UNIVERSITÀ DI PISA

Department of Computer Science

Project Report of Data Mining 1
Spotify Tracks Dataset

Toscano Giuseppe Vincenzo Dylan
Sacco Giuseppe

ACADEMIC YEAR 2023-2024

Contents

Chapter 1

Introduction

The aim of this project is to apply the principal data mining techniques to the Spotify Tracks dataset. The dataset contains data about audio tracks available in the Spotify catalogue. These tracks span 20 distinct genres, including techno, IDM, study, and black-metal. Each track is provided with fundamental attributes such as track name, artist, album name, and its popularity within the catalogue. Additionally, audio-derived features are included, encompassing aspects like danceability, energy, key, and loudness.

By leveraging these features, we intend to build models that can accurately classify tracks into their respective genres. This report contains the whole analysis performed on the dataset in four stages: Data Understanding and Preparation, Clustering, Classification, Pattern Mining.

Chapter 2

Data Understanding and Preparation

2.1 Data Semantic

In data semantic phase, we initiated our analysis with two primary datasets: test.csv and train.csv. Subsequently, these datasets were merged into a singular dataset, resulting in a unified corpus comprising 20000 rows and 24 features. The features are described in the following table:

Feature	Description	Type
name	Title of the track	object
duration_ms	Track length in milliseconds	int64
explicit	Whether or not the track has explicit lyrics (True = yes; False = no/unknown)	bool
popularity	Popularity of a track (0 to 100)	int64
artists	Artist(s) who performed the track	object
album_name	Album in which the track appears	object
danceability	How suitable a track is for dancing (0.0 to 1.0)	float64
energy	Perceptual measure of intensity and activity (0.0 to 1.0)	float64
key	Key of the track (standard Pitch Class notation)	int64
loudness	Overall loudness of the track in decibels (dB)	float64
mode	Modality (major or minor) of the track (major=1, minor=0)	float64
speechiness	Detects the presence of spoken words in the track	float64
acousticness	Confidence measure from 0.0 to 1.0 of whether the track is acoustic	float64
instrumentalness	Predicts whether a track contains no vocals	float64
liveness	Detects the presence of an audience in the recording	float64
valence	Musical positiveness conveyed by a track (0.0 to 1.0)	float64
tempo	Overall estimated tempo of a track in beats per minute (BPM)	float64
features_duration_ms	Duration of the track in milliseconds	int64
time_signature	Estimated time signature	float64
n_beats	Total number of time intervals of beats throughout the track	float64
n_bars	Total number of time intervals of bars throughout the track	float64
popularity_confidence	Confidence of the popularity of the song (0.0 to 1.0)	float64
processing	No information available	float64
genre	Genre of the track	object

Table 2.1: Description of Dataset Features

2.2 Distribution of the variables and statistics

Our initial exploration involved analyzing the distribution of each variable in the dataset. We employed descriptive statistics such as mean, median, standard deviation, minimum, and maximum values to get information about the central tendency and spread within the data. The features founded are divided as follow:

- Continuous variables: duration_ms, popularity, features_duration_ms, danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo, n_beats, n_bars, popularity_confidence.
- Discrete variables: key (12 values), processing (12 values) and time_signature (5 values).
- Categorical variables: mode and explicit, name, artists, album_name and genre (these will be label encoded in order to perform the analysis with all numerical columns).

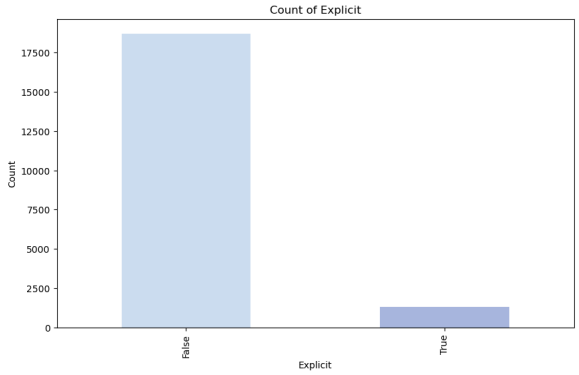

Figure 2.1: Explicit distribution

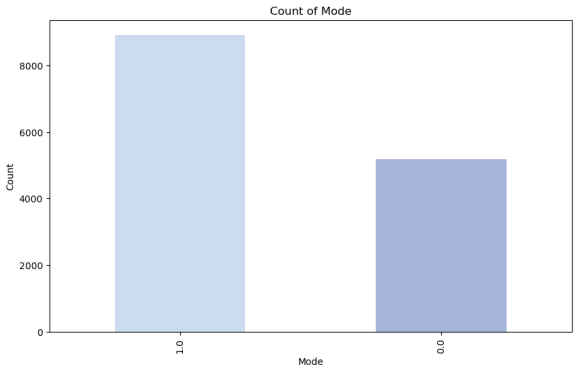

Figure 2.2: Mode distribution

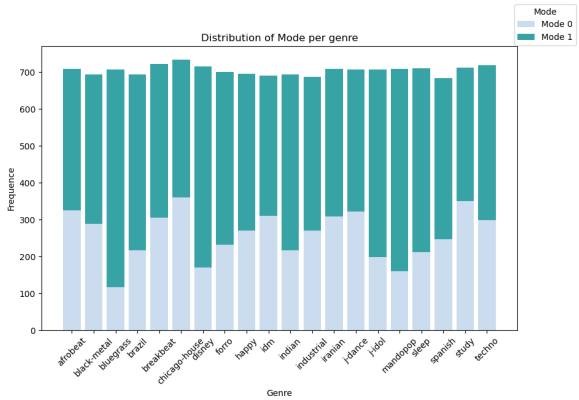

Figure 2.3: Distribution of mode per genre

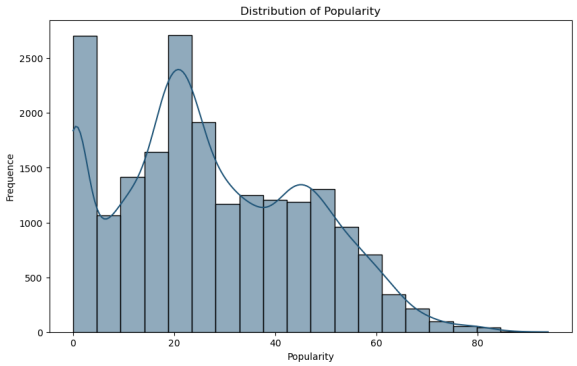

Figure 2.4: Distribution of Popularity

In Figure 2.1 we plot "explicit" distribution: the feature is unbalanced between its two values, in fact we have over than 17500 false value and slightly less 2500 true values, making this feature less informative. In Figure 2.2 we plot "mode" distribution: the feature suggests us that most dataset songs have 1.0, which means that are major, but, how we can see in Figure 2.3 mode is similarly distributed for each genre except for "bluegrass". Finally, analyzing popularity in Figure 2.4, we noticed that there are many songs with a low value, especially for values near to 0 and 20, and when this value increases, there are fewer and fewer occurrences in the dataset.

2.3 Data Quality

We examined the dataset for potential issues such as missing values, outliers and inconsistencies. Techniques like identifying rows with missing entries and analyzing extreme data points will help us to know about the presence of outliers.

2.3.1 Missing Values

Upon examining the dataset, we discovered that only three features presents missing values. These features are:

- Mode: 5911 out of 20000
- Time_signature: 2787 out of 20000
- Popularity_confidence: 17062 out of 20000

Figure 2.5: Distribution of Missing Values for Features that have Missing Data

As illustrated in Figure 2.5, the feature "popularity_confidence" exhibits a significantly high proportion of missing values (approximately 85%), while "mode" has about 30% missing values. Consequently, we have decided to address these issues at a later stage. For the "time_signature" feature, we initially decided to substitute the missing values with the mode of the song's genre. Subsequently, we noticed that the mode for the most part of genres is equal to 4 (Figure 2.6). Therefore, we chose 4 as the value to fill the null entries.

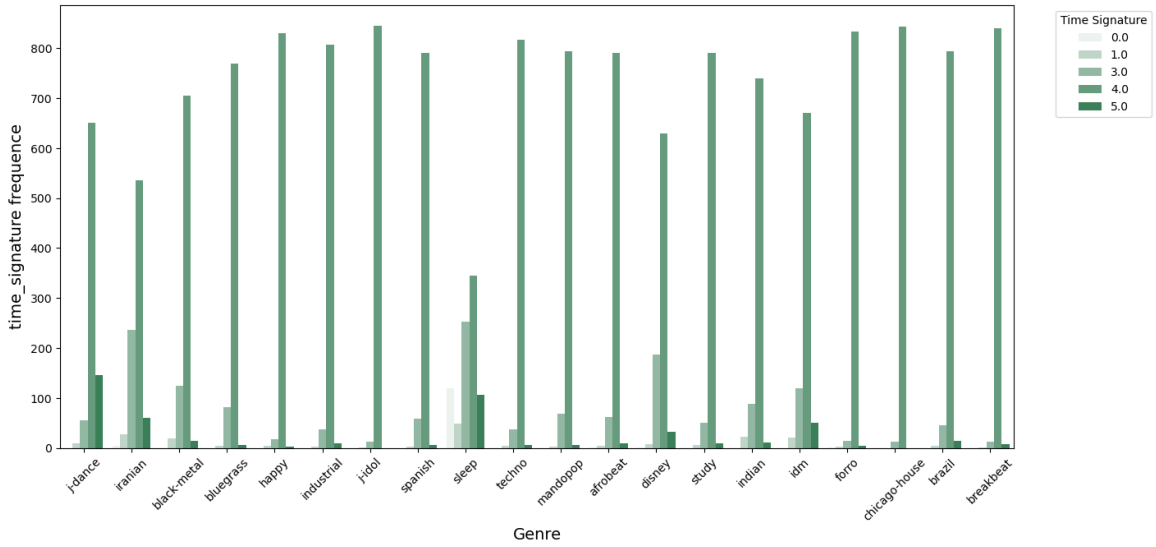

Figure 2.6: Distribution of Time Signature per Genre

2.3.2 Outliers

We had found out two solutions to identify outliers:

- PCA
- Quartiles

PCA

This approach is employed to decrease the data's dimensionality. In this instance, PCA was utilized to identify outliers by projecting the data into a three-dimensional space using three principal components (we discuss it later). By referring to Figure 2.7, we discern a predominant cluster of data points visually, with outliers identified as those significantly distant from this cluster.

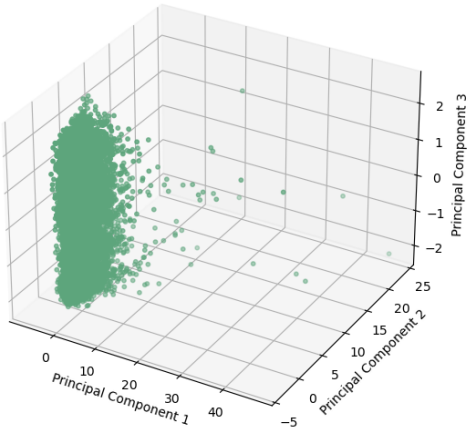

Figure 2.7: PCA outliers

Quartiles

In this section, we use box plots to visually identify outliers. We create graphs for all continuous variables and highlight the three that clearly show outliers: "speechiness", "duration_ms", and "loudness" (Figure 2.8). Additionally, we use scatter plots to visualize data points that stand out significantly from the rest. The box plots make it easy to see the range and distribution of the data, while the scatter plots provide a clear view of the individual outliers.

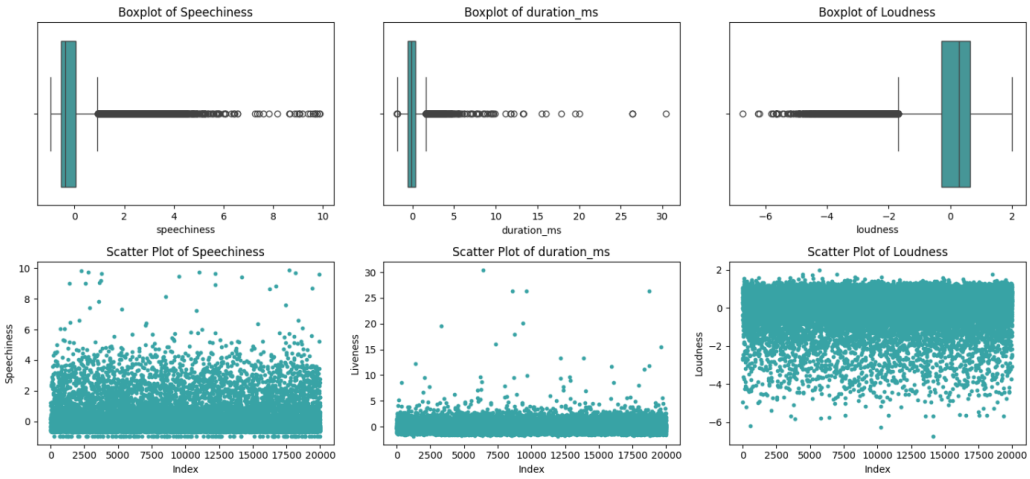

Figure 2.8: Box plot outliers in Speechiness, Duration_ms, Loudness

2.4 Correlations and eventual elimination of variables

We calculated the correlation matrix with Pearson method to assess the interrelationships between all variables. As we can see in the Figure 2.9 we identify highly correlated features like "n_bars" and "n_beat",

probably because both measure the rhythmic structure of the songs in fact they have highly correlation with "duration_ms" and "tempo". Furthering analysis, we identify: "energy" and "loudness" with correlation of 0.72 ; "energy" and "acousticness" with -0.7 negative correlation; "loudness" and "acousticness" with -0.55 negative correlation; "loudness" and "instrumentalness" with -0.45 negative correlation; "features_duration_ms" and "duration_ms" with a correlation of 1. We tought that this correlations are significative because of the genre of the song, for example if a song is more energetic probably it will be less acoustics. Since strong correlations can indicate redundancy, we carefully evaluated the context and meaning of these variables before eliminating any.

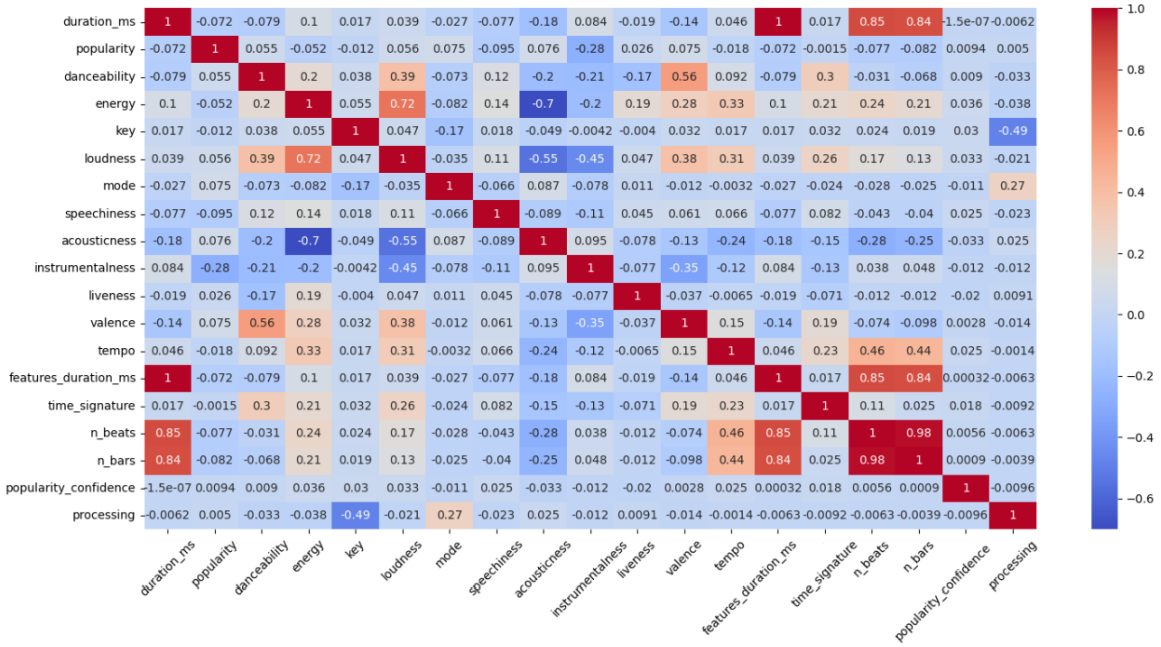

Figure 2.9: Correlation Matrix with encoded genre

After seen the correlation matrix and the previous analysis, we decided to eliminate the sequent features:

- popularity_confidence: due to the high number of missing values (Figure 2.5);
- mode: due to distribution among genres, as we can see in Figure 2.3, then we decided to not fill missing values and to remove the feature;
- feature_duration_ms: the values of this feature are the same as the feature "duration_ms";

2.5 Data sampling

We choose to create a sample from the original dataset. To ensure adequate representation of each musical genre, we employed stratified random sampling.For each genre, we randomly selected 400 songs, resulting in a total of 400 songs per genre (8000 songs in total). This approach ensured that all genres were equally represented within the sampled dataset. Our future data exploration and analysis will be on both the original dataset and the sampled. This allowed us to compare the results and assess the potential impact of sampling on our findings.

Chapter 3

Clustering

In this section, we describe the applications of the three clustering algorithms (KMeans, DBScan, Hierarchical) applied to the dataset and their respective results. Before starting, we select only the features that are necessary for the clustering process, i.e. numeric features. Categorical columns of type object, such as name, artists and album_name and genre, were removed; columns with discrete values, such as key, processing, explicit and time_signature, were also not selected. Furthermore, based on the study conducted on the dataset in the previous steps, we decided to apply clustering on two datasets to understand how the results vary based on different inputs. We will refer to the original dataset as X and the dataset of 8000 rows obtained during the sampling phase as XS.

3.1 Analysis by centroid-based methods

In this section, we use centroid-based clustering methods to analyze the song dataset. Centroid-based methods partition the data into clusters, where each cluster is represented by the center (centroid) of the data points.

3.1.1 K-Means

To determine the optimal number of clusters for our data, we employed the elbow method by calculating both the silhouette score and the sum of squared errors (SSE). We computed the SSE and silhouette score for various initial values of k (from 2 to 20) using the k-means algorithm. Subsequently, we applied the elbow method to the resulting graph to identify the best k (Figures 3.1, 3.2).

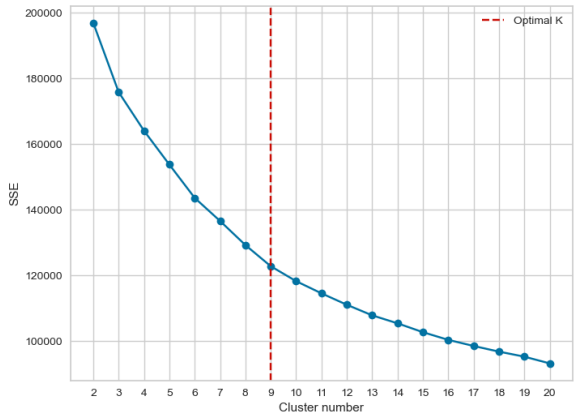

Figure 3.1: SSE score for dataset X

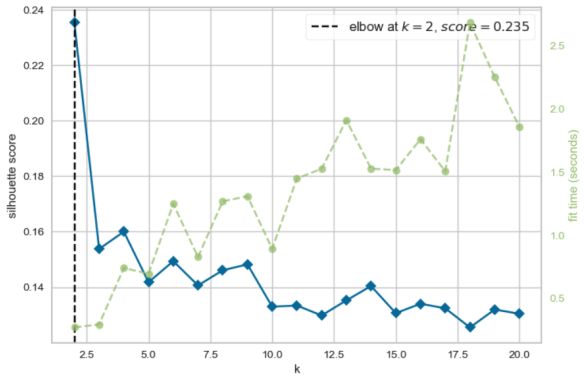

Figure 3.2: Silhouette score for dataset X

As shown in the figure, the elbow method yields an optimal value of $k = 9$ for both the datasets X and XS. Regarding the silhouette score, the composition of the obtained graph does not allow us to accurately determine the optimal value of k using the elbow method.

In conclusion, we have chosen an optimal value of $k = 8$ to reconcile the values obtained in terms of 'fit time' as indicated by the silhouette score graph.

At this point, we aim to evaluate the correctness and optimality of the clustering obtained with $k = 8$.

From the distribution plots (Figure 3.3), we can observe that the data points are not well distributed across the various clusters, but neither unbalanced in a single cluster.

The distance plots (Figures 3.5, 3.6) illustrate the composition of the clusters, with each line representing the coordinates of a cluster centroid.

Focusing on the three features (Figure 3.4) that seem to best identify the clusters (loudness, speechiness, and liveness), we create a 3D scatter plot to visually observe the cluster compositions.

The resulting plot indicates that the clusters obtained are not well-separated. To also verify this mathematically, we employ the ratio $\frac{BSS}{WSS}$ (Between-Cluster Sum of Squares by Within-Cluster Sum of Squares). The ratio calculated for our data yields a value of 0.86, which, being less than 1, signifies that our clustering is not well-separated.

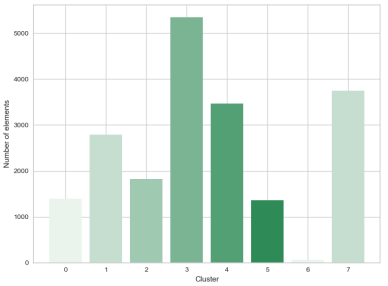

Figure 3.3: K-means clusters distribution for dataset X

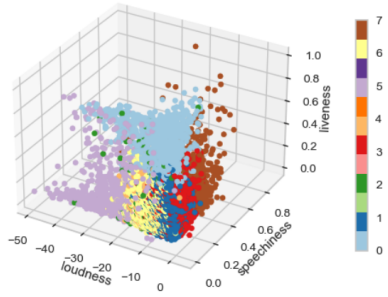

Figure 3.4: 3D scatter plot of 3 selected features

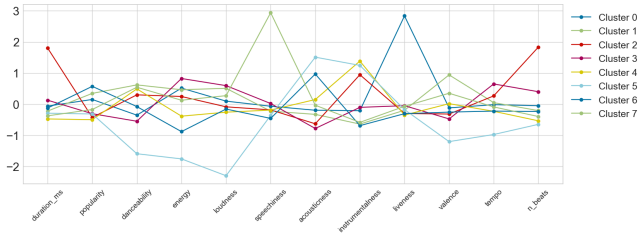

Figure 3.5: Clustering distance for X

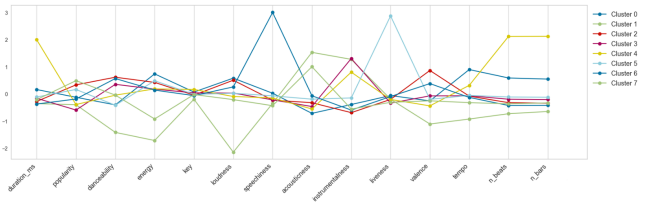

Figure 3.6: Clustering distance for XS

3.1.2 Bisecting K-Means

For this clustering method, we ran the Bisecting k-means algorithm 21 times with different values of k, calculating SSE and Silhouette Score each time. Using the elbow method, we figured out the best number of clusters based on these metrics.

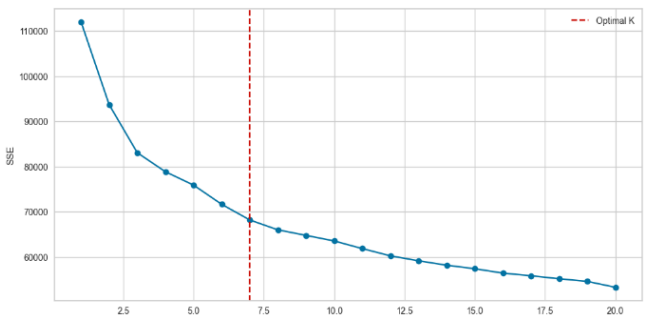

Figure 3.7: SSE score for dataset XS

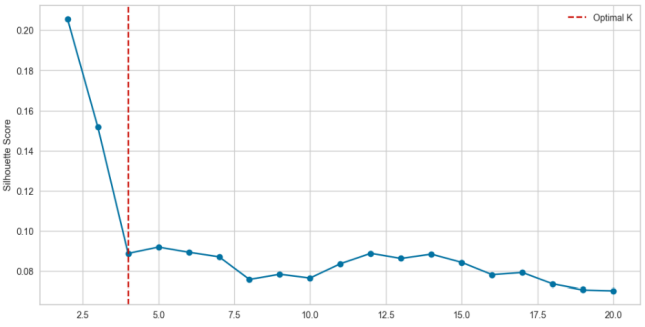

Figure 3.8: Silhouette score for dataset XS

SSE suggested 7 clusters (3.7) were optimal for XS dataset while for dataset X were 6 (3.9), while Silhouette Score indicated 4 (3.8) in XS and 3 in X (3.10). As in K-Means, the knee-locator function for the elbow method

is not helpful with the silhouette score graph because of the lack of convexity. At this point, we choose 7 as optimal number of cluster for XS dataset, also looking at silhouette graph we don't lose a lot of silhouette score from 5 clusters to 7. Meanwhile, for dataset X we choose 6 as optimal value as SSE graph suggest us, and, with the help of silhouette graph, we can see that 6 is the best cluster number with good silhouette score we can achieve with an SSE value relatively low.

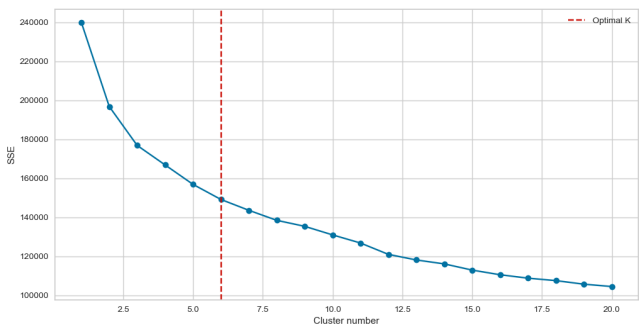

Figure 3.9: SSE score for dataset XS

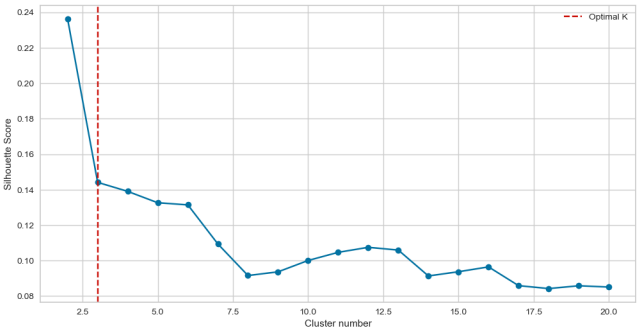

Figure 3.10: Silhouette score for dataset XS

Figure 3.11 shows the clusters plotted in a 3D space using liveness, loudness, and energy dimensions chosen because to make the clusters easier to see. It's clear that the clusters aren't well-separated, likely because of the nature of the data itself. In the Figure 3.12 we can see how the cluster are separated after bisecting k-means execution on XS dataset. The cluster sizes vary widely, this demonstrates that clusters are not well-separated.

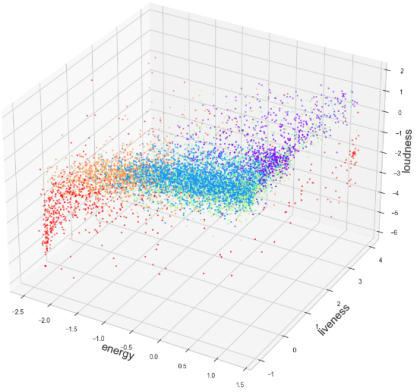

Figure 3.11: Clusters of dataset XS plotted in 3D

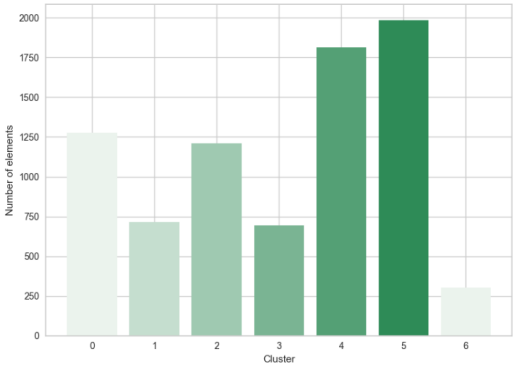

Figure 3.12: Distribution of XS dataset's clusters

3.2 Analysis by density-based clustering

3.2.1 DBSCAN

We applied DBSCAN to the dataset, by varying the eps parameter. We used a range of eps values from 1.5 to 3.4 with a step of 0.1 for each iteration.

During the clustering phase with DBSCAN, we tried to determine the optimal eps parameter to balance the best (possible) high silhouette score with a reasonable number of clusters.

From Figure 3.13, about X dataset, it is clear that achieving a high silhouette score would necessitate selecting an eps value around 2.5 or 2.6, resulting in only two clusters and compromising the data separation quality.

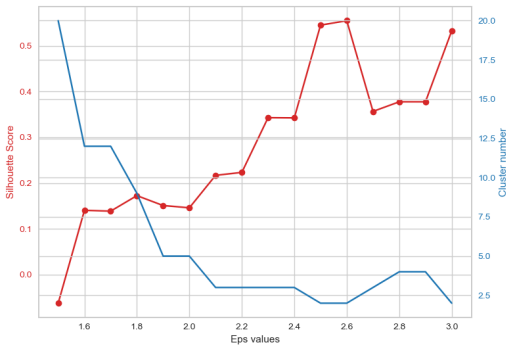

Figure 3.13: Finding best eps value for DBSCAN for X dataset

However, an eps value of 1.9 appears to be more promising, although with a low silhouette score of 0.15 and a resulting of 5 clusters.

Examining Figure 3.14 we can see that the DBSCAN approach resulted in five unbalanced clusters plus the cluster 0 that is the cluster that contains 480 noise points. We can notice that a single cluster contains a high number of data points.

Regarding XS dataset, we identified the optimal eps value to be 2.2, resulting in 5 clusters. The silhouette score for this configuration was found to be 0.396, indicating a relatively good clustering performance. Notably, DBSCAN identified 388 noise points, emphasizing its ability to isolate outliers effectively. This result as a bad clustering approach for our dataset.

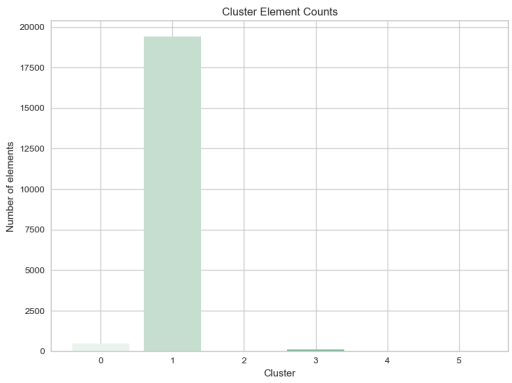

Figure 3.14: DBSCAN clusters distribution

3.2.2 OPTICS

This clustering method revealed his problem with high dimensional dataset, as is ours. We tried different starting parameter for the run, the main appears to be *min_cluster_size*, which indicate the minimum number of data point to identify a cluster. With a value of, 0.1 which means at least 100 points, it results in one big cluster and only 8 noise points (Figure 3.15). We also tried to modify the *xi* parameter: increasing it we reduce the number of noise point, but the result is even one big cluster. The best way to separate dataset in more than 1 cluster seems to decrease the value of *mi_cluster_size*, but the result is again one big cluster and other small clusters.

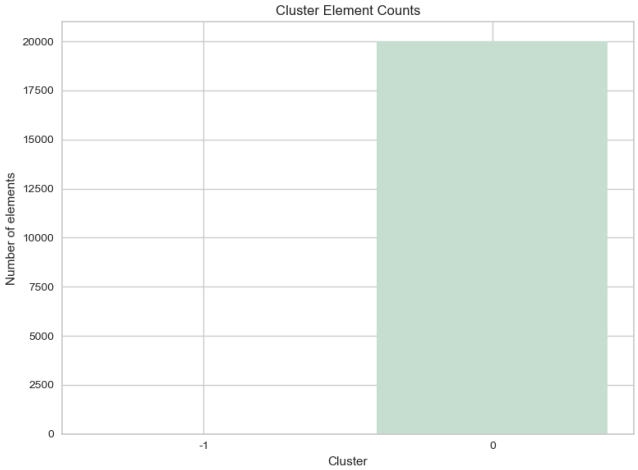

Figure 3.15: OPTICS cluster result for dataset X

3.3 Analysis by hierarchical clustering

We use agglomerative hierarchical clustering. This method helps us to visualize the nested relationships between clusters. By cutting the dendrogram at third level we obtain this groups as in Figure 3.16. Then we fix the number of cluster to 6 and calculate the silhouette score and clusters distribution for each selected linkage method (Single, Average, Complete and Ward) and dataset (Figure 3.17). The result of our analysis is that X

dataset has a better silhouette score for every linkage method, except for ward method, where the two datasets have the same values. The best linkage methods are single and average for both dataset, if we look at silhouette score, but if we assess the cluster's distribution, we can see that it is unbalanced, so this means that it is not a valid clustering separation. Taking a look at the Ward method linkage, it seems to have the data point to be well distributed among clusters, but with a low silhouette score.

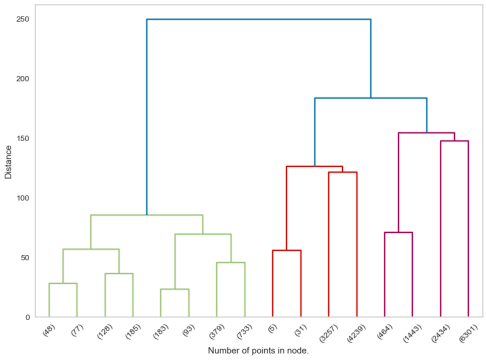

Figure 3.16: Dendrogram of X dataset with p=3 and ward distance method

linkage	dataset	silhouette score	cluster
Single	X	0,79	[2, 2, 3, 1, 19991, 1]
Single	XS	0,71	[7994, 1, 1, 1, 1, 1, 1]
Complete	X	0,35	[19860, 5, 103, 28, 2, 2]
Complete	XS	0,15	[6376, 954, 650, 1, 3, 2, 14]
Average	X	0,63	[2, 19964, 2, 2, 27, 3]
Average	XS	0,39	[25, 14, 7955, 1, 2, 1, 2]
Ward	X	0.08	[7532, 1826, 5131, 1907, 2434, 1170]
Ward	XS	0,08	[1541, 605, 1203, 790, 530, 2324, 1007]

Figure 3.17: Euclidean distance table of various linkage for X and XS dataset

3.4 Clustering result and discussion

The results obtained show centre-based method clustering algorithms like K-Means and Bisecting K-Means effectively partition the data, and work best with our dataset in contrast to density-based method clustering, and hierarchical clustering, which suffer from the high-dimensionality of data and density. Of the density-based clustering algorithms, OPTICS turned out to be the worst in this case, while DBSCAN performed better, but in spite of this, we judged its performance to be poor. With regard to the centre-based clustering algorithms (K-means, Bisecting K-means), we obtained distinct clusters with different distributions for both. Despite good silhouette values, the clusters turned out to be not well-separated. Finally, hierarchical clustering algorithms doesn't work too because of the density of our dataset.

For curiosity, we apply the PCA with 5 components to our dataset and then clustering on the results. We used K-means because it works well with our dataset and DBSCAN for seeing how it works with lower dimensionality. DBSCAN works bad anyway, meanwhile with K-means we obtained better results. We choose, after analysis, 7 as best number of cluster: as we can see in Figure 3.18, clusters are better distributed, although they are always not well separeted.

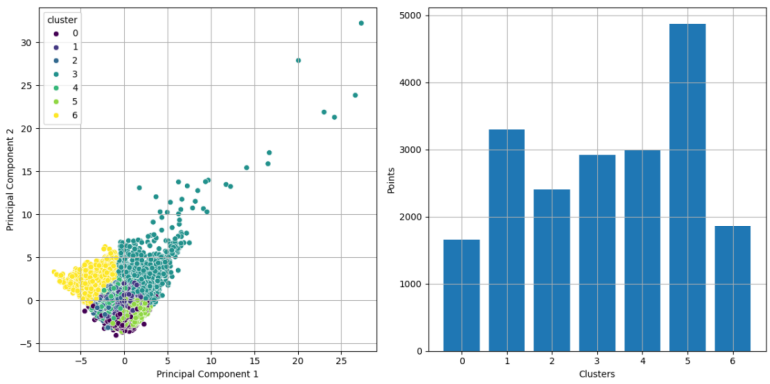

Figure 3.18: K-means results after PCA

Chapter 4

Classification

In this section we use supervised algorithm, therefore we have to chose one target variable. We have chosen the feature "genre", we believe that is the most suitable category for classification. In addition, we wanted to explore whether the various genres were distinct or had similarities.

4.1 Pre-processing

We prepared the data for classification using only numerical features as independent variables. The entire dataset of 20000 entries was divided into training set (80%) and test set (20%), then we scaled it using the **StandardScaler** from sklearn.preprocessing.

4.2 Decision-Tree

Before applying the algorithm, we used GridSearch, and accuracy metric, to figure out the best hyperparameters to set. For the gain criterion for trees, we obtain "Gini". For "max_depth" investigating between 5, 10, 15, 20, we got 10 as the optimal value; for "min_samples_split" investigating between 2, 5, 7, 9 we got 2 as the optimal value, "min_samples_leaf" investigating between 1, 2, 4, 6 we got 4 as the optimal value. As we can see in the Figure 4.1, we obtained 47% accuracy, not so good. Average accuracy and recall are balanced at about the same level, and F1-score is 48% circa, Micro and Macro AUC are both 0.70, which means model's ability to correctly distinguish classes is modest. Confusion Matrix (Figure 4.2) reveals how some genres are better predicted like sleep and study respect others.

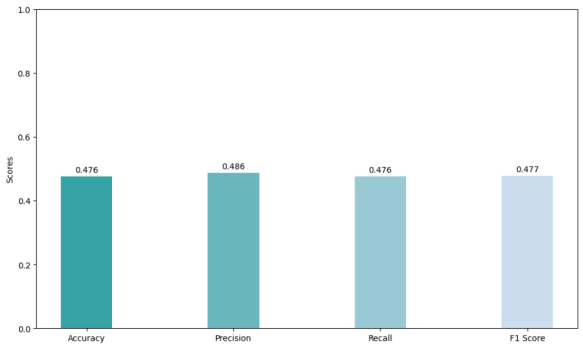

Figure 4.1: Decision Tree Metrics

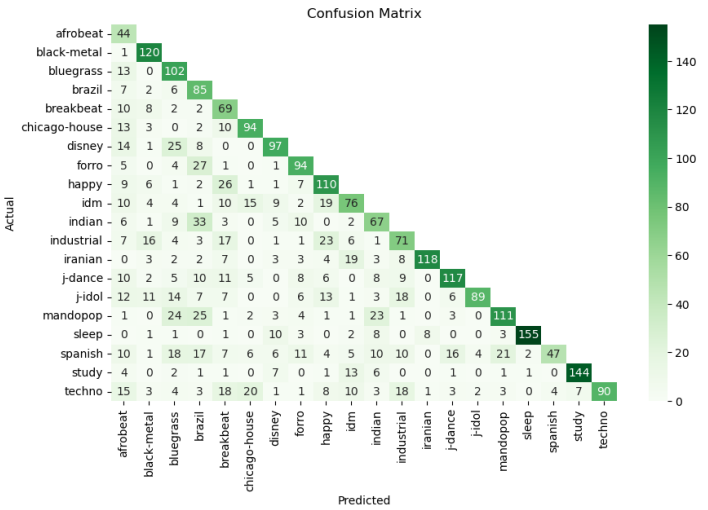

Figure 4.2: Decision Tree confusion matrix (only True positive and False Positive)

As we can see in Figure 4.3 The results of the first layer of the decision tree suggest that "instrumentalness" is an important variable for the classification of musical genre, in fact is the root. Then we found "popularity" and "danceability".

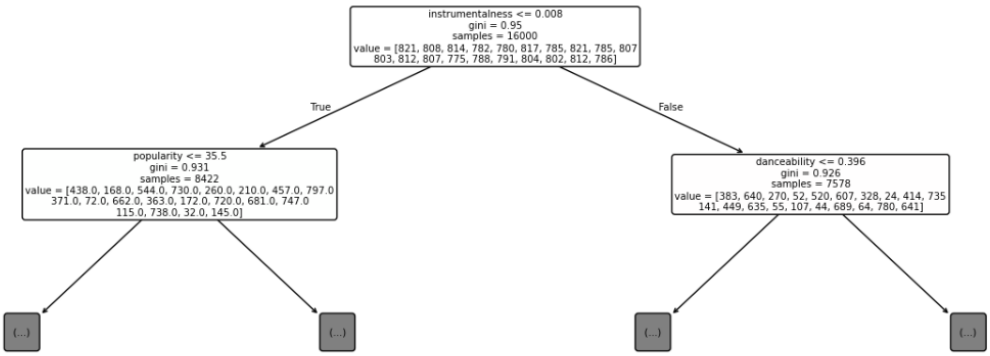

Figure 4.3: first layer Decision Tree

4.3 K-Nearest Neighbors

As before, we used GridSearch to figure out the best hyperparameters to set. For the best metric to use we obtain Manhattan, while for "n_neighbors" investigating between 3, 5, 7, 9, we got 9 as the optimal value. As we can see in the Figure 4.4 KNN achieved an accuracy of 52%. The model demonstrates slightly better prediction ability than Decision Tree, with average accuracy and recall both around 52%. The Micro and Macro AUC for KNN are both 0.83, suggesting good accuracy in predictions. We can notice that the confusion matrix resulted (Figure 4.5), similarly to Decision Tree, predict the two genres study and sleep better than others, and it improves also the prediction of others genres.

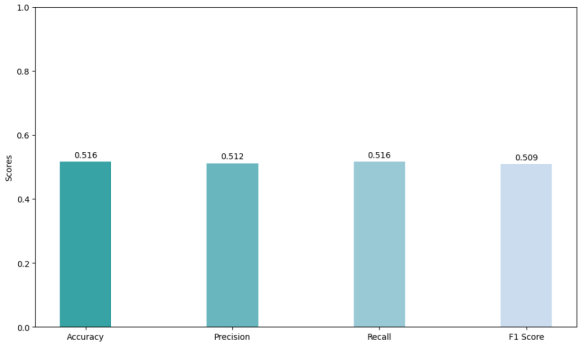

Figure 4.4: KNN Metrics

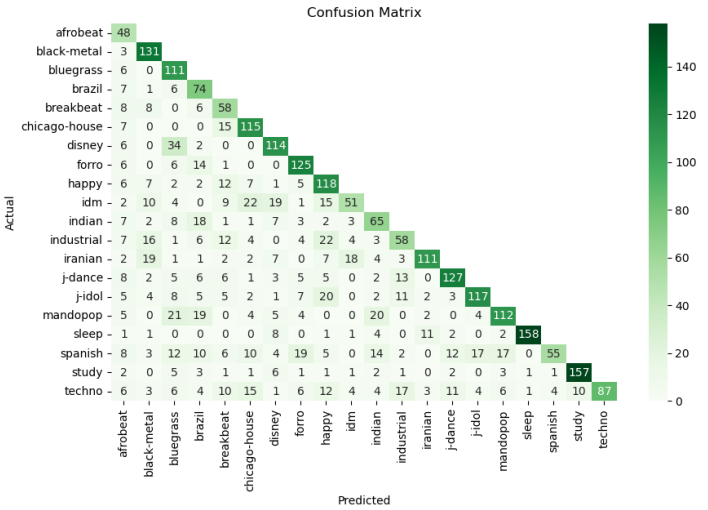

Figure 4.5: KNN confusion matrix (only True positive and False Positive)

4.4 Naive Bayes

We used the Gaussian variant of the Naive Bayes algorithm.

As we can see in the Figure 4.6 despite an accuracy of 27%, Naive Bayes showed very low average precision and recall, indicating difficulty in correctly discriminating classes. The model achieved an average F1-score of 25%, suggesting that its classification ability is limited. As for the Micro and Macro AUC of 0.86, having had

discordant values with the metrics, suggest to us that Naive Bayes is unable to model the relationships between the variables correctly, probably because of the data itself. The Confusion Matrix (Figure 4.7), shows how the Naive Bayes performs poorly for our dataset.

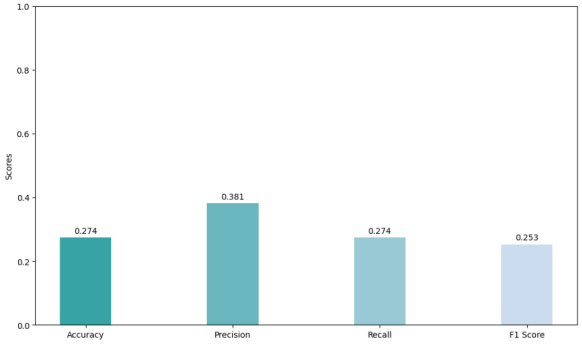

Figure 4.6: Naive Bayes Metrics

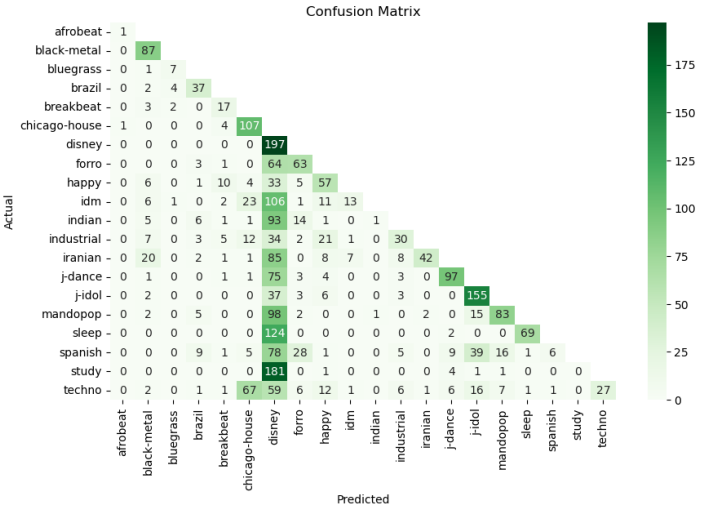

Figure 4.7: Naive Bayes confusion matrix (only True positive and False Positive)

4.5 Classification result and discussion

In the analysis on the genre classification of songs, the KNN proved to be the most effective with an accuracy of 52%, showing a higher predictive ability than the other models tested. The decision tree achieved an accuracy of 47%, while the Naive Bayes encountered difficulties with an accuracy of 27%. These results indicate that the KNN is the preferred model for the prediction of genres based on numerical features. This probably due to the uniform distribution of the features.

After that, we tried to run KNN on the dataset with lower number of genre, because we thought that we can increase classification performance. Based on the results of the K-means applied after PCA in the Section 3.4, we looked for how the genres were distributed in the clusters obtained. We then decided to create 5 macro-genres as in Figure 4.8

Macrogenres	Genres
macrogenre_1	bluegrass, brazil, forro, indian, mandopop, spanish
macrogenre_2	idm, j-dance, techno
macrogenre_3	disney, iranian, sleep, study
macrogenre_4	afrobeat, breakbeat, chicago-house, happy, j-idol
macrogenre_5	black-metal, industrial

Figure 4.8: Macrogenres

Then we apply the classification algorithm on the dataset, where we have replaced the genre with the macrogenre to which it belongs, and the results are encouraging. As we can see in Figure 4.9, each algorithm improved the previous results. Despite this, the KNN is still the best.

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.67275	0.6685	0.67275	0.66632
K-Nearest Neighbors	0.698	0.6946	0.698	0.6913
Naive Bayes	0.61475	0.6128	0.61475	0.6059

Figure 4.9: Classification on macrogenres

Macrogenre_1 and Macrogenre_3 showed the best performance, both in terms of precision, recall and F1-score, and in terms of AUC (0.90) as we can see in Figure 4.10, indicating that the model is very effective in classifying these two classes. Mocrogenre_2 has the worst performance, with significantly lower precision and recall than the other classes, indicating that the model has difficulty correctly distinguishing this class. Macrogenre_4 and Macrogenre_5 have intermediate performance with AUCs of 0.85 and 0.87 respectively, indicating that the model has good classification capabilities for these classes.

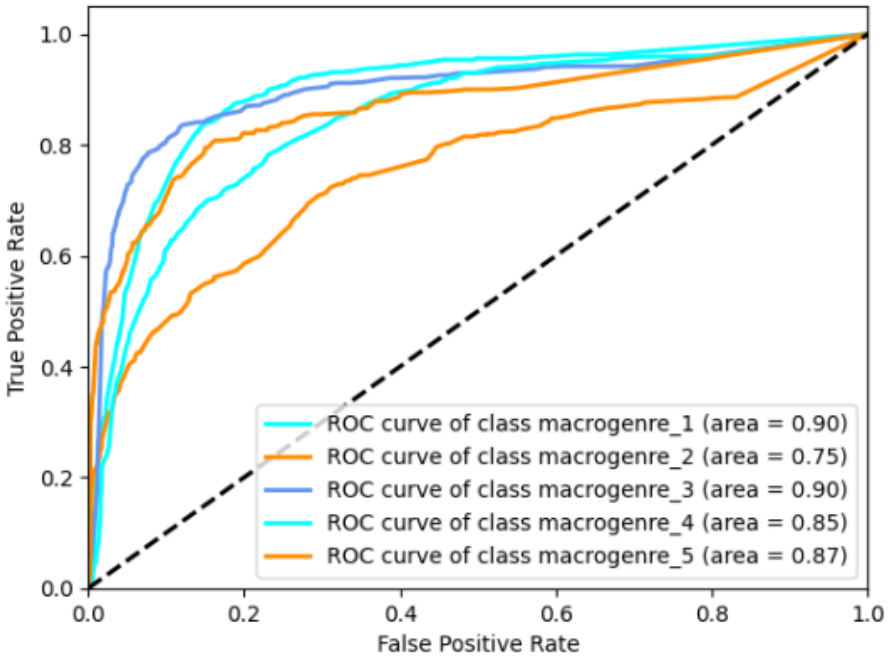

Figure 4.10: Roc Curve of macrogenres

Chapter 5

Pattern mining and Regression

5.1 Regression

5.1.1 Univariate Regression

First of all we decided the features to which to apply the linear regression, based on the correlation matrix in the Figure 2.9. We choose the pairs energy-loudness due to 0.7 correlation, loudness-acousticness due to -0.5 correlation and danceability-valence due to 0.56 correlation.

5.1.2 Multiple Linear Regression

We decided to select "loudness", "acousticness", "tempo", "valencev as independent features and "genre" as dependent feature. Initially, we find the best parameters for the models with GridSearchCV.

First of all we apply linear regression, and we obtained 0.0231 MSE (Mean Squared Error), 0.1154 MAE (Mean Absolute Error), and 0.6777 R^2 (Coefficient of determination), suggesting that the model explains circa 67% of data variability.

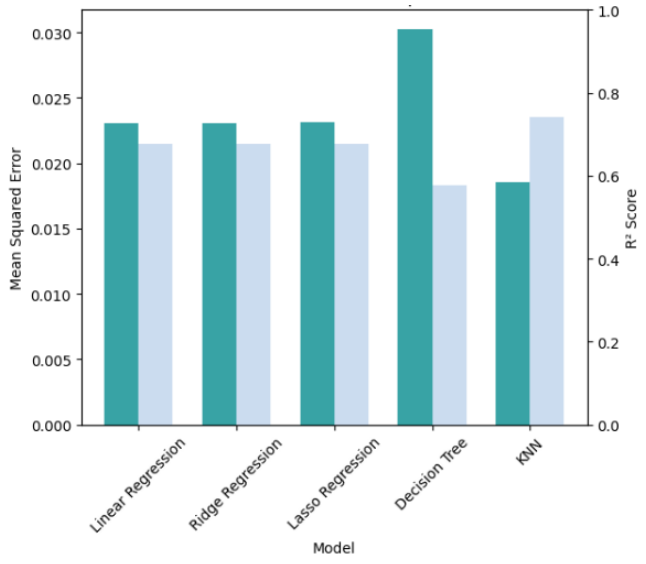

Figure 5.1: Models Evaluation

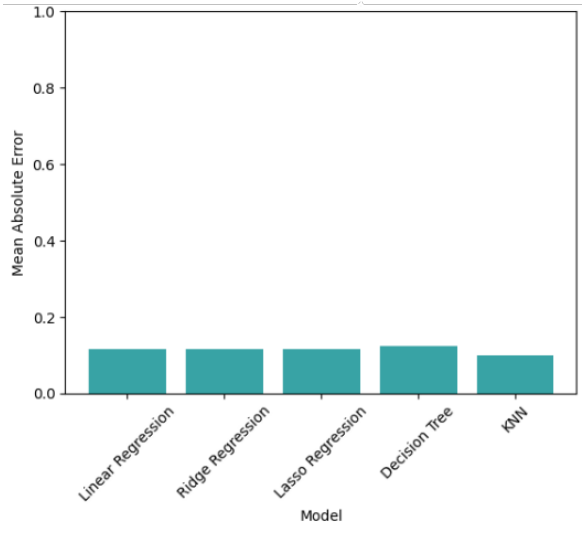

Figure 5.2: Mean Absolute Error

Applying the ridge regression and lasso regression, we obtained similar values: for ridge, 0.023 MSE, 0.115 MAE, and 0.677 R^2; for lasso, 0.023 MSE, 0.115 MAE, and 0.677 R^2. These results indicate that the parameters might be suboptimal. After changing the lambda of each model, we get the same values for ridge and worse values for lasso (0.037 MSE, 0.474 R^2, 0.115 MAE), suggesting possible multicollinearity among the independent variables or overfitting.

Regarding the Decision Tree, despite values of 0.03 MSE, 0.123 MAE, and 0.581 R^2, it still performs lower than others.

Instead, the KNN obtained very good values: with 0.018 MSE, 0.100 MAE, and 0.741 R^2, it gets the best performance among the applied models.

In Figure 5.1 there is a graphic visualization of MSE and R^2 score, meanwhile in Figure 5.2 there is a graphic visualization of MAE for each model.

The results of the first two layers of the decision tree (Figure 4.3) suggest that "loudness" and "acousticness" are important variables in predicting genre. Songs with low loudness and low acoustics tend to have a lower target value than songs with medium or high loudness and high acoustics.

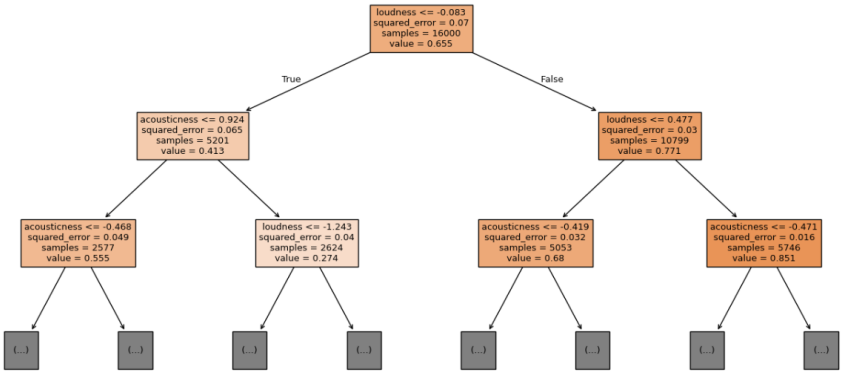

Figure 5.3: First two layers of Decision Tree

5.2 Pattern mining

Before starting the pattern mining phase, we prepared the data. First, removing "name", "album-name" e "artists". Then, we discretized the continuous values using the percentile technique to select the optimal number of bins for each feature (Figure 5.4).

Quantile	duration_ms	popularity	danceability	energy	loudness	speechiness	acousticness	instrumentalness	liveness	valence	tempo	n_beats
0	8.586,00	0,00	0,00	0,00	-49,53	0,00	0,00	0,00	0,00	0,00	0,00	0.0
1	196.901,33	0,19	0,469	0,56	-9,26	0,04	0,03	0,03	0,11	0,27	109,61	372.0
2	265.330,33	0,37	0,66	0,83	-5,79	0,07	0,42	0,42	0,21	0,58	134,99	560.0
3	4.120.258,00	1,00	0,98	1,00	5.36	0,94	1,00	1,00	16.34	0,99	220,53	7348.0

Figure 5.4: Percentile technique on selected data

Finally, we mapped the categorical features to the actual values, for example, by substituting each key value for the corresponding Pitch Class Notation value (0 represents C pitch, 1 represents C♯, and so on).

5.2.1 Frequent Patterns

Apriori

During this phase, we are looking for some item set really meaningful for our goal. We find out that using a minimum number of item set less than 3 with a minimum support of 10% the item sets found rappresents about some basic information that we are not interested in. For example, the pattern [4_signature, Non_Explicit] with a support of 82% is meaning less for us given that, both class of the feature, taken single, have a support greater than 87% in dataset. We also noticed that closed type results are equal to frequent in our case.

We tried different values of support from 1 to 30 and values of zmin (minimum number of items for item set) from 1 to 5. We simplified the visualization in Figure 5.5 reporting the test for support from 1 to 10 and for zmin from 3 to 6.

Figure 5.6 shows the items sets obtained at varying of supp parameter even for frequent and maximal types.

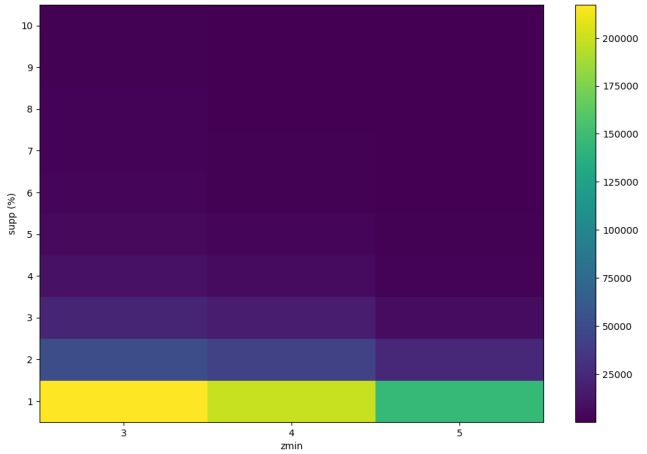

Figure 5.5: Item set obtained varying supp and zmin parameters

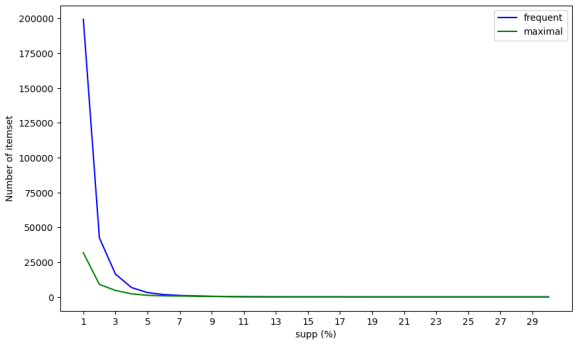

Figure 5.6: Varying support parameter for frequent and maximal

Hence, the parameter we decide to take in consideration are $min\_sup = 10\%$ and $z\_min = 4$. Running the algorithm we noticed that, both frequent and closed types given 260 patterns with 22% of max support, maximal type, instead given us 228 patterns with 16% as highest support. In the Figure 5.7 we can observe the highest item sets we found for the different Apriori types.

We can see that frequent and closed type give the same result, but 4_signature and Non_Explicit are even in all the high supported sets. In addition, as we can also see the presence of duration_ms but with two different items, with a time between 3 and 4 minutes, and between 4 minutes and 1 hour.

Item set type	Item set	Support (%)
Frequent	(n_beats_[560.0-7348.0], duration_ms_[265330.333-4120258.0], 4_signature, Non_Explicit)	22.17
	(energy_[0.832-1.0], loudness_[-5.791-3.156], 4_signature, Non_Explicit)	18,79
Closed	(n_beats_[560.0-7348.0], duration_ms_[265330.333-4120258.0], 4_signature, Non_Explicit)	22.17
	(energy_[0.832-1.0], loudness_[-5.791-3.156], 4_signature, Non_Explicit)	18,79
Maximal	(danceability_[0.656-0.98], valence_[0.576-0.995], 4_signature, Non_Explicit)	16,09
	(n_beats_[372.0-560.0], duration_ms_[196901.333-265330.333], 4_signature, Non_Explicit)	15,65

Figure 5.7: Highest support item sets for Apriori algorithm execution for different types

FP-Growth

For this section we used the same approach used for Apriori, but the result is the same even for the application of type frequent, closed and maximal.

5.2.2 Association Rules

The goal is to find the best set of rule with high confidence and lift values. To achieve that, we run the Apriori algorithm with different type of value for confidence and support, within a range from 10 to 30 for support and from 50 to 90 for confidence. We can take a look at the result in Figures 5.8 and 5.9.

We would choose a significant value of confidence and support looking at the graphs, without cut out relevant information and neither have a lot of insignificant rules. We choose to run the algorithm with 75% of confidence and 12.5% as minimum support. It results in 214 rules (Figure 5.10), and we show the five rules with the highest lift value.

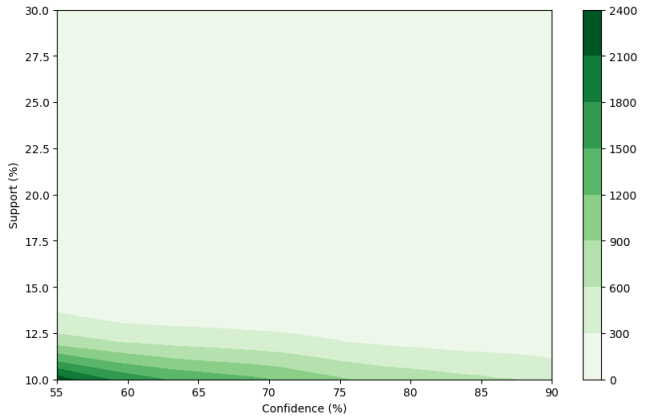

Figure 5.8: Number of association rules found varying support and confidence

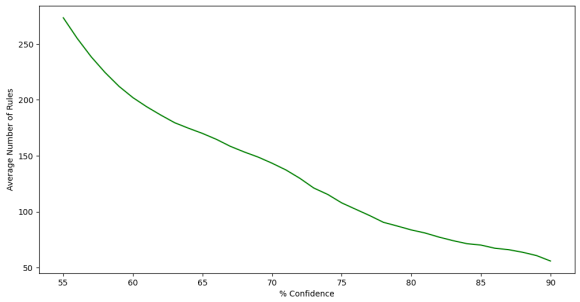

Figure 5.9: Average of association rules found for each value of confidence

consequent	antecedent	abs_support	%_support	confidence	lift
n_beats_[0.0-372.0]	(duration_ms_[8586.0-196901.333], tempo_[0.0-109.613], Non_Explicit)	2593	12,96	0,99	2,98
duration_ms_[265330.333-4120258.0]	(n_beats_[560.0-7348.0], instrumentalness_[0.377-1.0], Non_Explicit)	2291	11,45	0,9	2,70
energy_[0.0-0.56]	(loudness_[-49.531--9.263], acousticness_[0.416-1.0], 4_signature, Non_Explicit)	2545	12,75	0,89	2,68
energy_[0.0-0.56]	(loudness_[-49.531--9.263], acousticness_[0.416-1.0], Non_Explicit)	3526	17,63	0,89	2,68
energy_[0.0-0.56]	(loudness_[-49.531--9.263], acousticness_[0.416-1.0], 4_signature)	2574	12,87	0,89	2,68

Figure 5.10: Association rules sorted by lift parameter

Although the best lift is 2.98 we identify as most important rule the fourth of the table: it has a high support value and it appears in 3526 rows. It could be rappresented by the sentence: "A song with relative low loudness with high acousticness and non-explicit, it will probably have a low energy".