 177 rules, where the top ones were identical to the ones mined by Apriori. For both algorithms, the highest lift value observed was circa 1.55.

consequent	antecedent	abs_support	%_support	confidence	lift
veryLowAcousticness	('veryHighEnergy', 'veryLowSpeechiness')	4355	29.03333333333333	0.8325367998470655	1.545742294554522
veryLowAcousticness	('veryHighEnergy', 'Major', 'veryLowSpeechiness')	3143	20.953333333333333	0.8323622881355932	1.545418284692895
veryLowAcousticness	('veryHighEnergy', 'veryLowSpeechiness', 'notExplicit')	4009	26.726666666666667	0.824049331963001	1.5299839063553677
veryLowAcousticness	('veryHighEnergy', 'Major')	3410	22.733333333333334	0.8238705001208021	1.5296518754563724
veryLowAcousticness	('veryHighEnergy',)	4718	31.453333333333333	0.8229548229548229	1.5279517693182751

Figure 25: Rules extraction using Apriori

5.7 Optimization of Confidence and Support for Rules

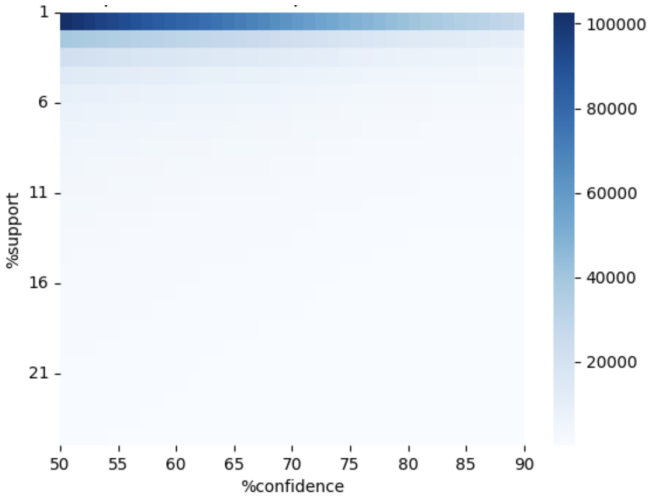


Figure 26: Optimization heatmap

We employed an iterative approach to enhance both confidence and support, with the FP Growth algorithm undergoing testing across different combinations.

In Figure 26, the heatmap illustrates the diminishing number of extracted rules as support increases. We achieved near-optimal results, extracting a significant number of rules at 25% support and 65% confidence, with 208 rules.

5.8 Target Variable Prediction

We have selected the **explicit** variable as our target, specifically when the value is False or "notExplicit". Using the FP-Growth rules, according to Figure 27, when **duration_ms** is considered very long, thus between the range of [240000, inf], and when **liveliness** and **speechiness** are considered very low, thus they fall inside the range [0.0, 0.2], the song is classified as "*notExplicit.*"

consequent	antecedent	abs_support	%_support	confidence	lift
notExplicit	('veryLongDuration', 'veryLowLiveness', 'veryLowSpeechiness')	4097	27.313333333333333	0.9710831950699218	1.0379256039652862
notExplicit	('unpopular', 'veryLowSpeechiness')	4610	30.733333333333334	0.9662544539928736	1.0327644869526225
notExplicit	('veryLongDuration', 'veryLowLiveness')	4250	28.333333333333332	0.9654702407996365	1.0319262941424077
notExplicit	('veryLowLiveness', 'Major', 'veryLowSpeechiness')	6480	43.2	0.963855421686747	1.030200322452701
notExplicit	('veryLongDuration', 'veryLowSpeechiness')	6054	40.36	0.9632458233890214	1.0295487637762093

Figure 27: Explicit as target variable for pattern mining