UNIVERSITÀ DI PISA

Data Mining II

Project 2023/2024

Exploration of Spotify tracks dataset

Nicola Pastorelli (656431), Salvatore Ergoli (660527), Marco Sanna (660521)

Contents

1 Data understanding and preparation

For this project, there are two tabular datasets (`tracks.csv` and `artists.csv`) and one folder containing the time series. The first tabular dataset has 109,547 rows and 34 features, the second one has 30,141 rows and 5 features. Instead, inside the folder there are a total of 10,000 separated time series, evenly distributed among 20 genres, that we gathered inside a `.csv` dataset. By comparing the ID of both the tracks and the time series, we found that every time series corresponds to a song in the track tabular dataset.

1.1 Tabular datasets

1.1.1 Track dataset and additional features

As said before, the `tracks.csv` dataset has a total of 109,547 data point and 34 properties. The *genre* variable in this dataframe includes a total of 114 genres, more than half of which are perfectly balanced with 1000 records for each genre.

From the `tracks.csv` dataframe, all features considered irrelevant and those redundant (highly positively or negatively correlated) have been removed, for a total of 13 features.

In order to create new features and extract additional information among the tracks, five new features have been added to the dataframe:

- *season* (categorical), indicating the season of the song's release. This feature has been created by looking at the album_release_date_precision column, but it is necessary to consider that there are 5856 tracks with only the year of publication. Therefore, since there was not way to approximate this value, we decided to remove the rows with the missing information;
- *decade* (categorical), indicating the decade of the song's release (12 decades in total, starting from 1899-1909 to 2020-2029);
- *months_from_publication* (integer), indicating the number of months from the song's release date to the time the dataset has been created, September 2023. Since we removed the rows with only the year of publication, it was possible to obtain a precise value for every song;
- *century* (binary), indicating whether the song was released in the 21st or 20th century;
- *single_artist* (binary), indicating whether the song is a collaboration or not.

For the data understanding phase, it has been decided to show the distribution of the newly added variables.

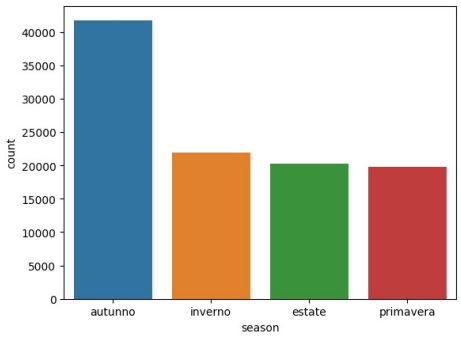


Figure 1: Distribution of season variable

Season

Starting with *season* variable, it can be observed that the season with a clear majority is autumn. However, it is important to consider that the songs were released in countries belonging to different hemispheres. This means that when it is autumn in the Northern Hemisphere, it is spring in the Southern Hemisphere, and vice versa. Consequently, the season variable may reflect different times of the year depending on the hemisphere where the songs were released. Unfortunately, we do not have any geographical information about the country where the track has been produced, therefore we consider the publication seasons with reference to only the Northern Hemisphere.

Decade and months from publication

The variables *decade* and *months_from_publication* both have positively skewed distributions. This skewness can be explained by the nature of the Spotify dataset.

For *decade*, there are more songs from recent decades. This could be because music production and release have increased over time, and Spotify, being a modern platform, has a larger collection of recent music compared to older tracks.

For *months_from_publication*, the skewness might be due to improvements in record-keeping over the years. In the past, release dates might not have been recorded as precisely as they are now. With the advent of digital distribution and better data management, more recent albums have very precise release dates, while older albums might only have the release year or even just the decade noted.

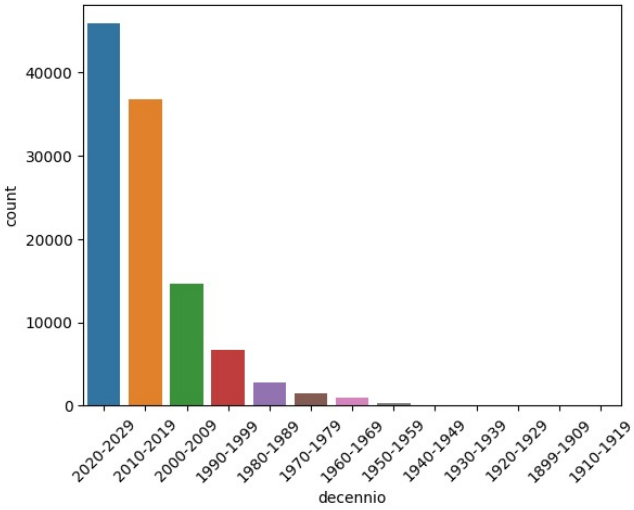


Figure 2: Distribution of decade variable

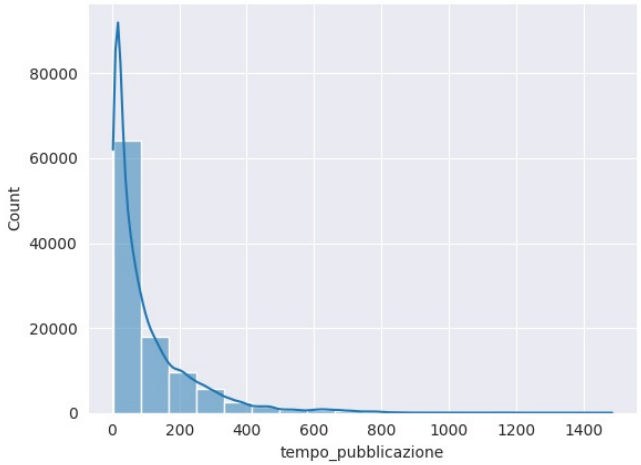


Figure 3: Distribution of months from publication variable

Century and single artist

For the variable *century*, there are almost 100,000 songs released in the 21st century and just over 10,000 songs released in the 20th century.

For the variable *single_artist*, the ";" symbol was checked in each row of the *artists* column to denote multiple artists, indicating a "featuring" scenario. There are a total of 27,908 songs where multiple artists are featured (label 1 in Figure 5), which is significantly less than the 80,000 songs by solo artists (label 0).

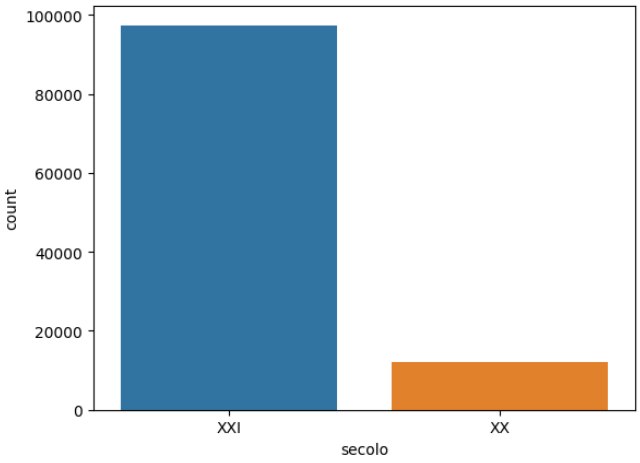


Figure 4: Distribution of century variable

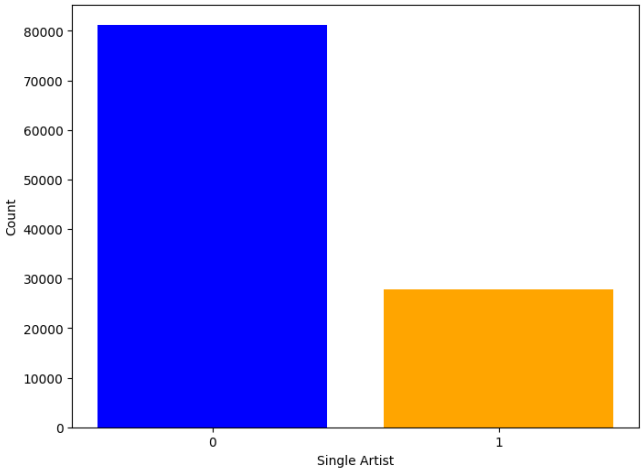


Figure 5: Distribution of single artist variable

1.1.2 Artists dataset

The columns of the `artists.csv` dataframe are as follows:

- *id*;
- *name*;
- *popularity*;
- *followers*;
- *genres*.

1.1.3 Merged dataset

Considered that the majority of the artists are comprehended in both datasets, we decided to merge these dataframes by taking only the common artists and using the artist column as the key for the merge. To allow merging, all homonymous artists have been removed from the `artists.csv` dataframe (518 in total). Removing homonymous artists ensures that each artist is uniquely identifiable, preventing any potential conflicts or ambiguities during the merging process with the `tracks.csv` dataframe.

This resulting dataset has 80256 rows $\times$ 38 columns. However, since the distribution of the genres has become very unbalanced, we decided to not use this dataset for future analyses.

1.2 Time series

The time series have been placed inside a .csv dataframe, where every row is a different series. We also added additional columns for keeping information about the series ID, genres and an encoded version of the genre as an integer value, with respect to the genres alphabetical order. Every time series has 1280 observations.

Each time series shows the variation of the spectral centroid of a song over time. As the spectral centroid is the weighted average of the frequencies, each time series shows the variation of the frequencies of a song over time. The time series are divided into 20 genres, to each of which exactly 500 time series belong. For each genre, it was decided to represent the average time series. The result consists of 20 time series, each representing the frequency variation over time typical of each genre. However, before representing the average time series of the genres, the time series in the dataset were pre-processed with various transformations and approximations.

Initially, 4 datasets were prepared, 3 datasets for subsequent clustering and classification with dynamic time warping and 1 dataset for clustering and classification with Euclidean distance. In all 4 datasets, the time series were normalised with Min-Max Scaler and noise was removed with moving average calculation; therefore the number of observations was reduced to 1271 due to the use of the moving average. Furthermore, in the Euclidean distance dataset, the time series were approximated with the Discrete Fourier Transform (DFT); this approximation reduces the observations to a total of 635 (1271/2 coefficients). With regard to the three datasets for dynamic time warping operations, Symbolic Aggregate Approximation (SAX) was used in one dataset, while Piecewise Aggregate Approximation (PAA) was used in the other two datasets; the latter two datasets differ in that the offset translation transformation was used in one dataset, while it was not used in the other. These approximations reduce the observations to a total of 100.

Of the three dynamic time warping datasets, the pre-processed dataset with PAA without offset translation is the one that offers a better representation of the average time series of the genres. In fact, in this dataset, the average time series of the genres are well separated from each other and, consequently, are well interpretable (Figure 6). On the contrary, in the dataset pre-processed with PAA and offset translation and in the dataset pre-processed with SAX, the average time series of the genres tend to overlap and, consequently, are less interpretable.

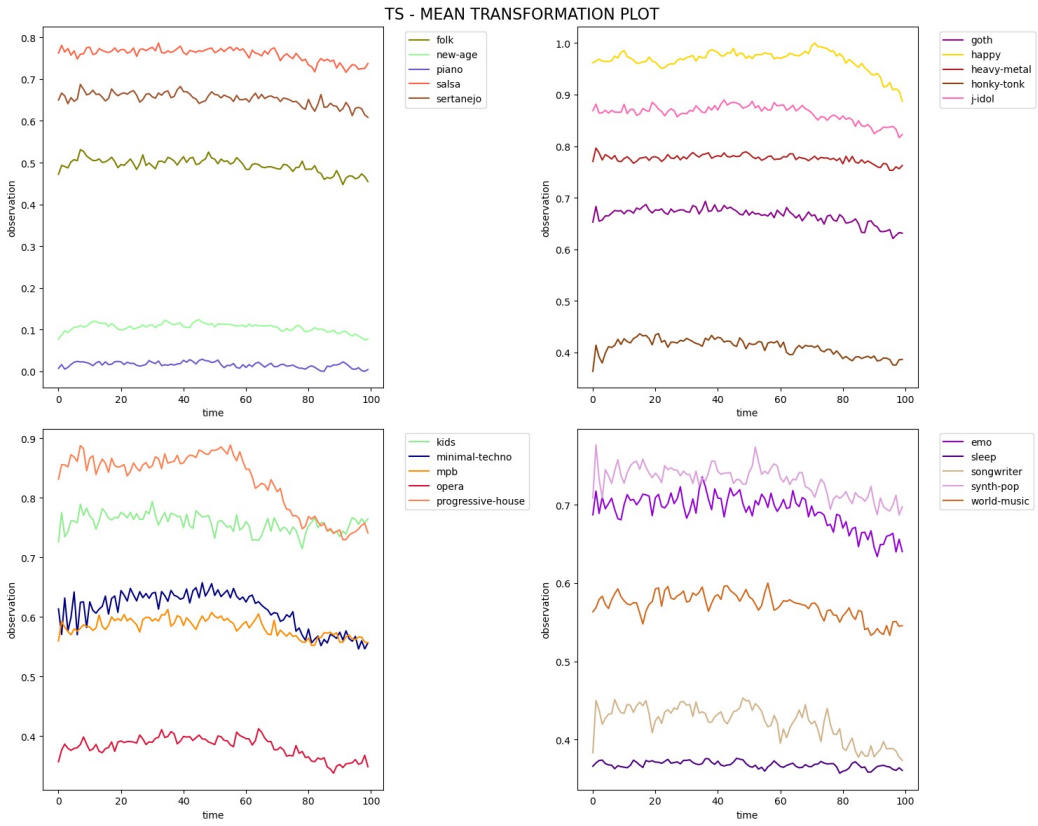


Figure 6: Time series plot with PAA approximantion and MinMax normalization

2 Time Series Analysis

2.1 Clustering

2.1.1 K-means

The first clustering algorithm that was performed on the time series is k-means. The distance that has been used with this algorithm is dynamic time warping. The choice of this distance is due to the fact that the k-means algorithm has a low computational cost, which is why it is possible to use it with the dynamic time warping distance, which is computationally more expensive than the Euclidean distance but gives generally better results.

Although k-means has a low computational cost, it was chosen to reduce the computational cost of the dynamic time warping distance by using the Sakoe-Chiba Band global constraint with radius equal to 1. In addition, a number of clusters k equal to 2 was chosen; in fact, as will be seen later, this is the number of clusters that shows the best distribution of genres.

Having chosen to use k-means with dynamic time warping, the algorithm was run on the 3 pre-processed datasets for dynamic time warping operations. In the pre-processed dataset with PAA and offset translation and in the pre-processed dataset with SAX, the two clusters identified do not have a well-defined distribution of genres; in other words, there are no prevailing genres in one cluster or the other.

In contrast, in the dataset pre-processed with PAA without offset translation, the two identified clusters have a well-defined distribution of genres. In fact, as can be seen in figure 7, the genres j-idol, heavy-metal, happy, sertanejo, salsa and progressive-house have a clear prevalence in cluster 0; on the contrary, the genres piano and new-age have a clear prevalence in cluster 1. The distribution of genres in the two clusters is reasonable when looking at figure 8, where the average time series of the two clusters are shown. As can be seen in the figure, the average time series of cluster 0 shows higher frequencies than the average time series of cluster 1. In fact, the j-idol, heavy-metal, happy, sertanejo, salsa and progressive-house genres prevalent in cluster 0 have higher frequencies, while the piano and

new-age genres prevalent in cluster 1 have lower frequencies.

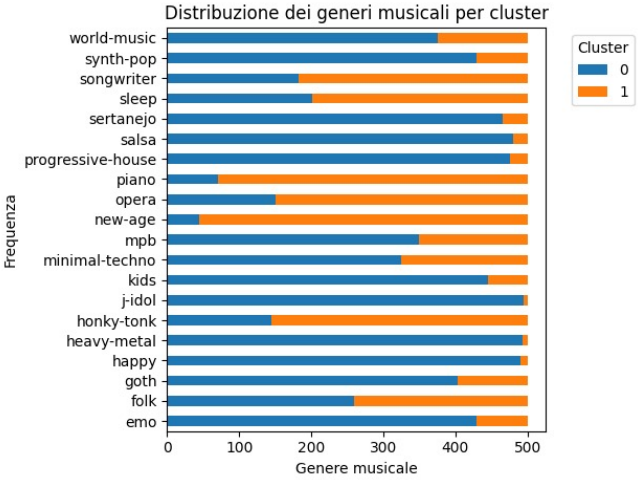


Figure 7: Distribution of genres in k-means clusters

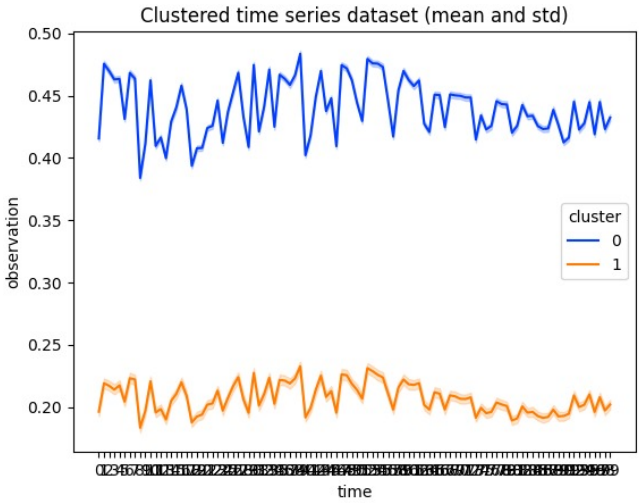


Figure 8: Average time series of k-means clusters

The two identified clusters were further visualised using two dimensionality reduction techniques: Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). In both cases, it was decided to reduce the initial dimensions to 2 dimensions; in this way, it was possible to represent the clusters in a scatterplot whose x and y axes are the two dimensions with the maximum variance. In particular, in figure 9 it is possible to observe the clusters in the two dimensions found with PCA: the two clusters have similar values in the second dimension, whereas in the first dimension, cluster 0 has values between -0.4 and -0.2 and cluster 1 has values between 0.6 and 0.8. Figure 10 shows the clusters in the two dimensions found with SVD: this time, the two clusters have similar values in the first dimension, while in the second dimension, cluster 0 has values between -0.75 and -0.25 and cluster 1 has values between 0.75 and 1.25.

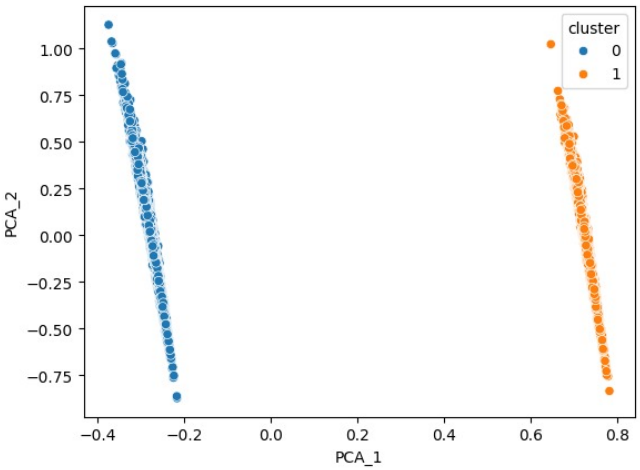


Figure 9: K-means clusters visualized with PCA

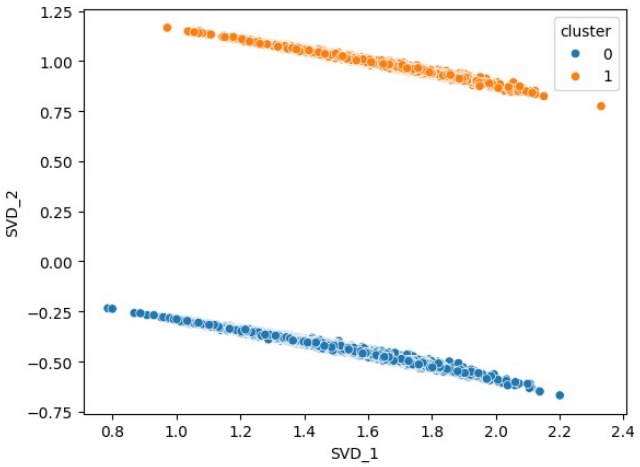


Figure 10: K-means clusters visualized with SVD

2.1.2 Hierarchical

The second clustering algorithm that was performed on the time series is agglomerative hierarchical clustering. The distance that was used with this algorithm is the Euclidean distance. The choice of this distance is due to the fact that agglomerative hierarchical clustering has a high computational cost, which is why it is appropriate to use it with the Euclidean distance, which is computationally less expensive than the dynamic time warping distance.

Having chosen to use hierarchical clustering with Euclidean distance, the algorithm was run on the dataset approximated with DFT. To obtain clusters comparable to the clusters obtained with k-means, the Ward method was chosen with a number of clusters k equal to 2. In figure 11, the distribution of genres in the two clusters can be observed, while figure 12 shows the average time series of the two clusters. As can be seen in figure 11, the sleep, piano and new-age genres prevail in cluster 1. In fact, as can be seen in figure 12, the average time series of cluster 1 shows lower frequencies than the average time series of cluster 2. However, as can be seen in figure 11, there are no prevailing genres in cluster 2. This leads to the conclusion that the two clusters found with k-means have a sharper genres distribution than the two clusters found with hierarchical clustering.

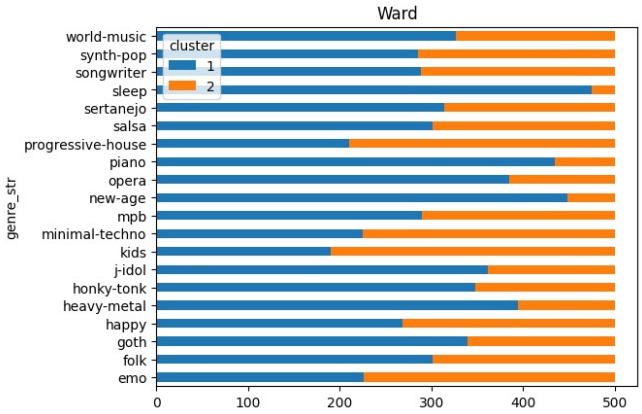


Figure 11: Distribution of genres in hierarchical clusters

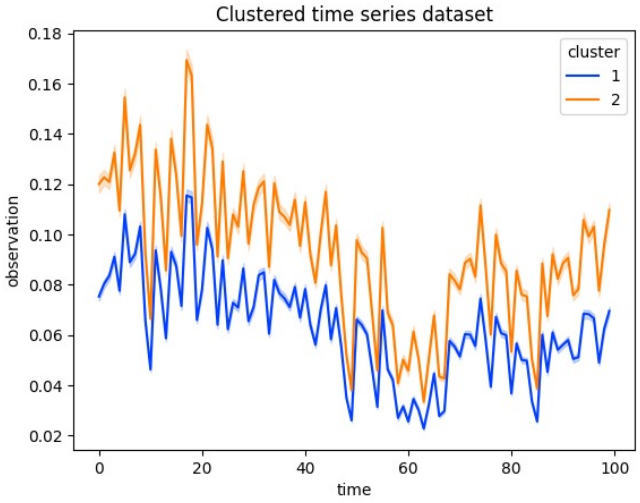


Figure 12: Average time series of hierarchical clusters

Once again, the two identified clusters were visualised using PCA and SVD with two dimensions. In Figure 13 and Figure 14, one can observe the clusters in the two dimensions with maximum variance found with PCA and SVD, respectively. In both the visualisation with PCA and the visualisation with SVD, the two clusters have similar values in the second dimension, whereas in the first dimension, cluster 1 has low values and cluster 2 has high values.

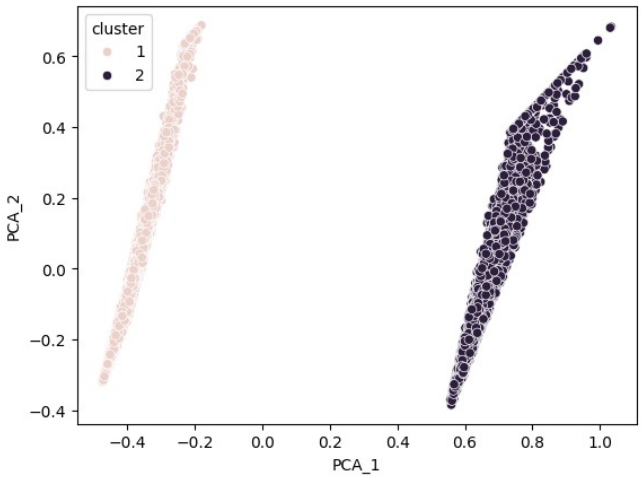


Figure 13: Hierarchical clusters visualized with PCA

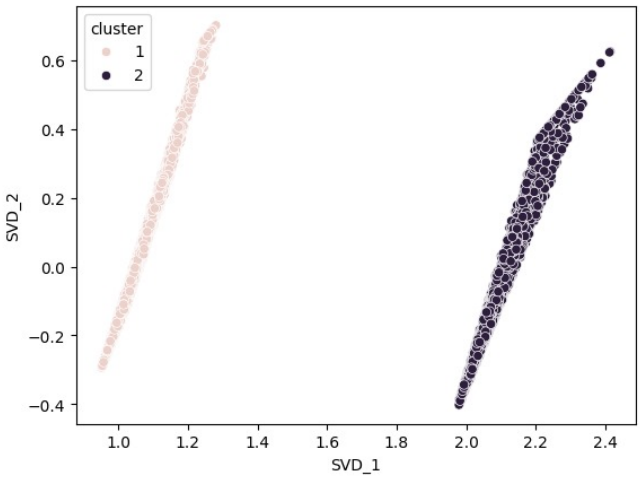


Figure 14: Hierarchical clusters visualized with SVD

2.2 Classification

The classification task that was defined is the genres classification of the time series. To solve this task, the dataset consisting of 10000 time series was divided into a training set containing 8000 time series and a test set containing 2000 time series.

2.2.1 KNN

To solve the classification task that was defined, the KNN algorithm with dynamic time warping was initially used. To reduce the computational cost of the dynamic time warping distance, the Sakoe-Chiba Band global constraint with radius equal to 1 was used. In addition, a number k of nearest neighbours equal to 1 was chosen.

Since the KNN had to be used with dynamic time warping, the algorithm was run on the three pre-processed datasets for dynamic time warping operations. In the pre-processed dataset with SAX, an accuracy on the test set of 0.13 was achieved. This is an unsatisfactory score, as it is slightly higher than the random value of 0.05 (1/20 classes). The same accuracy value was obtained in the pre-processed dataset with PAA and offset translation. In contrast, in the pre-processed dataset with PAA without offset translation, an accuracy on the test set of 0.21 was achieved (figure 15). This is a modest score, but more satisfactory than the previous results.

In a second step, the KNN with Euclidean distance was used. Having to use the KNN with the Euclidean distance, the algorithm was run on the dataset approximated with DFT. In order to have a result comparable with previous results, once again a number k of nearest neighbours equal to 1 was chosen. In this case, the accuracy value obtained on the test set is 0.23 (figure 16). This is an unexpected result, as it is slightly higher than the best score obtained with dynamic time warping (0.21). This may be due to the fact that our time series may be quite aligned; it is in fact in cases where the time series are misaligned that dynamic time warping works better than Euclidean distance.

	precision	recall	f1-score	support
emo	0.13	0.10	0.11	100
folk	0.10	0.08	0.09	100
goth	0.08	0.09	0.09	100
happy	0.24	0.41	0.30	100
heavy-metal	0.12	0.16	0.14	100
honky-tonk	0.24	0.24	0.24	100
j-idol	0.15	0.24	0.18	100
kids	0.20	0.12	0.15	100
minimal-techno	0.22	0.19	0.20	100
mpb	0.10	0.06	0.07	100
new-age	0.35	0.29	0.32	100
opera	0.20	0.18	0.19	100
piano	0.44	0.37	0.40	100
progressive-house	0.28	0.30	0.29	100
salsa	0.07	0.07	0.07	100
sertanejo	0.25	0.23	0.24	100
sleep	0.39	0.60	0.47	100
songwriter	0.23	0.18	0.20	100
synth-pop	0.20	0.16	0.18	100
world-music	0.12	0.11	0.11	100
accuracy			0.21	2000
macro avg	0.21	0.21	0.20	2000
weighted avg	0.21	0.21	0.20	2000

Figure 15: KNN with dynamic time warping distance

	precision	recall	f1-score	support
emo	0.19	0.18	0.19	100
folk	0.04	0.03	0.04	100
goth	0.07	0.07	0.07	100
happy	0.52	0.33	0.40	100
heavy-metal	0.22	0.14	0.17	100
honky-tonk	0.13	0.18	0.15	100
j-idol	0.29	0.28	0.29	100
kids	0.11	0.05	0.07	100
minimal-techno	0.62	0.47	0.53	100
mpb	0.16	0.13	0.14	100
new-age	0.24	0.30	0.27	100
opera	0.12	0.29	0.17	100
piano	0.30	0.32	0.31	100
progressive-house	0.34	0.32	0.33	100
salsa	0.57	0.32	0.41	100
sertanejo	0.17	0.18	0.18	100
sleep	0.46	0.53	0.49	100
songwriter	0.11	0.14	0.12	100
synth-pop	0.28	0.22	0.25	100
world-music	0.15	0.19	0.17	100
accuracy			0.23	2000
macro avg	0.25	0.23	0.24	2000
weighted avg	0.25	0.23	0.24	2000

Figure 16: KNN with euclidean distance

2.2.2 MiniRocket

	precision	recall	f1-score	support
emo	0.10	0.07	0.08	100
folk	0.07	0.02	0.03	100
goth	0.04	0.01	0.02	100
happy	0.25	0.65	0.36	100
heavy-metal	0.13	0.06	0.08	100
honky-tonk	0.12	0.09	0.10	100
j-idol	0.25	0.09	0.13	100
kids	0.12	0.13	0.12	100
minimal-techno	0.31	0.71	0.43	100
mpb	0.03	0.01	0.02	100
new-age	0.19	0.42	0.26	100
opera	0.16	0.13	0.14	100
piano	0.15	0.13	0.14	100
progressive-house	0.32	0.19	0.24	100
salsa	0.11	0.19	0.14	100
sertanejo	0.16	0.19	0.18	100
sleep	0.28	0.25	0.27	100
songwriter	0.13	0.06	0.08	100
synth-pop	0.21	0.26	0.24	100
world-music	0.14	0.13	0.14	100
accuracy			0.19	2000
macro avg	0.16	0.19	0.16	2000
weighted avg	0.16	0.19	0.16	2000

Figure 17: Classification results with MiniRocket method

The same classification task was performed using the MiniRocket method, which was preferred to the Rocket method as it is much faster and more accurate. First, MiniRocket was used to transform the time series into useful features using random convolutional kernels. The transformed features were then used to train a linear classifier, namely the Ridge classifier. The classifier's prediction on the test set achieved an accuracy of 0.19 (Figure 17). This is a score in line with previous results.

2.2.3 Shapelets

The classification process based on shapelets was divided into the four steps shown in Figure 18. First, the shapelets were extracted with the Random Shapelet Transform method: as the extraction of shapelets is a computationally expensive process, it was decided to extract shapelets only from the first 3000 time series ('shapelet finder'). After extracting the shapelets, the time series were transformed into distances between the time series and the extracted shapelets ('shapelet transformer'). In this way, the original dataset was transformed into a dataset in which each record is a vector of distances between a time series and each shapelet ('shapelet dataset'). Finally, the KNN classifier was trained on the transformed dataset by choosing a number k of nearest neighbours equal to 1 ('ML model'). The classifier's prediction on the test set achieved an accuracy of 0.10 (figure 19). This is an unsatisfactory score, as it is slightly higher than the random value of 0.05. This may be due to the fact that the shapelets extracted at the beginning have a low information gain, so they are not very representative of the genres of the time series.

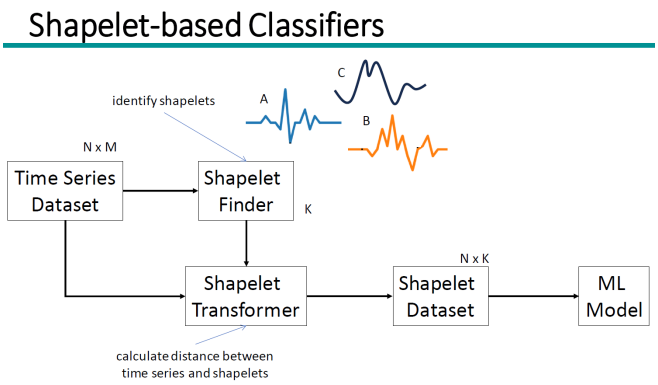


Figure 18: Shapelets-based classification process

	precision	recall	f1-score	support
emo	0.07	0.07	0.07	15
folk	0.12	0.13	0.13	15
goth	0.07	0.07	0.07	15
happy	0.21	0.20	0.21	15
heavy-metal	0.06	0.07	0.06	15
honky-tonk	0.16	0.20	0.18	15
j-idol	0.05	0.07	0.06	15
kids	0.07	0.07	0.07	15
minimal-techno	0.40	0.27	0.32	15
mpb	0.00	0.00	0.00	15
new-age	0.10	0.07	0.08	15
opera	0.14	0.13	0.14	15
piano	0.00	0.00	0.00	15
progressive-house	0.20	0.13	0.16	15
salsa	0.00	0.00	0.00	15
sertanejo	0.00	0.00	0.00	15
sleep	0.24	0.33	0.28	15
songwriter	0.06	0.07	0.06	15
synth-pop	0.07	0.07	0.07	15
world-music	0.00	0.00	0.00	15
accuracy			0.10	300
macro avg	0.10	0.10	0.10	300
weighted avg	0.10	0.10	0.10	300

Figure 19: Shapelets-based classification results

After classifying according to shapelets, it was decided to identify - among the shapelets extracted at the beginning

- the most representative shapelets of the 20 genres of the time series. To do this, the shapelet with the highest information gain of each genre, i.e. the most representative shapelet of that genre, was selected. In this way, 20 shapelets were obtained, one for each genre. Subsequently, these shapelets were visualized on the time series from which they were extracted. They were then compared graphically with motifs and anomalies; they will then be illustrated in the following section.

2.3 Motifs/discords

At this stage of the project, it was decided to extract motifs and discords from the same time series from which the most representative shapelets of the genres were extracted. The aim was to find out whether shapelets are more similar to motifs or discords, i.e. whether patterns or anomalies are more representative of genres. To do this, we first selected the 20 time series from which the most representative shapelets of the genres were extracted. Next, the shapelet was visualized on each time series. Finally, exactly one motif and one anomaly were extracted from each time series using a window of length 10.

In Figure 20, the time series of the happy, minimal-techno, new-age and sleep genres can be appreciated. For the happy genre, the shapelet intersects more with the motif than with the anomaly, suggesting that the pattern is more representative of the happy genre than the anomaly. With regard to the minimal-techno genre, the shapelet intersects with the motif and the anomaly in equal measure, suggesting that the pattern and the anomaly are both representative of the minimal-techno genre. With regard to the new-age genre, the shapelet only intersects with the motif and not with the anomaly; this suggests that the new-age genre is well discriminated by the pattern while it is not discriminated by the anomaly. Finally, with regard to the sleep genre, the shapelet intersects more with the anomaly than with the motif; this suggests that the anomaly is more representative of the sleep genre than the pattern.

In conclusion, it is difficult to determine whether shapelets are more similar to motifs or discords, i.e. whether patterns or anomalies are more representative of genres. However, our analysis seems to suggest that shapelets are more similar to motifs and that therefore genres are better discriminated by patterns than by anomalies. Further analysis is however necessary to answer this question.

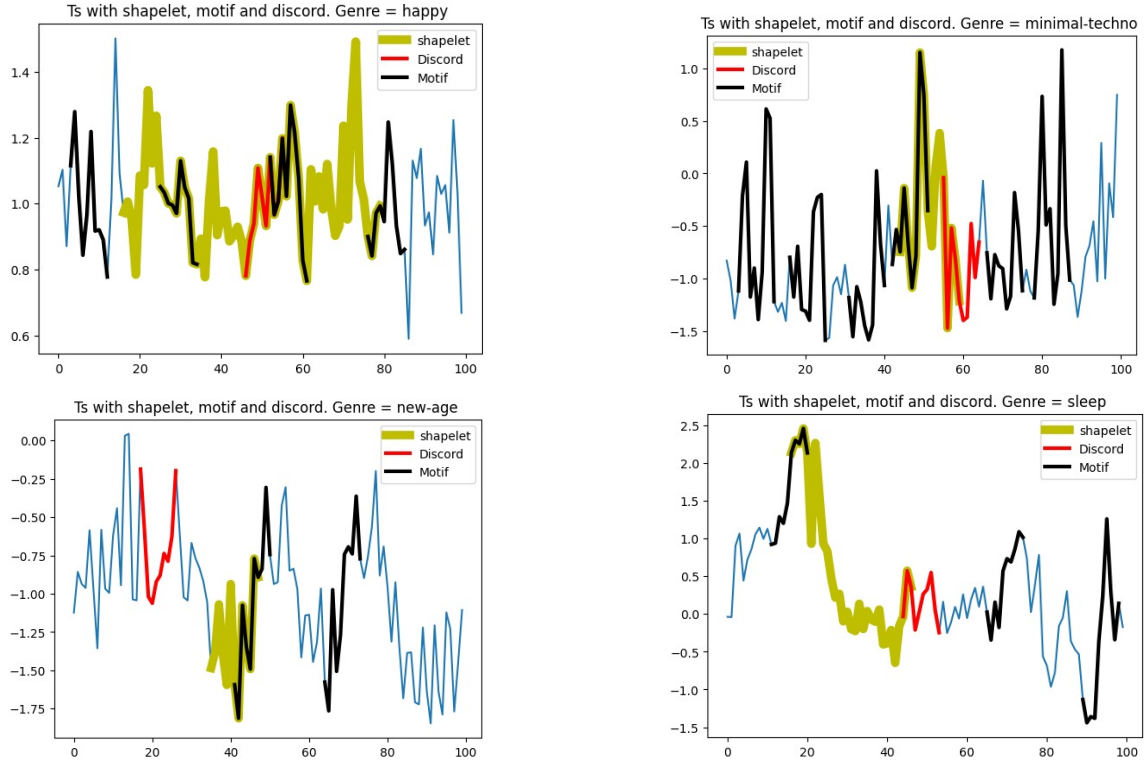


Figure 20: Relationship between shapelets, motifs and discords

3 Advanced Data-Preprocessing

3.1 Outlier detection

Outlier detection was performed on the 'track' dataset. Specifically, outliers were identified on the basis of the following variables: duration ms, popularity, danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo. Before proceeding with outlier detection, the values of these variables were normalised with Min-Max Scaler.

3.1.1 KNN

The first outlier detection approach used was a distance-based approach, the KNN. In the context of outlier detection, the KNN calculates the outlier score of a point as the distance to its k nearest neighbours. Although this approach normally assigns a score and not a label to outliers, it can also be used to assign a label to a point, which tells us whether that point is an outlier or not. It was decided to use this approach in the second way.

When training the algorithm, a number k of nearest neighbours equal to 2 was chosen and the contamination parameter was set to 0.1 to ensure that the outliers found were about 10% of the data. However, the prediction of the algorithm returned a number of inliers equal to 97788 and a number of outliers equal to 5903. In figure 21, the genres distribution among the outliers found can be seen. As can be seen in the figure, the genre containing the highest number of outliers is study. In figure 22, on the other hand, it is possible to observe the outliers found and a sample of 10000 inliers: the data are visualized in the two dimensions with maximum variance found with PCA. As can be seen in the figure, the inliers are more numerous in the first half of the 'rhombus', while the outliers are more visible in the second half.

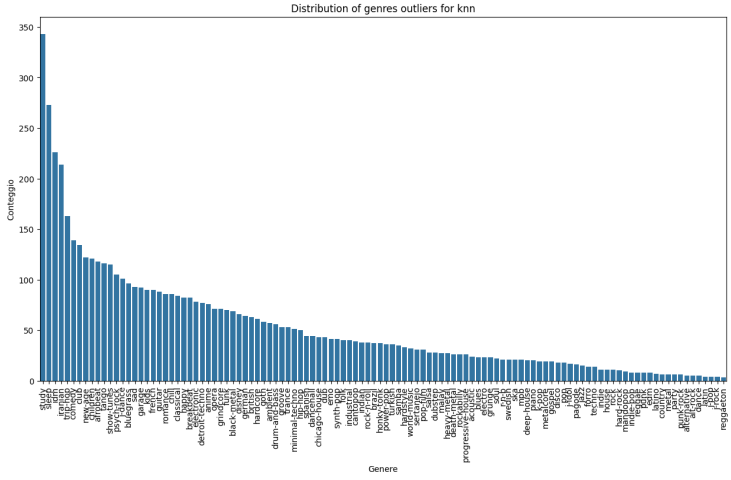


Figure 21: Genres distribution among KNN outliers

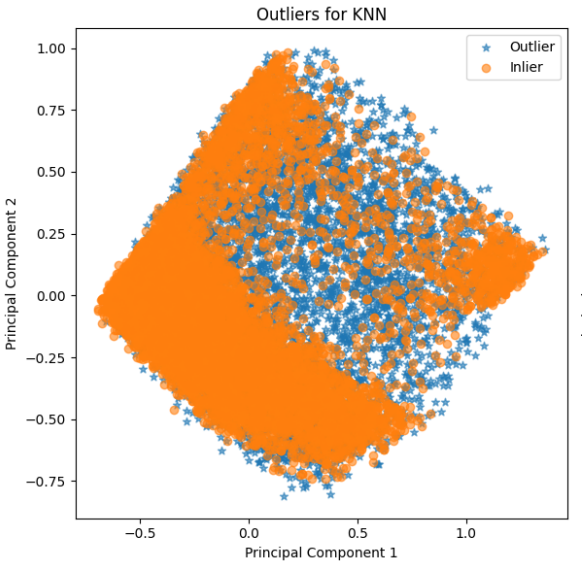


Figure 22: KNN outliers visualized with PCA

3.1.2 LOF

The second outlier detection approach used was a density-based approach, namely the LOF (Local Outlier Factor). The LOF calculates the outlier score of a point as its relative density, i.e. the density of that point compared to the density of its neighbours. Since points with a LOF value ≈ 1 are considered inliers and points with a LOF value $>$ 1 are considered outliers, also this approach was used to label points as inliers or outliers.

During algorithm training, the contamination parameter was again set to 0.1. This time, the prediction of the algorithm returned a number of inliers of 93322 and a number of outliers of 10369. The genres distribution among

the detected outliers is shown in figure 23: this time, the genre containing the highest number of outliers is sleep. Although the genres distribution is different from the one observed in the KNN outliers, the visualisation of the LOF outliers with PCA (figure 24) is very similar to the visualisation of the KNN outliers. This suggests the idea that, in our case, isolating outliers by density instead of distance does not change the composition of the detected outliers.

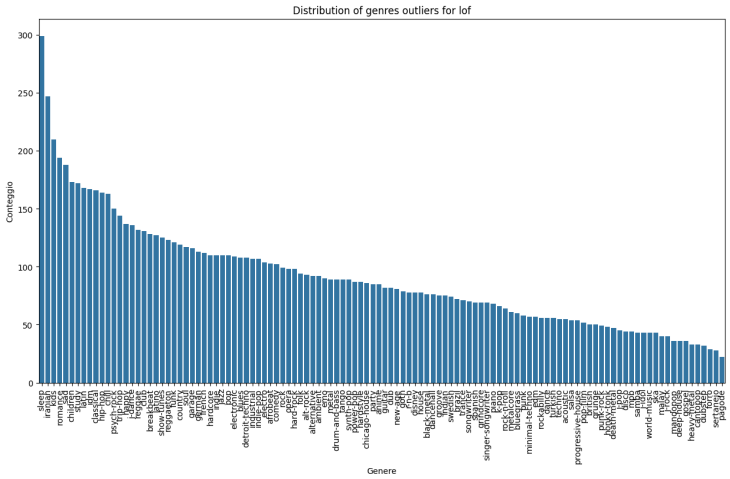


Figure 23: Genres distribution among LOF outliers

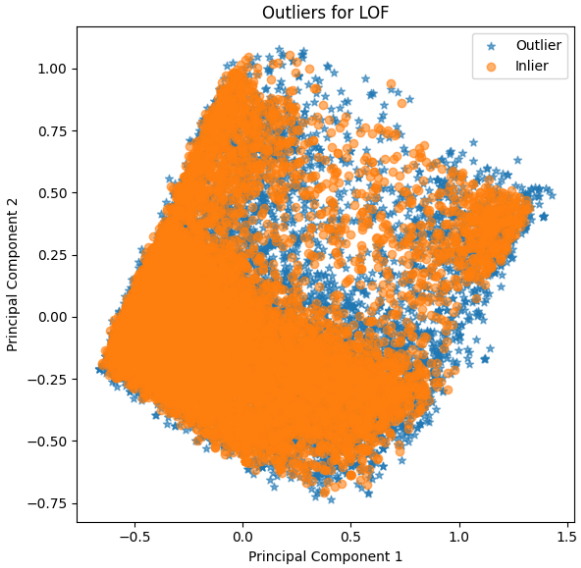


Figure 24: LOF outliers visualized with PCA

3.1.3 Isolation Forest

The third and last outlier detection approach used was a model-based approach, namely Isolation Forest. The idea behind this approach is that outliers can be isolated faster than inliers. For consistency with the approaches used before, also this approach was used not to score outliers, but simply to label points as inliers or outliers.

When training the model, once again the contamination parameter was set to 0.1. The model prediction returned a number of inliers of 93322 and a number of outliers of 10369. Thus, the number of outliers detected with Isolation Forest is equal to the number of outliers detected with LOF. However, the outliers in common - i.e. those detected with both approaches - are 5114, about half. Given the high number of outliers in common, the distribution of genres among the outliers detected with Isolation Forest (figure 25) is quite similar to the distribution of genres among the outliers detected with LOF: in both cases, the genre containing the highest number of outliers is sleep. Furthermore, the visualization of Isolation Forest outliers with PCA (figure 26) is similar to the visualization of LOF outliers and KNN outliers. This suggests the idea that the outliers in the dataset are so characteristic that isolating them with different methods belonging to different families does not change their composition.

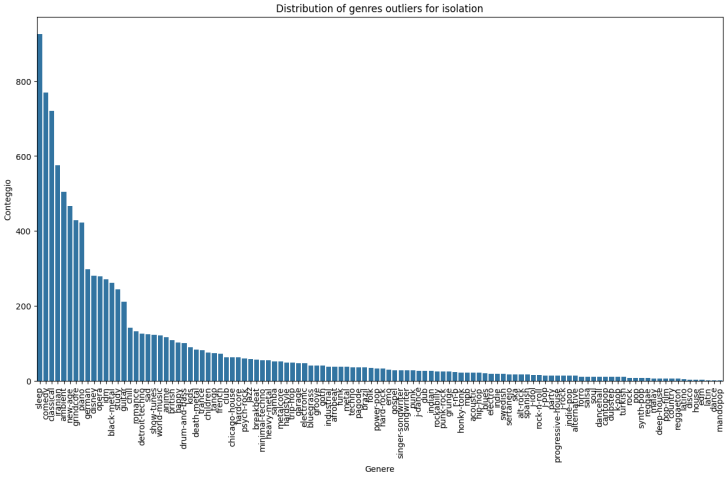


Figure 25: Genres distribution among Isolation Forest outliers

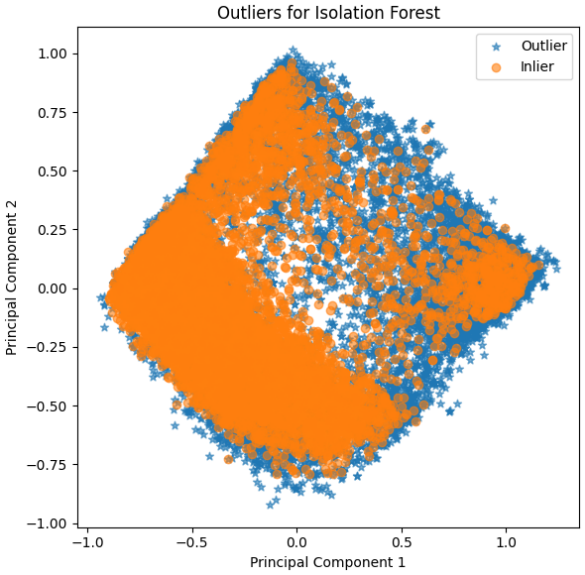


Figure 26: Isolation Forest outliers visualized with PCA

3.1.4 Top 1% outliers

After detecting outliers with KNN, LOF and Isolation Forest, it was decided to identify common outliers, i.e. those detected with all three approaches. There are 1421 outliers in common, i.e. approximately 1% of the data in the track dataset (precisely 1.37%). These outliers were considered the top 1% outliers. In fact, they were detected by three different methods belonging to three different families. Thus, although the three approaches were used to assign a label and not a score to the outliers, it is very likely that the outliers in common have a very high score, so that they can be considered the top 1% outliers.

Of the top 1% outliers, it was decided not to report the genres distribution and visualisation with PCA. In fact, since the outliers detected with KNN, LOF and Isolation Forest showed similar genres distribution and similar visualisation with PCA, it is obvious that the outliers in common show equally similar results. In particular, among the outliers in common the most numerous genre is again sleep. Moreover, their visualisation with PCA is in line with previous visualisations.

With regard to the treatment of the top 1% outliers, it was decided not to remove them from the dataset. Indeed, their removal would have slightly unbalanced the genres, resulting in a slight deterioration of the classification results on the genre variable.

3.2 Imbalanced learning

3.2.1 Unbalanced classification

The unbalanced classification task that was defined is the classification on the 'century' variable created in module 1. The two classes of this variable are very unbalanced. In fact, 94515 records have the 21st century as their class, while 9176 records have the 20th century as their class; in other words, the class 'XXI' contains about 91% of the records, while the class 'XX' contains about 9% of the records. Since the dataset is already unbalanced with respect to this variable, it was decided to leave it as it is, i.e. not to do any further unbalancing.

In order to solve the classification task that was defined, first of all the attributes of the dataset were divided into an attribute Y (the century variable) and a set of attributes X containing the following variables: duration ms, popularity, danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo. The values of the variables in X were normalised with Min-Max Scaler. In a second step, the dataset was divided into a training set containing 80% of the records and a test set containing 20% records. In figure 27, the distribution

of the century variable in the training set can be observed: the class 'XXI' contains 75656 records, while the class 'XX' contains 7296 records.

At this point, the Decision Tree algorithm was trained on the training set. When training the model, the weights of the classes were balanced in order to make the algorithm more sensitive to the rare class. However, the model's prediction on the test set achieved an accuracy of 0.90 (Figure 28). This is an unsatisfactory score as it is lower than the accuracy of a trivial classifier, which in this case would be 0.91. In light of the poor result, oversampling and undersampling techniques were used to balance the two classes.

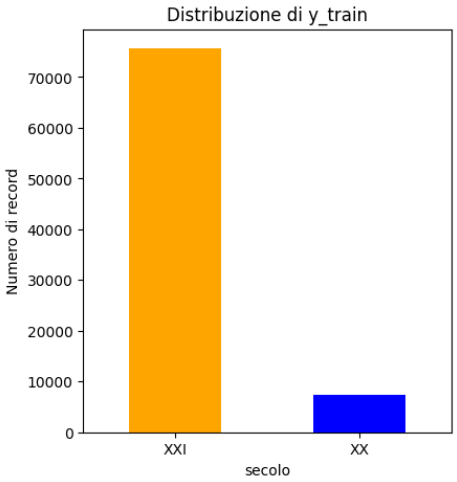


Figure 27: Distribution of century variable in training set

	precision	recall	f1-score	support
XX	0.45	0.44	0.45	1880
XXI	0.94	0.95	0.95	18859
accuracy			0.90	20739
macro avg	0.70	0.69	0.70	20739
weighted avg	0.90	0.90	0.90	20739

Figure 28: Unbalanced classification results

3.2.2 Oversampling

The first oversampling technique used was SMOTE. Through this technique, the training set was balanced with respect to the century variable. In particular, oversampling was done on the minority class, which thus achieved the same number of records as the majority class. Figure 29 shows the distribution of the century variable in the training set after using SMOTE: both the 'XXI' class and the 'XX' class now contain 75656 records.

At this point, the Decision Tree algorithm was trained on the balanced training set. The prediction of the model, which was done on the unbalanced test set, achieved an accuracy of 0.85 (Figure 30). This result is lower than the one obtained before (0.90). However, the minority class recall (0.62) is significantly higher than the one obtained before (0.44). The same classification results were obtained using the ADASYN oversampling technique. This makes sense, as ADASYN is a variant of SMOTE.

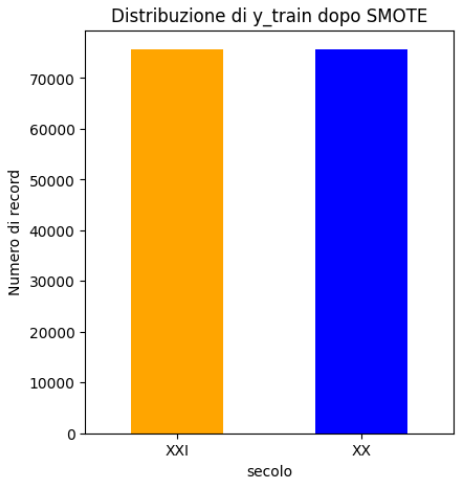


Figure 29: Distribution of century variable after SMOTE

	precision	recall	f1-score	support
XX	0.32	0.62	0.42	1880
XXI	0.96	0.87	0.91	18859
accuracy			0.85	20739
macro avg	0.64	0.74	0.67	20739
weighted avg	0.90	0.85	0.87	20739

Figure 30: Classification results after SMOTE

3.2.3 Undersampling

The undersampling technique used was Random Undersampling. Also through this technique, the training set was balanced with respect to the century variable. In particular, undersampling was done on the majority class, which in this way achieved the same number of records as the minority class. Figure 31 shows the distribution of the century variable in the training set after the use of Random Undersampling: both class 'XXI' and class 'XX' now contain 7296 records.

Again, the Decision Tree algorithm was trained on the balanced training set. This time the prediction of the model, which was made on the unbalanced test set, achieved an accuracy of 0.73 (figure 32). This score is lower than the one obtained using the SMOTE and ADASYN oversampling techniques.

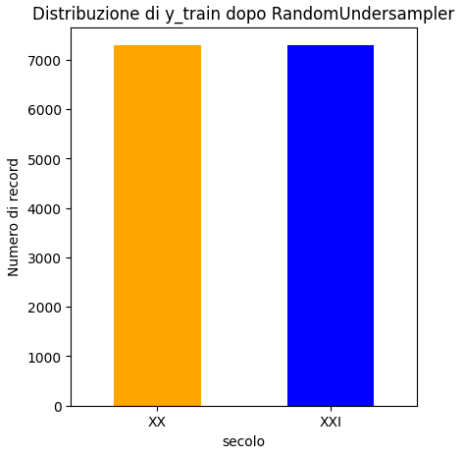


Figure 31: Distribution of century variable after Random Undersampling

	precision	recall	f1-score	support
XX	0.20	0.68	0.31	1880
XXI	0.96	0.73	0.83	18859
accuracy			0.73	20739
macro avg	0.58	0.71	0.57	20739
weighted avg	0.89	0.73	0.78	20739

Figure 32: Classification results after Random Undersampling

4 Advanced ML and XAI

4.1 Advanced classification

At this stage of the project, the classification task that was defined is the classification on the genre variable of the track dataset. To avoid a classification on 114 genres, it was decided to classify only the 20 genres in the time series dataset. The original dataset was therefore reduced to a dataset containing 19970 records, approximately 1000 records for each genre.

To solve the classification task that was defined, first the attributes of the dataset were divided into an attribute Y (the genre variable) and a set of attributes X containing the variables most correlated with the encoded genre variable: popularity, danceability, energy, loudness, speechiness, acousticness, tempo. The values of the variables in X were normalised with Standard Scaler. In a second step, the dataset was divided into a training set containing 90% of the records and a test set containing 10% of the records.

4.1.1 Logistic Regression

The first classification method that was used is Logistic Regression. Before training this algorithm on the training set, a grid search with 5 fold cross-Validation and shuffling enabled was performed to find the best model parameters. The parameters tested are:

- **'C'**: this parameter has been chosen in order to set the regularization of the weights: smaller the values, stronger the regularization. The best parameter found by the grid search is "10000". A randomized search was performed to further optimise this parameter: the best value found is 13119.9413 (table 1);
- **'Penalty'**: this parameter specifies the type of penalty term during the weights update phase (like Ridge or Lasso) and it depends on the solver that is selected. The grid search returned "L2" (Ridge regularization) as the parameter;
- **'Solver'**: this parameter specifies the type of the algorithm to use in the optimization problem. The "Newton-cg" type has been selected, thanks to its capability to handle multinomial loss for multiclass problems;
- **'Class_weight'**: this parameter specifies the weights associated with classes by using the same weight for all the classes or not. The grid search returned "None" (default value) as the best choice.

At this point, the model with the best parameters was used to make a prediction on the test set. The prediction of the model on the test set achieved an accuracy of 0.49 (figure 33). This is a good score, as it is almost 10 times higher than the random value of 0.05 (1/20 classes).

	precision	recall	f1-score	support
emo	0.31	0.30	0.30	94
folk	0.25	0.15	0.19	100
goth	0.21	0.07	0.10	119
happy	0.61	0.56	0.58	97
heavy-metal	0.45	0.61	0.52	105
honky-tonk	0.46	0.79	0.58	84
j-idol	0.46	0.57	0.51	99
kids	0.48	0.63	0.55	105
minimal-techno	0.79	0.90	0.85	115
mpb	0.45	0.20	0.28	101
new-age	0.67	0.62	0.65	106
opera	0.47	0.61	0.53	89
piano	0.62	0.37	0.47	94
progressive-house	0.44	0.39	0.42	99
salsa	0.44	0.70	0.54	89
sertanejo	0.64	0.53	0.58	89
sleep	0.77	0.83	0.80	106
songwriter	0.36	0.29	0.32	119
synth-pop	0.28	0.34	0.31	99
world-music	0.36	0.40	0.38	88
accuracy			0.49	1997
macro avg	0.48	0.49	0.47	1997
weighted avg	0.48	0.49	0.47	1997

Figure 33: Logistic Regression results

Parameter	Value
C	13119.9413
Penalty	L2
Solver	Newton-cg
Class_weight	None

Table 1: Logistic Regression best parameters

4.1.2 Support Vector Classifier (SVC)

The second classification method that was used is a Support Vector Classifier. Again, a grid search with 5 fold cross-Validation and shuffling enabled was performed to find the best model hyperparameters (Table 2). The parameters tested are:

- **'C'**: this parameter has been chosen in order to set the number of misclassification errors allowed by the soft margin classifier. The best parameter found by the grid search is '10.0';
- **'Kernel'**: this parameter specifies the kernel function. **Rbf** (Radial Basis Function Kernel) has resulted as the best choice: it is used for non-linear problems and it behaves like a weighted Nearest Neighbor model;
- **'Gamma'**: it is the kernel coefficient, which can be tuned since the Rbf has been selected by the grid search. The value **'Scale'** has been selected as the best parameter configuration;
- **'Class_weight'**: as before, this parameter selects the weights associated with classes. 'None' has resulted as the best configuration.

At this point, the model with the best parameters was used to make a prediction on the test set. The model's prediction on the test set achieved an accuracy of 0.55 (figure 34). This is a very good score, as it is exactly 11 times higher than the random value of 0.05. This score is better than the one obtained with Logistic Regression.

	precision	recall	f1-score	support
emo	0.43	0.39	0.41	94
folk	0.29	0.24	0.26	100
goth	0.33	0.20	0.25	119
happy	0.74	0.70	0.72	97
heavy-metal	0.43	0.54	0.48	105
honky-tonk	0.59	0.83	0.69	84
j-idol	0.61	0.62	0.61	99
kids	0.53	0.66	0.59	105
minimal-techno	0.82	0.84	0.83	115
mpb	0.38	0.29	0.33	101
new-age	0.68	0.71	0.69	106
opera	0.62	0.65	0.64	89
piano	0.76	0.45	0.56	94
progressive-house	0.57	0.64	0.60	99
salsa	0.54	0.66	0.60	89
sertanejo	0.69	0.69	0.69	89
sleep	0.88	0.84	0.86	106
songwriter	0.37	0.36	0.37	119
synth-pop	0.32	0.32	0.32	99
world-music	0.44	0.52	0.48	88
accuracy			0.55	1997
macro avg	0.55	0.56	0.55	1997
weighted avg	0.55	0.55	0.55	1997

Figure 34: Support Vector Classifier results

Parameter	Value
C	10.0
Kernel	Rbf
Gamma	Scale
Class_weight	None

Table 2: Support Vector Machines best parameters

4.1.3 Random Forest Classifier

The third classification method that was used is an ensemble method, namely Random Forest, which uses an ensemble approach by combining different decision trees. Once again, a grid search with 5 fold cross-Validation and shuffling enabled was performed to find the best model parameters (Table 3). The parameters tested are:

- **"N_estimators"**: the number of trees used for the model design. The best parameter found by the grid search is "150". In order to visualize the trend of the accuracy w.r.t. the number of estimators, a plot has been plotted by iteratively increasing the number of estimators and measuring the accuracy, mantaining the same values for the other hyperparameters:

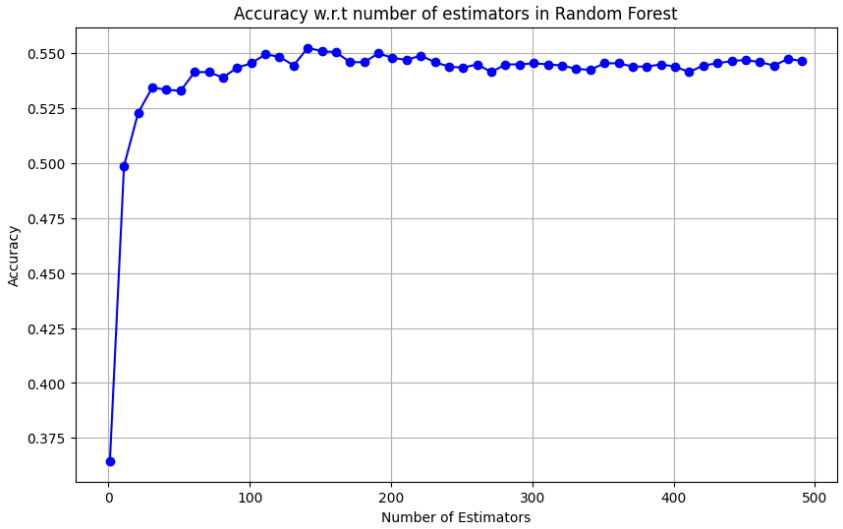


Figure 35: accuracy trend w.r.t number of estimators

- **"Max_depth"**: this is the maximum depth of each tree. "50" has resulted as the best hyper-parameter;

- **"Criterion"**: the function to measure the quality of a split, which is fundamental for the design of the model. **'Gini'** has resulted as the best choice.

At this point, the model with the best parameters was used to make a prediction on the test set. Again, the prediction of the model on the test set obtained an accuracy of 0.55 (Figure 36). This score is the same as the one obtained with Support Vector Machines.

	precision	recall	f1-score	support
emo	0.48	0.30	0.37	94
folk	0.24	0.25	0.25	100
goth	0.35	0.33	0.34	119
happy	0.58	0.84	0.69	97
heavy-metal	0.60	0.24	0.34	105
honky-tonk	0.61	0.82	0.70	84
j-idol	0.57	0.61	0.59	99
kids	0.53	0.67	0.59	105
minimal-techno	0.85	0.90	0.87	115
mpb	0.34	0.27	0.30	101
new-age	0.69	0.74	0.71	106
opera	0.60	0.72	0.65	89
piano	0.69	0.51	0.59	94
progressive-house	0.58	0.72	0.64	99
salsa	0.50	0.71	0.59	89
sertanejo	0.76	0.36	0.49	89
sleep	0.90	0.92	0.91	106
songwriter	0.42	0.34	0.38	119
synth-pop	0.29	0.51	0.36	99
world-music	0.58	0.22	0.31	88
accuracy			0.55	1997
macro avg	0.56	0.55	0.53	1997
weighted avg	0.56	0.55	0.53	1997

Figure 36: Random Forest results

Parameter	Value
N_estimators	150
Max_depth	50
Criterion	Gini

Table 3: Random Forest best parameters

Moreover, with Random Forest a barplot of the most important features, which have conditioned the classification process, has been plotted:

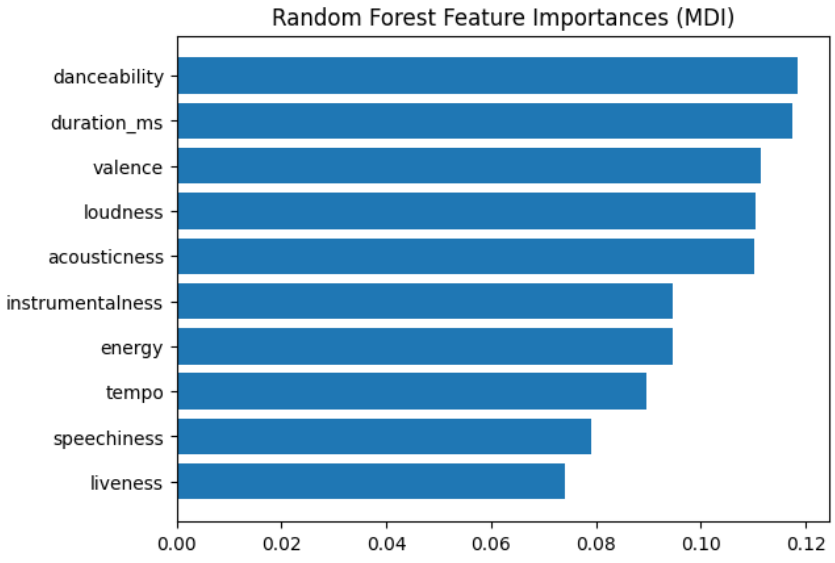


Figure 37: Most important features identified by the Random Forest Classifier

4.1.4 Bagging Classifier

The fourth classification method that was used is another ensemble method, namely Bagging. In particular, the basic model that was used is the KNN. The parameters tested in the grid search with 5 fold cross-Validation and shuffling enabled (Table 4) are:

- **"Estimator_n_neighbors"**: this parameter allows to select the number of k for every estimator in the Bagging Classifier and it can influence the performances of the model. The best value found by the grid search is "10". A graph which represents the trend of the accuracy with respect to the value of k is shown:

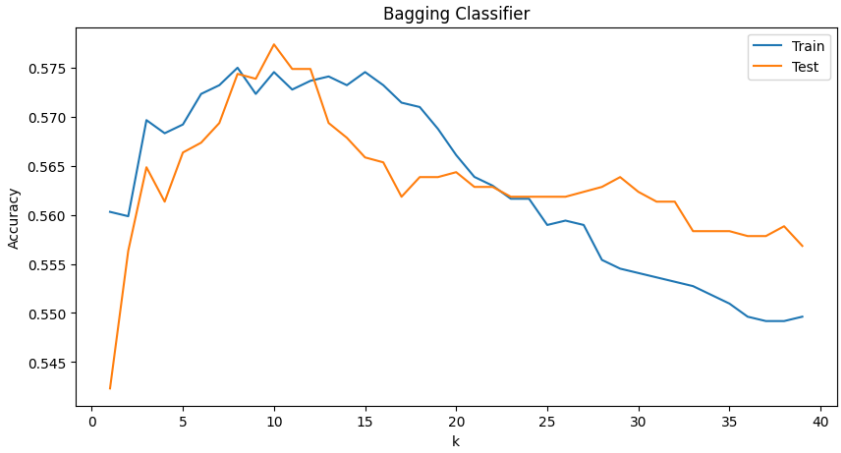


Figure 38: accuracy trend w.r.t number of estimators

- **"Estimator_weights"**: this parameter specifies the weight function of the KNN estimators. The grid search has returned **"Distance"** as the best choice;
- **"N_estimators"**: as Random Forest, this parameter indicates the number of estimators of the classifier. Figure 39 shows the trend of the accuracy w.r.t. the number of estimators: it is possible to notice that the accuracy does not significantly change after a value of 100 estimators;

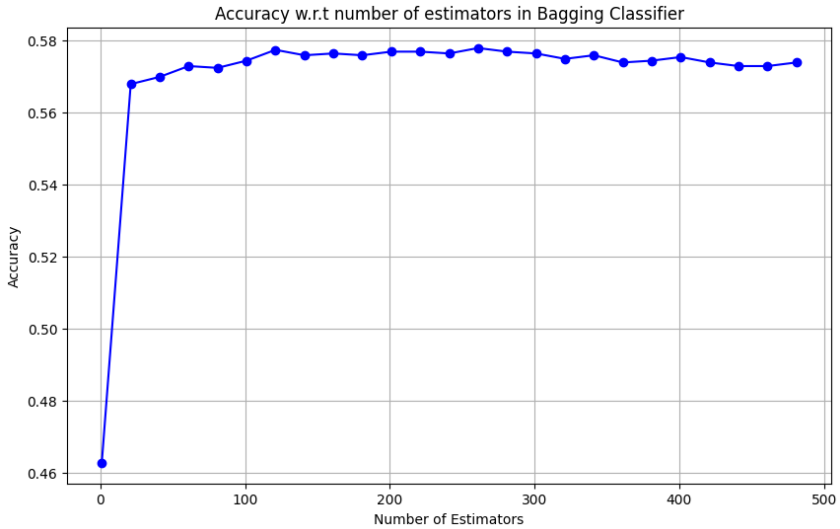


Figure 39: accuracy trend w.r.t estimators value of k

- **"Max_samples"**: this parameter indicates the number of samples used to train each estimator. "1.0" has been chosen by the grid search as best parameter;
- **"Max_features"**: the number of features used to train each base estimator. "0.7" has resulted as the best value;
- **"Bootstrap"**: this parameter indicates whether samples are drawn with replacement. Sampling without replacement (**False**) has produced an increment of the accuracy comparing to the "True" value;
- **"Bootstrap_features"**: this parameter indicates whether features are drawn with replacement. As the bootstrap for the samples, **False** has resulted as the best value.

The model with the best parameters was used to make a prediction on the test set. This time, the model's prediction on the test set achieved an accuracy of 0.58 (figure 40). This score is more than 11 times higher than the random value of 0.05 and is the highest score obtained so far.

	precision	recall	f1-score	support
emo	0.48	0.43	0.45	94
folk	0.28	0.22	0.25	100
goth	0.44	0.18	0.26	119
happy	0.74	0.69	0.72	97
heavy-metal	0.47	0.54	0.50	105
honky-tonk	0.58	0.89	0.70	84
j-idol	0.63	0.68	0.65	99
kids	0.60	0.68	0.63	105
minimal-techno	0.82	0.92	0.87	115
mpb	0.36	0.22	0.27	101
new-age	0.68	0.68	0.68	106
opera	0.58	0.63	0.61	89
piano	0.62	0.51	0.56	94
progressive-house	0.62	0.74	0.68	99
salsa	0.52	0.72	0.60	89
sertanejo	0.62	0.71	0.66	89
sleep	0.92	0.90	0.91	106
songwriter	0.43	0.39	0.41	119
synth-pop	0.48	0.38	0.43	99
world-music	0.43	0.56	0.48	88
accuracy			0.58	1997
macro avg	0.56	0.58	0.57	1997
weighted avg	0.57	0.58	0.56	1997

Figure 40: Bagging results

Parameter	Value
Estimator_n_neighbors	10
Estimator_weights	Distance
N_estimators	300
Max_samples	1.0
Max_features	0.7
Bootstrap	False
Bootstrap_features	False

Table 4: Bagging best parameters

4.1.5 Extreme Gradient Boosting

The fifth classification method that was used is Extreme Gradient Boosting. The parameters tested in the grid search with 5 fold cross-Validation and shuffling enabled (Table 5) are:

- **Max_depth**: as before, the maximum depth of the trees used by the model. "21" has resulted as the best value;
- **N_estimators**: the number of trees. "1450" has been found during the parameter tuning phase;
- **Learning_rate**: the learning rate increases the stability of the model during the training and avoids overfitting, so it has been considered an important parameter to tune. "0.05" has resulted as the best value;
- **Min_child_weight**: this parameter indicates the minimum sum of instance weight needed in a child. "1" has resulted as the best value;
- **Max_delta_step**: this parameter makes the update step more conservative. "1" has resulted as the best choice;

- **Subsample**: the fraction of training data sampled prior to growing trees. The value of "0.8" means that more that the 80% of the data is sampled. This parameter can help to prevent overfitting;
- **Lambda**: L2 regularization term on weights. The grid search has returned the value of "1".

The model with the best parameters was used to make a prediction on the test set. The prediction of the model on the test set achieved an accuracy of 0.55 (figure 41). This score is equal to the one obtained with Support Vector Classifier and Random Forest Classifier.

	precision	recall	f1-score	support
emo	0.44	0.45	0.44	94
folk	0.26	0.29	0.27	100
goth	0.31	0.28	0.29	119
happy	0.66	0.84	0.74	97
heavy-metal	0.55	0.28	0.37	105
honky-tonk	0.62	0.82	0.71	84
j-idol	0.64	0.69	0.66	99
kids	0.63	0.64	0.63	105
minimal-techno	0.85	0.87	0.86	115
mpb	0.33	0.28	0.30	101
new-age	0.66	0.70	0.68	106
opera	0.52	0.71	0.60	89
piano	0.63	0.49	0.55	94
progressive-house	0.56	0.76	0.64	99
salsa	0.49	0.78	0.60	89
sertanejo	1.00	0.20	0.34	89
sleep	0.89	0.91	0.90	106
songwriter	0.41	0.34	0.38	119
synth-pop	0.29	0.41	0.34	99
world-music	0.59	0.23	0.33	88
accuracy			0.55	1997
macro avg	0.57	0.55	0.53	1997
weighted avg	0.56	0.55	0.53	1997

Figure 41: Extreme Gradient Boosting results

Parameter	Value
Max_depth	21
N_estimators	1450
Learning_rate	0.05
Min_child_weight	1
Max_delta_step	1
Subsample	0.8
Lambda	1

Table 5: Extreme Gradient Boosting best parameters

4.1.6 Neural Network

The neural network was implemented using the *Pytorch* library. The design process for the architecture relied on an initial extensive experimentation to determine the optimal number of layers and units per layer. The final architecture adopted for our classification incorporated two hidden layers, each consisting of 150 nodes. Infact, employing a deep neural network with additional hidden layers yielded no significant performance improvement, suggesting that the complexity of the underlying relationship between the input and output variables was adequately captured with two hidden layers. The number of training epochs was fixed at 100, since even with higher epoch did not achieve any appreciable improvement in model convergence.

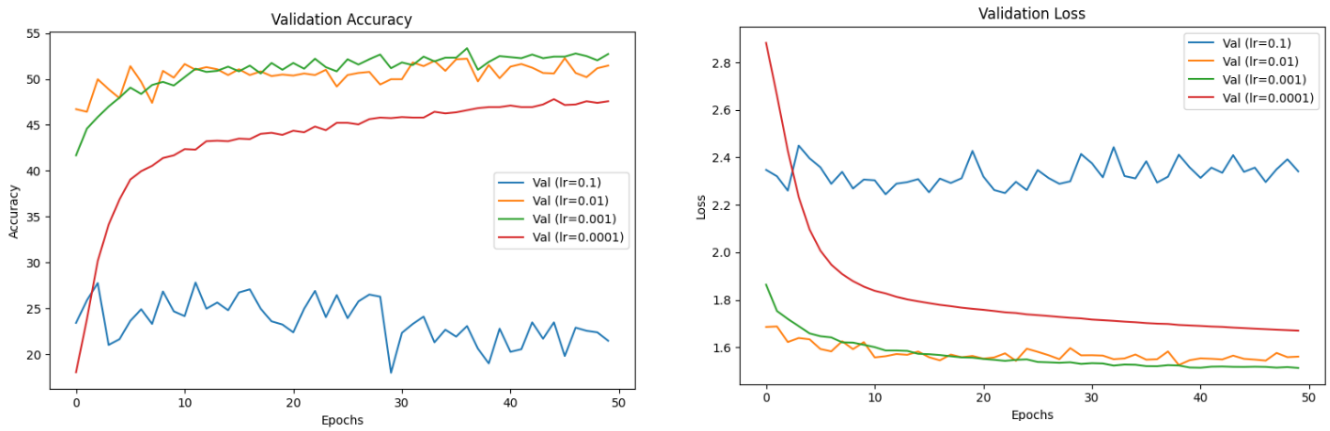
Having established the number of epochs, the optimal number of hidden layers and nodes through preliminary experimentation, the research efforts were focused on empirically determining the remaining hyperparameters using Adam as the optimizer and a batch size of 64.

The chosen hyperparameters were as follows:

- **Learning rate**: [0.1, 0.01, 0.001, 0.0001];
- **Activation function**: [Tanh, Relu, Sigmoid];
- **Drop-out**: [0.1, 0.2, 0.3].

Each neural network was trained employing an early stopping criterion to mitigate the risk of overfitting and reduce unnecessary training time. By halting the training process when validation performance plateaued or began to degrade, early stopping helps to identify a model that generalizes well to unseen data without sacrificing computational efficiency.

The **learning rate**, a crucial hyperparameter influencing model convergence, was carefully considered. Larger learning rates allow the model to traverse the parameter space more quickly but risk overshooting optimal solutions, leading to instability. Conversely, excessively small learning rates, while potentially more stable, may result in prolonged training times or convergence to suboptimal local minima. Our graphs highlights that a learning rate of 0.001 seems to achieve the best balance. The validation metrics (loss and accuracy) steadily improve with relatively smooth curves, indicating stable learning.

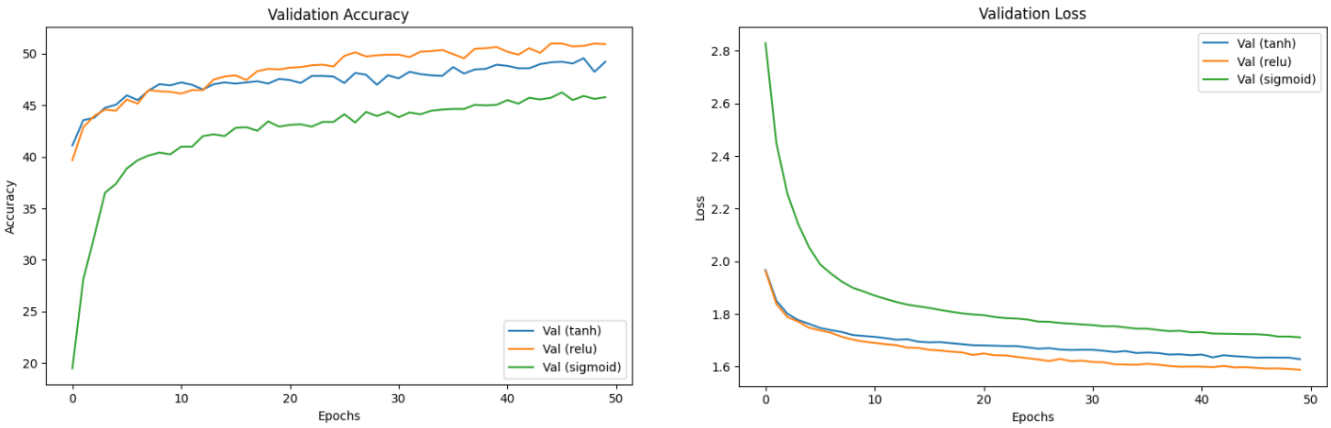


(a) Validation curve for learning rate's accuracy

(b) Validation curve for learning rate's loss

Figure 42: Validation curves for learning rate

The selection of the **activation function** is crucial in shaping the non-linearity of a neural network, significantly impacting is ability to learn complex patterns within the data. Our curves suggests that ReLU offers the best overall performance, achieving the highest validation accuracy while exhibiting a faster reduction in the loss, likely due to its computational efficiency and ability to handle complex data structures effectively. Tanh demonstrates a similar performance but experiences some fluctuations in accuracy.



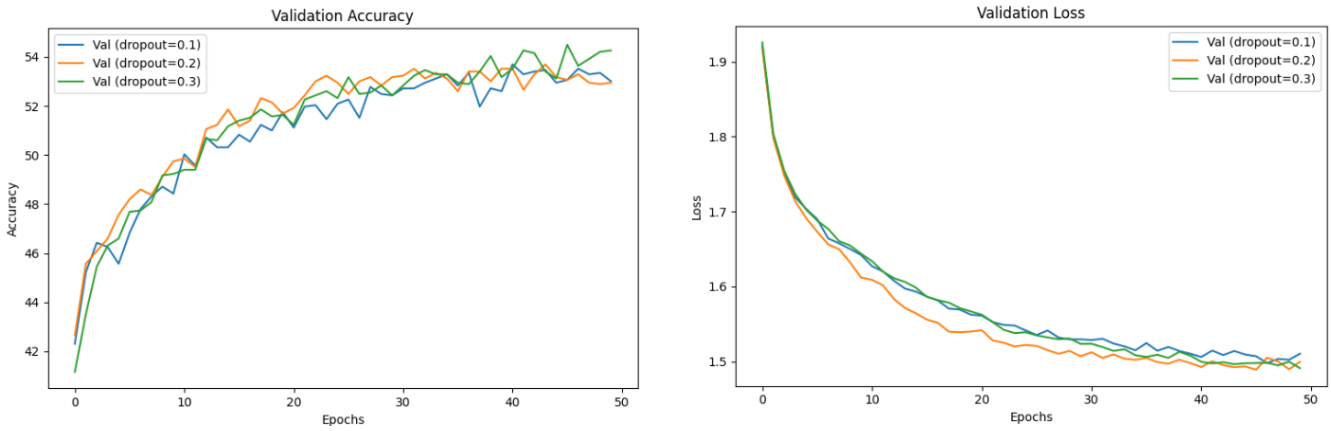
(a) Validation curve for activation functions' accuracy

t

(b) Validation curve for activation functions' loss

Figure 43: Validation curves for activation functions

The last hyperparameter taken into account was the **drop-out**, a regularization technique that randomly "turns off" a percentage of a layer's neuron outputs during each training step. This prevents overfitting by reducing complex co-adaptations between neurons and forcing the network to learn multiple, independent representations of the data. As can be observed, all three drop-out rates reach a similar peak in validation accuracy. For the loss, instead, the drop-out rate of 0.3 shows the most drastic decline in loss during the initial epochs, suggesting it is initially the most effective in preventing overfitting and generalizes well to unseen data.



(a) Validation curve for drop-out accuracy

t

(b) Validation curve for drop-out loss

Figure 44: Validation curves for drop-out functions

Following the hyperparameter exploration, three final models were trained to optimize the performance:

1. **Model with early stopping**: this model implements early stopping with a patience value of 20 epochs. This strategy allows to stop the training when validation performance stagnates or declines for 20 consecutive epochs;
2. **Model with L2 regularization**: this second model utilizes L2 regularization to minimize the complexity of the model. By adding a penalty term to the loss function that is proportional to the squared sum of the weights, L2 regularization encourages the model to have smaller weights, which in turn leads to reduced complexity and improves generalization;
3. **Model with drop-out rate of 0.3**, as suggested by our experimentation.

Models	Test accuracy
Model with early stopping	0.58
Model with L2 regularization	0.58
Model with drop-out	0.59

Table 6: Test accuracy for our three models

4.1.7 Neural Network with additional features

In order to try again the classification task with some of the features created during the feature engineering phase, a Neural Network has been implemented. The features that have been selected are: popularity, danceability, energy, loudness, speechiness, acousticness, tempo, explicit, months_from_publication, album_total_tracks and single_artist.

These features were added to the previously best-performing model: the neural network with drop-out (0.3), which achieved an AUC of 0.94. The model was retrained with the new features, which improved the final classification accuracy, with a final score of 0.63.

	precision	recall	f1-score	support
minimal-techno	0.60	0.51	0.55	190
honky-tonk	0.34	0.28	0.31	184
j-idol	0.39	0.21	0.27	182
sleep	0.76	0.82	0.79	191
songwriter	0.54	0.66	0.59	180
synth-pop	0.64	0.90	0.75	161
world-music	0.64	0.74	0.69	182
emo	0.73	0.71	0.72	195
folk	0.78	0.85	0.81	198
goth	0.41	0.16	0.22	180
happy	0.67	0.69	0.68	175
heavy-metal	0.66	0.73	0.69	183
progressive-house	0.72	0.51	0.59	193
salsa	0.67	0.78	0.72	199
sertanejo	0.69	0.76	0.73	178
mpb	0.58	0.75	0.66	186
new-age	0.89	0.89	0.89	200
kids	0.50	0.55	0.52	194
opera	0.44	0.42	0.43	178
piano	0.59	0.62	0.60	194
accuracy			0.63	3723
macro avg	0.61	0.63	0.61	3723
weighted avg	0.61	0.63	0.61	3723

Figure 45: Neural network results

4.1.8 Conclusions

Giving the conclusions, advanced ML models demonstrated to be able to achieve better performances than simpler models, being able to better discriminate the genres of the tracks. In addition, the features created during the feature engineering phase have resulted useful to add information for this classification task.

Looking at the results, it is possible to notice that some of the genres, especially folk and mpb (Música popular brasileira) are more difficult to being correctly classified by every model.

A possible approach in order to increase the results could be the implementation of other models, like an ensemble model with another ensemble model as estimator.

In order to visually compare the performances of the models, the ROC curve plot has been represented by using a single curve for every model, which aggregates their single classes curves.

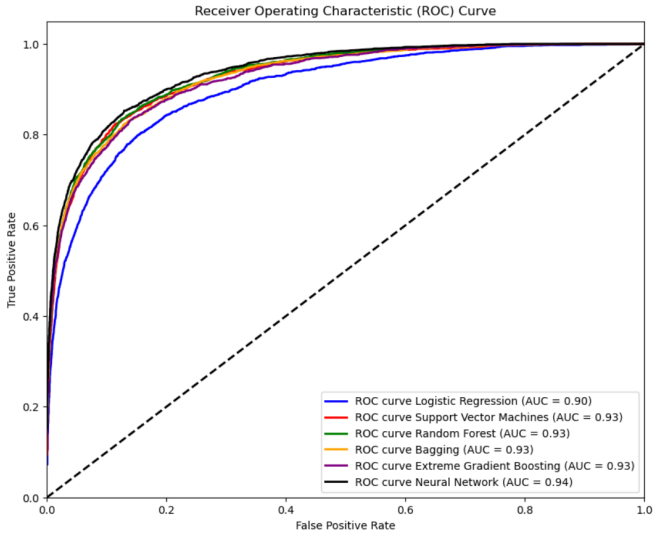


Figure 46: ROC curve of the classification models

4.2 Advanced regression

For the regression task, two different models have been implemented in order to predict the popularity of the track dataset. It was decided to use the dataset of 103691 records in order to use as much data as possible. To solve the regression task that was defined, the attributes of the dataset have been divided into an attribute Y (the popularity variable) and a set of attributes X containing the independent variables most correlated with the dependent variable: track_number, album_total_tracks, months_from_publication and instrumentalness. The values of the variables in X have been normalized with Standard Scaler and the popularity has been normalized with MinMax Scaler in order to have a float variable compatible with the predictions of the models. In a second step, the dataset has been divided into a training set containing 90% of the records and a test set containing 10% of the records. In order to evaluate the performances of the regressors, R2, Mean Squared Error (MSE) and Mean Absolute Error metrics have been chosen.

4.2.1 Support Vector Regressor (SVR)

The first model that has been chosen is Support Vector Regressor. A grid search with 5 fold cross-Validation was performed to find the best model hyperparameters. The tested parameters are:

- **'C'**: this parameter has been chosen in order to set the number of misclassification errors allowed by the model. The best parameter found by the grid search is '1.0';
- **'Kernel'**: **Rbf** (Radial Basis Function Kernel) has resulted as the best function for this model;
- **'Gamma'**: the value **'Auto'** has been selected as the best parameter configuration;
- **'epsilon'**: it specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value. The best parameter found by the grid search is '0.1'.

The model with the best parameters has been used to make a prediction on the test set. The prediction of the model on the test set achieved a R2 score of 0.385, MSE = 0.036 and MAE = 0.151.

Parameter	Value
C	1.0
Kernel	Rbf
Gamma	Auto
epsilon	0.1

Table 7: SVR best parameters

Metric	Value
R2	0.385
MSE	0.036
MAE	0.151

Table 8: SVR test scores

A plot of the predicted examples has been visualized:

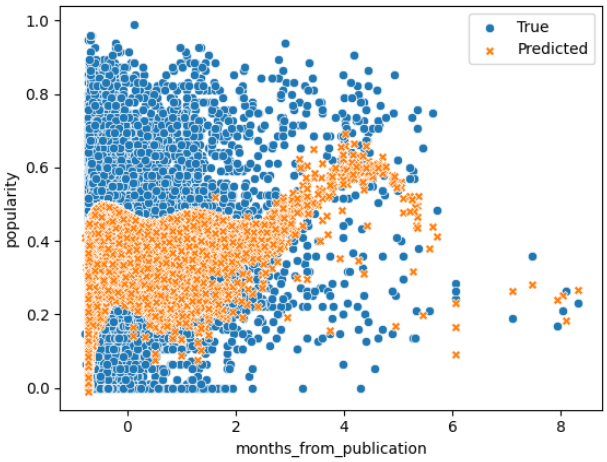


Figure 47: svr true-predicted plot

4.2.2 Gradient Boosting Regressor

The second model that has been chosen is Gradient Boosting Regressor. A grid search with 5 fold cross-Validation has been performed in order to find the best model hyperparameters. The tested parameters are:

- **'loss'**: this parameter indicates the loss function to be optimized. We decided to keep the default 'squared_error' loss;
- **'learning_rate'**: this parameter regularizes the contribution of each tree of the model. The value of 0.1 has resulted as the best choice;
- **'max_depth'**: it has been decided to search the best maximum depth of the individual estimators in order to prevent overfitting. A value of 7 has been selected by the grid search;
- **'min_sample_split'**: it specifies the minimum number of samples required to split an internal node. The grid search has returned a value of 2.

The model with the best parameters has been used to make a prediction on the test set. The prediction of the model achieved a R2 score of 0.482, MSE = 0.031 and MAE = 0.131.

Parameter	Value
loss	squared_error
learning rate	0.1
max_depth	7
min_sample_split	2

Table 9: GB Regressor best parameters

Metric	Value
R2	0,482
MSE	0.031
MAE	0.131

Table 10: GB Regressor test scores

A plot of the predicted examples has been visualized:

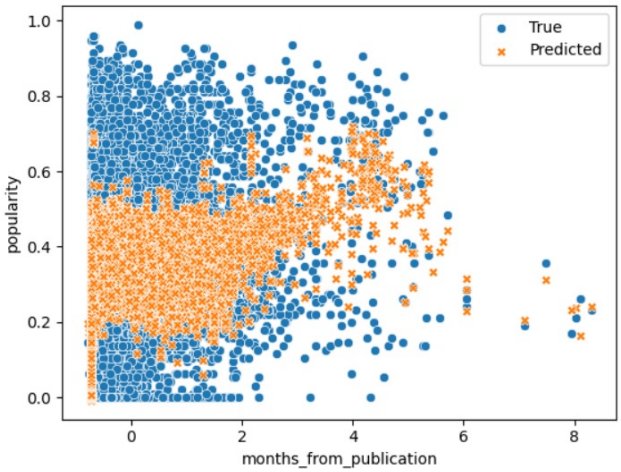


Figure 48: gb true-predicted plot

As we can see, this model has better performances w.r.t to the SV regressor, meaning that an ensemble approach of decision trees is able to better predict the popularity variable.

4.3 Explainability

At the end of the classification phase, it was noticed that the folk genre consistently emerged as the most challenging class for our classifiers. To understand the factors contributing to this misclassification, it was employed **Local Interpretable Model-agnostic Explanations** (LIME), a technique that provides local explanations for individual predictions by learning a simpler, interpretable model (like linear model, for example) around the instance being explained. To achieve this, all instances belonging to the folk class have been isolated, regardless of their classification outcome (correct or incorrect). In this way, it was generated a set of local feature importance vectors, one for each example. To obtain a global perspective on feature influence, it was averaged the features importance scores across all instances.

4.3.1 LIME

It has been decided to explain the reasons after the predictions of the Support Vector Classifier towards the "folk" genre. During the initialization of the LIME explainer, it has been decided to discretize the continuous features in quartiles and standardized with StandardScaler for a better visual representation. After isolating the folk instances, the feature importance scores for every instance were gathered inside a dictionary. After that, the mean importance scores have been calculated for every feature and represented into a bar chart.

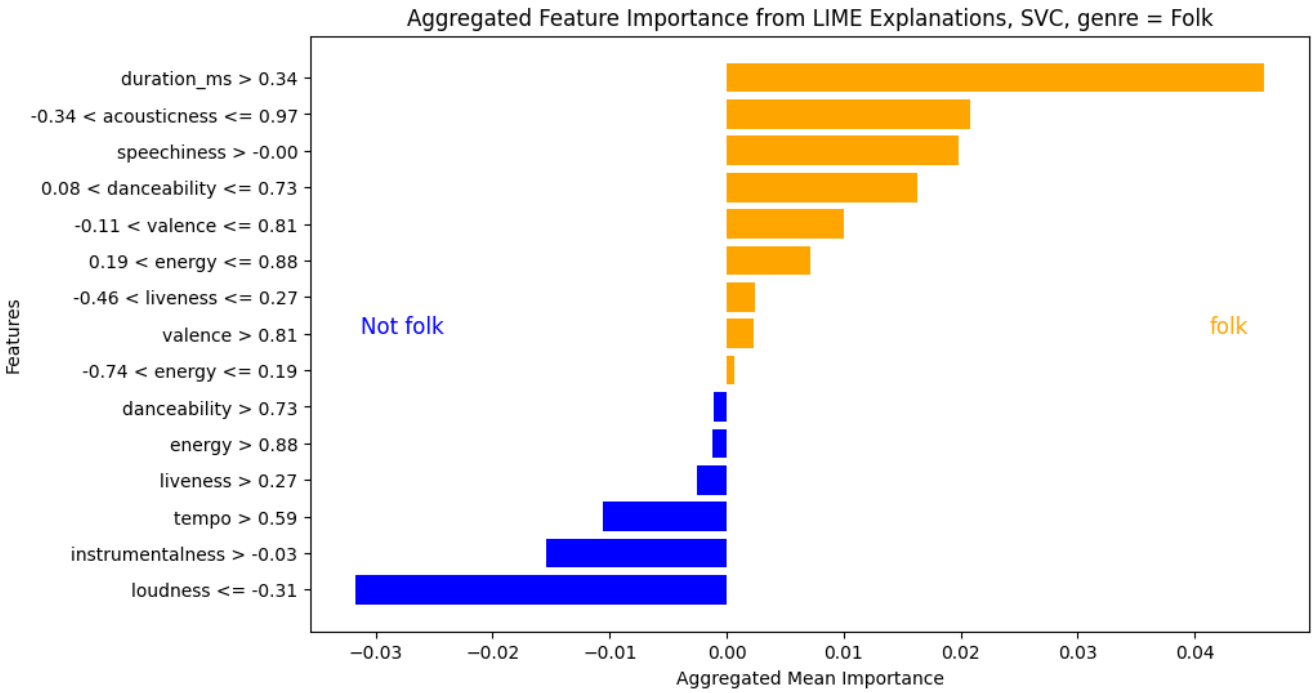


Figure 49: LIME Mean importance scores for the folk genre (predictions with SVC)

As we can see, the most important condition for identifying the folk class is a value of duration higher than the z-score of 0.34, whereas a value of loudness lower than 0.31 makes the model predict for another label.