The role of musical attributes in predicting the popularity of songs: A Spotify case study

Data Mining II

Sana Afreen, Helen Naomi Cedeño Manrique, Luis Hernán Pinto Alemán
Master Degree in Data Science and Business Informatics
University of Pisa
{s.afreen, h.cedenomanrique, l.pintoaleman}@studenti.unipi.it

1 Introduction

In the digitalization era, companies have seen the need to store and manage large volumes of data in order to obtain meaningful insights [9] that improve their interaction with target customers. A big part of these contributions are related to data mining techniques, which blend traditional statistical methods with big data analysis algorithms, enabling the management sector to make informed, evidence-based decisions.

While these techniques are widely used across the entire product sector, they have particular relevance in the service industry, particularly in digital services. This is the case of Spotify, the music streaming company with the most users worldwide [8]. The activity maintained in this application involves the generation and input of gigantic volumes of information, from which a small portion will be analyzed in this project. In particular, it intends to explore the determining factors in the popularity of a song, generating results of interest to both the streaming and music production sectors.

2 Data Understanding and Preparation

The information available for analysis purposes is divided into three datasets, two in the form of two-dimensional data tables and the other is a collection of time series. In all cases, preprocessing and exploratory analysis of the data will be carried out, according to its corresponding structure.

2.1 Tabular Datasets

2.1.1 Tracks

This dataset contains information about the main characteristics of a sample of songs hosted on Spotify, and is composed by 109547 observations and 34 attributes. Among the columns are 12 discrete variables, distributed between the nominal data particular to each song such as id, name and artists; the album information including name, type and release date, and discrete musical attribute information such as genre, key and mode. Similarly, there are 22 continuous attributes mainly related to sound features, as energy, danceability, speechiness, loudness and number of beats and bars, as well as the confidence index of each musical attribute, and the duration and popularity of the tracks.

For the exploratory data analysis, the nominal variables will not be considered, since they do not provide any meaningful insights. Also, it can be observed that a song identified by its unique code can appear more than once in the dataset, because only one genre is considered in each record but a song can belong to more than one genre. According to the objective of our study, it is important to examine the main characteristics of the popularity metrics, as well as their relationship with other variables.

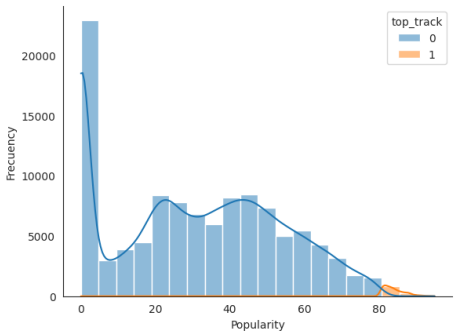


Figure 1: Popularity histogram.

As can be seen in Figure 1, *popularity* has a positive skewness, so most of the observations have low levels of popularity and very few records have high popularity. In fact, between the third quartile and the maximum value there is a difference of 46 points of popularity, while only the 1 percent of the most popular songs is greater than

80. This, added to the fact that only 1.1% of all artists and songs produced around the world are considered mainstream [4], allow us to create the the binary variable *top_track*, which indicates whether the song is in the top 1 percentile of *popularity* and helps to describe the main characteristics of the most popular songs.

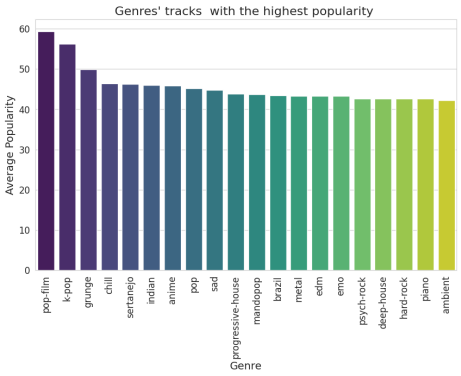


Figure 2: Genres with the highest average popularity.

Figure 2 shows the popularity of songs by genre, where it is observed that, in average, the songs with the highest popularity are *pop-film*, *k-pop* and *grunge*. Within the songs in the highest percentile of popularity, the most popular genres are *pop* and *rock*.

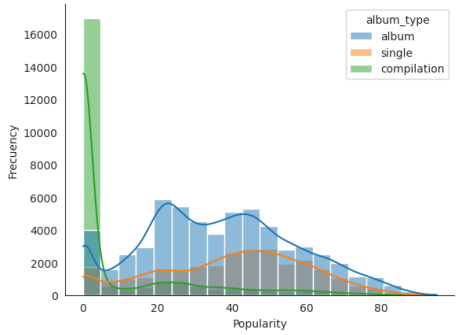
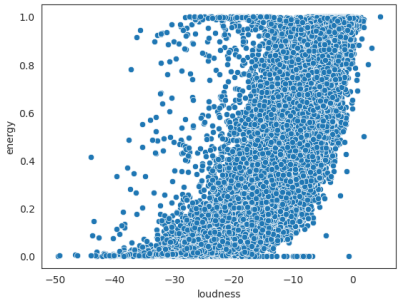


Figure 3: Popularity histogram by album type.

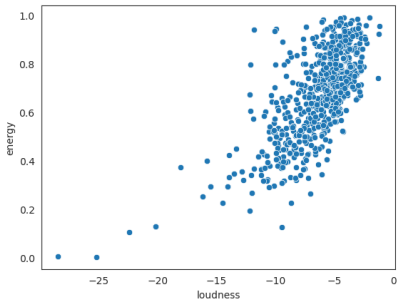
Regarding the album type, in figure 3 it can be seen that songs coming from compilation albums tends to have very low popularity. In contrast, songs coming from albums as such have the highest levels of popularity.

For the sound attributes analysis, the duration of the track has very distant extreme values, but 99% of the data are below 10 minutes. The median duration of the song is 3.5 minutes, while its maximum value is approximately one hour and 8 minutes.

Also, it was observed that *loudness* and *energy* are directly proportional to each other, so as the loudness increases, the energy increases as well. Moreover, looking at the songs belonging top 1% of artists in the figure below, most of the songs are concentrated at the top of *loudness* and *energy* values.



(a) General



(b) Top 1%

Figure 4: Loudness vs. Energy.

Once the exploratory data analysis has been done, **data quality** steps are performed in order to ensure that data is reliable, leading to better outcomes and reducing the risk of misleading or biased results. The first variable addressed was *album_release_date*, so it was decided to keep the just the year in all records aiming to for maintain a standardized format and avoid losing information from the records that do not have a month or day specified.

Next, a syntactic and semantic check was done, leading to the finding of several inconsistent observations of *loudness*. According to Podolak [6], in digital audio the measure used to quantify loudness is 0, which is the maximum possible full-scale decibel value (dBFS). When a track has a higher noise level, it is digitally trimmed to a maximum level of 0 dBFS. In the dataset, there are 84 observations with noise values higher than 0, which are considered erroneous and, consequently, were removed. On the other hand, the dataset contains negative values for confidence-related values. Specifically, it's notable that *key_confidence* and *mode_confidence* have values below zero, which are semantically incorrect. Therefore, rows with these values will be eliminated.

Finally, the dataset contained several records that had repeated values in all its component variables, which is mainly related to the fact that a song can be present in several types of albums. Therefore, it was decided to remove these du-

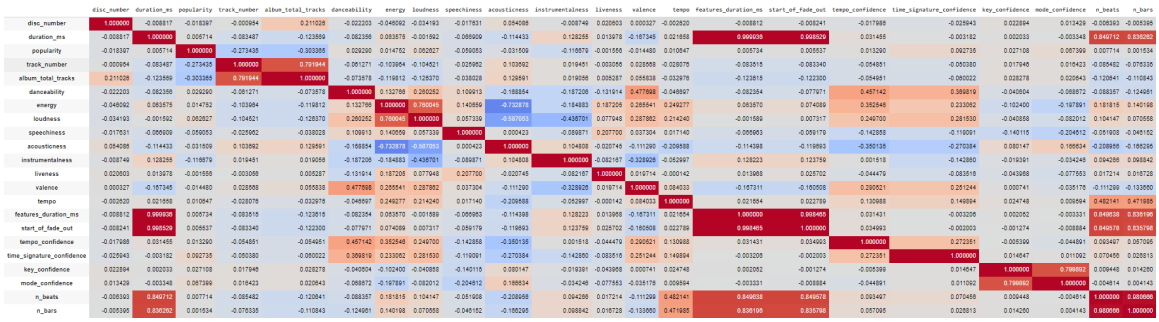


Figure 5: Correlation Matrix

plicates from the dataset, always prioritizing the duplicate observation belonging to albums, then those that are singles and finally those that are part of a compilation.

After the process of cleaning up records with semantic or syntactic errors, as well as removing duplicate records, 483 records have been removed. For subsequent analysis and implementation of data mining algorithms, the following variable transformations were made:

- The variables *key*, *mode* and *time_signature* will be changed from numeric to categorical, with the objective of avoiding performing statistical calculations of these variables when computing the correlation matrix.
- Since *popularity* is defined between 0 and 100, it will be divided by 100 to be related to the other variables whose interval is between 0 and 1, and to obtain a better decimal precision in case of performing regressions with respect to this variable in later steps.
- A scaling will be made to *loudness* using the min-max method, so that it is related to the scale of the other variables present in the data set.

The correlations between each pair of variables can be analyzed using the correlation matrix. As visualized in figure 5, there is a correlation very close to 1 between *duration_ms*, *start_to_fade_out* and *features_duration_ms*. This indicates that all these variables are highly similar to each other so two of the variables can be deleted, which in this case will be *features_duration_ms* and *start_to_fade_out*.

Regarding the *popularity*, the negative relationship it has with *instrumentalness* is noteworthy. This means that as the probability that a song does not contain vocals increases, the popularity decreases. It can also be noted the relationship of *popularity* with *track_number* and *album_total_tracks*. The strong positive relationship between *loudness* and *energy* is also highlighted, in addition to the negative relationship of *acousticness* with *loudness* and *energy*. This shows that tracks with high energy and noise have low acoustic level.

Finally, via the correlation analysis it was detected a subset of attributes that provide information for the classification of musical genres and the prediction of popularity of the songs contained in the dataset. These attributes are, *duration_ms, explicit, popularity, album_type, album_relase_date, album_total_tracks, track_number, danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, n_beats, n_bars and genre*. The variable *id* is also retained to facilitate the linking of the table with the other datasets.

2.1.2 Artists

The artist information dataset contains 30141 records with 5 variables, of which three are nominal and two are numeric. Inside the table are the id of each artist, its popularity index, their number of followers on Spotify, and the most representative musical genre of their songs.

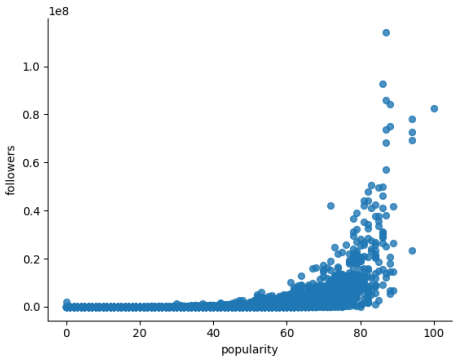


Figure 6: Artists' Popularity vs Followers

Inside the dataset is an underlying relationship between followers and popularity. As can be seen in Figure 6, artists with a higher number of followers tend to have a higher level of popularity. However, when analyzing the last percentile of popularity and followers separately, it is observed that in some cases, the artist who is in the top 1% in terms of followers, is not in the top 1% of popularity. Specifically, 212 artists are in the top

1% of both categories, while 222 others do not comply this criterion.

On the other hand, only a row containing only missing values along the attributes was detected, as well as duplicate values in the dataset. Thus, these records were removed, summing up to 3 observations deleted.

2.2 Time Series Dataset

The time series dataset consists of a set of 10000 time series, representing the spectral centroids of the audio files of songs from the Spotify catalog in .mp3 format. At the time of loading the series, the number of time steps was standardized to 1280 time segments each, so that if any series went over this amount it would be truncated so that all series would have the same length.

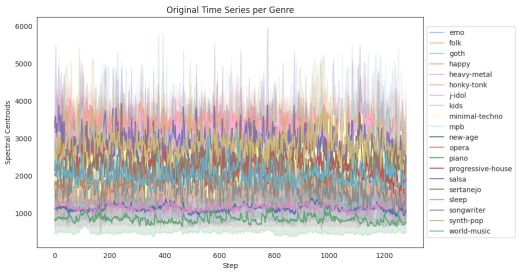


Figure 7: Time series per genre

Each time series is a song with a unique identifier and an associated genre. As can be visualized in Figure 7, there are 20 different music genres distributed in a balanced way, i.e. 500 songs per genre.

2.2.1 Transformations

To choose the transformations to be used on the time series, experiments were carried out with all the possible transformations: Offset Translation, Amplitude Scaling, Linear Trending and Noise Removal. Finally, it was decided to use a combination of Offset Translation and Noise Removal, in that order, because it presented the best preliminary clustering and classification results among all the transformations tested.

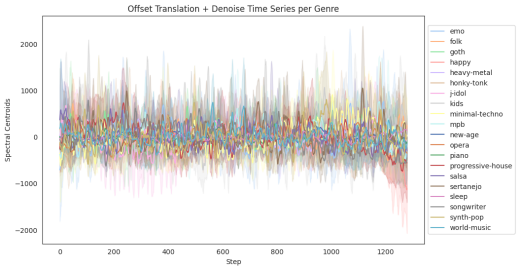


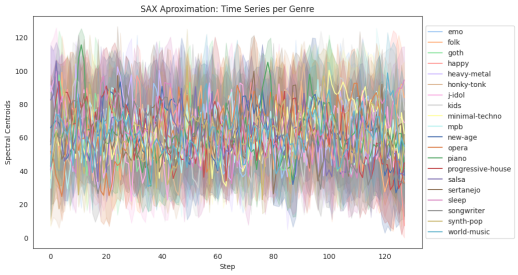
Figure 8: Offset Translation + Denoising time series per genre.

This combination of transformations allows the normalization of the time series and the removal of information that does not contribute to the algorithms used. Figure 8 shows the time series grouped by gender, once the previously mentioned transformation processes have been applied.

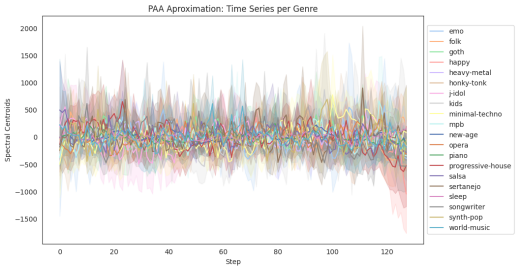
2.2.2 Approximations

Since the time series has a long length, the use of approximation algorithms will considerably reduce the computation time for the data mining work.

For this reason, preliminary clustering tests were performed with the approximation models that maintain the temporal structure of the time series, i.e. Symbolic Aggregate Approximation (SAX) and Piecewise Aggregate Approximation (PAA). From these tests, better results were obtained when using PAA, so this approximation will be used as input in the algorithms of the following section.



(a) SAX



(b) PAA

Figure 9: Time series after approximation processes by genre

As shown in figure 9, the SAX approach shows a higher dispersion of the time series, and therefore a higher variability, but also the series are more detailed. On the other hand, the series after applying PAA are more accumulated around zero, centralizing the values of each time step, and show less variability and noise. After testing with both methods, the PAA approach was chosen because it gave the best results.

3 Time Series Analysis

3.1 Matrix Profile

A matrix profile P of time series T is a vector of the Euclidean distances between each subsequence $T_{i,m}$ and its nearest neighbor [12]. Once the data was transformed and approximated, a matrix profile was generated to enable the extraction of motifs and discords from the time series.

3.1.1 Motifs

A motif is a recurring pattern that appears in a time series, which is identified using the matrix profile. In this case, the index location where the matrix profile has the smallest value corresponds to the position of the most similar subsequence, i.e. the motif. To extract the motifs, it is also necessary to establish a rolling window size that is used to segment the time series into sub-sequences, where in each sub-sequence the corresponding motifs will be identified.

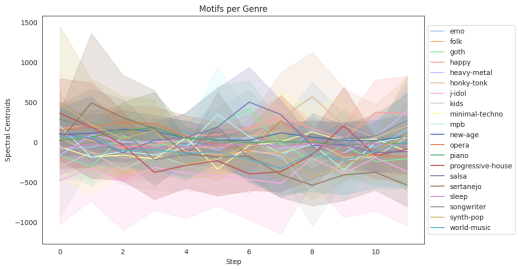


Figure 10: Motifs per Genre.

The graph shows 5 motifs extracted from the time series by gender, using a sliding window of size $m = 12$, which roughly corresponds to 10% of the time intervals of each measurement and maintains a balance between captured data and extraction capacity. In this figure, it can be observed that the motifs are divided into two groups, those with steep rises and falls and those that remain quasi-stable around zero. Besides, there are genres whose motifs show a higher dispersion, such as *heavy metal*, *synth pop* and *honky tonk*.

3.1.2 Discords

Discords, unlike motifs that present repetitive substrings, represent exceptions or atypical events compared to the rest of the time points. These sub-series can indicate errors or anomalies, which is useful to identify outliers and better understand the underlying behavior of the data. To find the discords, one must find the location of the index where the matrix profile has the highest value.

Following the same format as before, 5 discords per genre were plotted using the same series used for plotting the motifs. As can be seen in the plot above, the discords present a much more variable behavior than the series and motifs, showing very steep upward slopes in most cases, and less steep drops that tend to stay close to zero. This may suggest that anomalous events cause more abrupt increases in values than decreases. On the other hand, the behavior of the discords varies between genres, with genres such as progressive house, synth pop, and minimal techno showing a larger spread, indicating a possible greater presence of anomalies.

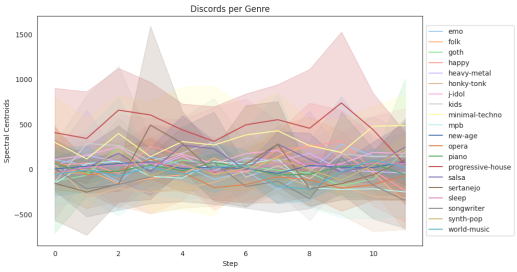


Figure 11: Discords per Genre.

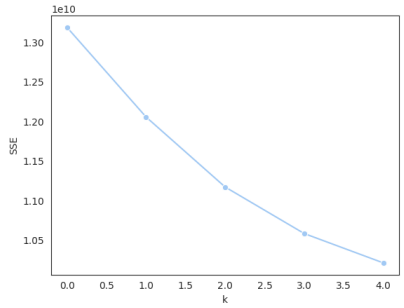
3.2 Clustering

In this section, several clustering algorithms and the result of their application will be discussed. In order to reduce the execution time of the clustering techniques, the input used were the extracted motifs from the time series after denoising, offset translation and PAA approximation. This combination was also chosen -among other tests as raw data, different transformations, other approximations as SAX and DFT- since it provide better clustering results in terms of the reduction of SEE and a greater Silhouette coefficient.

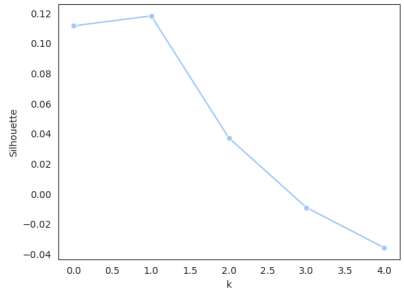
3.2.1 K-Means Time Series

The starting point is to determine the optimal number of clusters. To achieve this, the sum of squared errors (SSE) between 2 and 6 potential clusters was computed. Given the SEE of each cluster, the value with the lowest number of clusters corresponding to the lowest possible SSE was chosen, since this shows that its centroids best represent the observations of each cluster [9].

In this case, both Euclidean and Dinamic Time Warping (DTW) were tested, but since Dynamic Time Warping shown better results than Euclidean distance only the results with DTW will be displayed. The main reasoning behind the difference of performance is that certain time series may be almost identical, but they may also shift in time -along the time axis- [3], which benefies DTW applications.



(a) SSE



(b) Silhouette

Figure 12: Selection criteria for choosing k.

As can be observed in figure 12, the configuration that performs best in both SSE and Silhouette - where in fact the coefficient is maximized- is three clusters. Therefore, all results shown are where $k = 3$.

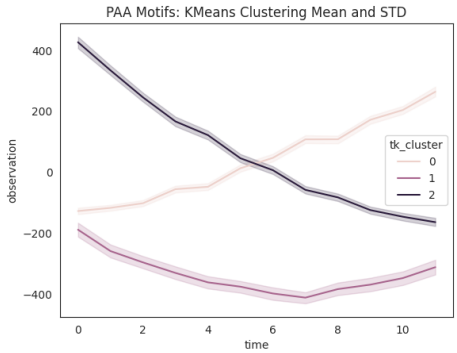
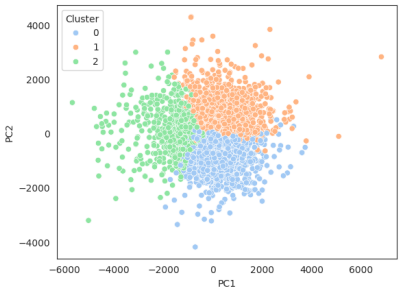


Figure 13: Clusters obtained from motifs extraction.

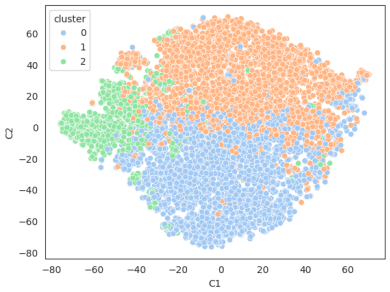
Figure 13 shows the average shape of the obtained clusters. Thus, three different shapes of the time series considered as centroids can be distinguished. The **cluster 0** has an increasing trend starting from a value lower than the average value of the time series. On the other hand, the **cluster 1** has a slightly concave shape, staying for values below the average. Finally, the **cluster 2** has a decreasing trend, starting with the highest values among the clusters and ending with values below the mean.

A detailed analysis of the characteristics of each cluster reveals the following:

- **Cluster 0:** This cluster contains *j-idol*, *goth* and *heavy-metal* as the most representative genres. The songs contained in this cluster have the lowest level of popularity, danceability and duration with respect to the other clusters.
- **Cluster 1:** The most representative genres in this cluster are *sleep*, *piano* and *minimal-techno*. In this cluster, the songs are characterized by a high instrumentalness and acousticness, as well as a low energy level. The popularity level of these is surrounding the average of the whole dataset.
- **Cluster 2:**. The most frequent genres in this cluster are *progressive-house*, *minimal-techno* and *songwriter*. These songs have the highest level of noise, energy and popularity, but in contrast, they possess the lowest level of instrumentalness and acousticness. This may hint that the songs with high noise and energy are the most popular within the dataset.



(a) PCA



(b) t-SNE

Figure 14: 2-dimensional clustering representation.

As shown in Figure 14, when visualizing the time series with dimensionality reduction using Principal Component Analysis, the K-Means algorithm with the previously defined hyperparameter configuration presents a structure with well-defined clusters, with no overlapping between clusters. When using t-distributed Stochastic Neighbor Embedding (t-SNE), a little overlapping between clusters is displayed, but the shape of the clusters remains defined.

3.2.2 Hierarchical Clustering

The clustering with hierarchical approach was performed on the denoising and offset transla-

tion transformation of the time series, without applying any approximation. Approximation of the data was avoided because this type of clustering uses feature extraction to generate a static reproduction of the time series [7], so by using approximations, information that is important for the computation of features describing the distribution of the variables would be lost.

Using feature extraction, it is possible to adapt the data to classical clustering algorithms and avoid problems related to the high dimensionality of the data set. In particular, the features extracted from the time series were the mean, standard deviation, minimum and maximum, as well as the quantiles 0.1, 0.25, 0.5, 0.75 and 0.9.

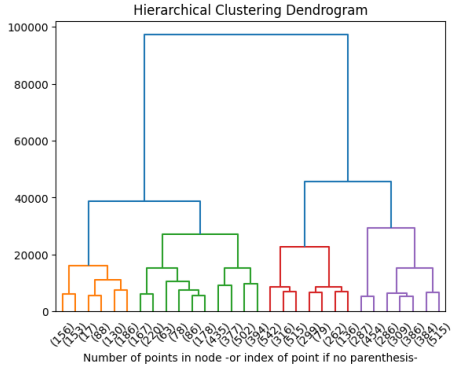
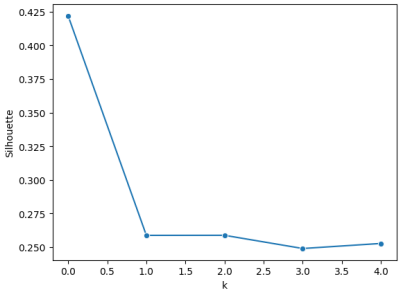


Figure 15: Dendogram plot for ward linkage

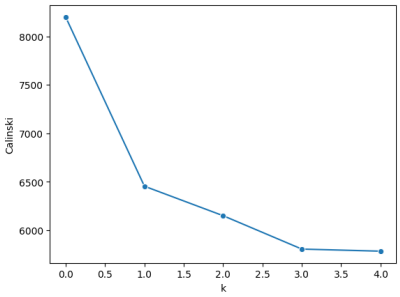
Once the feature extraction process is completed, the hierarchical dendogram is plotted in order to decide the type of linkage to be used in this algorithm. In this way, the dendogram that uses the ward linkage is the one that presents the best clustering structure. As can be seen in figure 15, this type of linkage has a well-defined and separated cluster structure.

Next, a performance evaluation was performed for $k \in [2, 6]$, in order to determine the optimal number of clusters. As shown in Figure 16, clustering is better when four clusters are used, so this configuration will be used. This allowed us to generate a clustering model with the following characteristics:

- **Cluster 0:** This cluster has *sleep*, *new-age* and *piano* as its most representative genres. The songs contained in this cluster are characterized by a longer duration, the highest level of instrumentalness among the clusters, a relatively high level of acousticness, and low level of popularity and danceability.
- **Cluster 1:** The most representative musical genres in this cluster are *sertantejo*, *kids* and *songwriter*. All these musical genres are characterized by high levels of energy and danceability.
- **Cluster 2:** Mainly composed by *salsa*, *happy* and *j-idol*, these songs are characterized by low acousticness and the highest energy ratings among the clusters.
- **Cluster 3:** The most predominant genres inside this cluster are *emo*, *mpb* and *kids*. The average popularity and acousticness are the highest, while these songs have the lowest duration -a common pattern in popular songs as they are easier to retain- and instrumentalness.

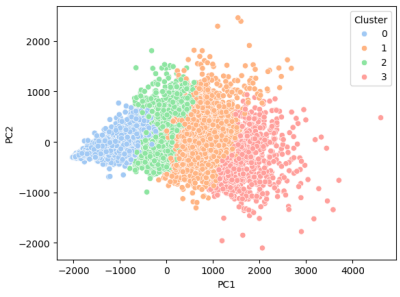


(a) Silhouette

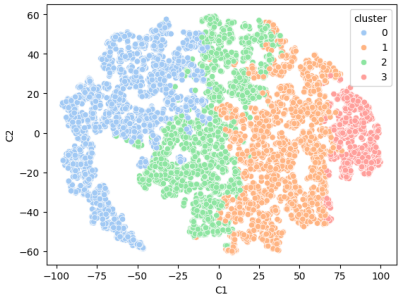


(b) Calinski–Harabasz

Figure 16: Performance scores for several numbers of hierarchical clusters.



(a) PCA



(b) t-SNE

Figure 17: 2-dimensional hierarchical clustering representation

Figure 29 shows the clustering structure in a two-dimensional projection of the data. Thus, it is noticeable that the dimensionality reduction using PCA maintains a well-defined cluster division, whereas, using t-SNE, there are less overlaps reflected compared to the model from K-Means.

3.2.3 Algorithms Performance

It is crucial to assess each clustering algorithm's performance after it has been defined for the case data set in order to choose the best one and apply it for the final cluster determination. The following metrics were employed for this purpose: Calinski-Harabasz index, Silhouette coefficient, and Squared Sum of Errors (SSE). Furthermore, every technique used to the previously described data set enabled the prediction of cluster labeling across the testing dataset. Having said so, the metrics evaluation over the extracted clusters is given below:

Algorithm	SSE	Silhouette Score	Calinski-Harabasz
K-Means	9514562.1350	0.1228	1109.5431
Hierarchical	-	0.2613	6235.1246

Table 1: Clustering evaluation metrics.

The table 1 gives interesting results. First, SSE is not comparable between algorithms because it can not be computed for the Hierarchical approach. On the other hand, the one that maximizes the Silhouette coefficient is Hierarchical Clustering, which means that this approach achieves better cluster separation and cohesion compared to K-Means. Finally, also Hierachical Clustering is the algorithm with the highest Calinski-Harabasz index, which implies that this algorithm forms better shaped-clusters. For segmentation-based decision making, it is essential that the clusters generated by the algorithms are adequately separated and defined, so that the clusters can be characterized more effectively. Therefore, we conclude that *Hierarchical* is the clustering algorithm with the best results on the dataset, in addition to having a big difference in performance in the Calinski-Harabasz index compared to K-Means, it also has the highest Silhouette coefficients among the models evaluated. All of these results are also reflected in the shared features among the clusters and in the two-dimensional projection plots of the PCA and t-SNE techniques.

3.3 Classification

In this section, several classification techniques will be applied on the genre corresponding to each time series. Among the methods used are lazy learners such as K-Nearest Neighbors -both with preprocessed data and with the shapelets- and Machine Learning methods such as ROCKET.

3.3.1 K-Nearest Neighbors

In order to label the observations in the testing set by gender, the Nearest Neighbors classifier was used in two ways. In the first, the time series were used after being transformed with denoising and offset translation and then approximated by PAA. It is important to emphasize that tests were also performed with the series approximated with SAX, but the results were quite poor compared to those obtained with PAA, which will be presented below.

After splitting the dataset between training and testing with a stratified 80:20 ratio split, the distances to be used were determined, which in this case were Euclidean and Dynamic Time Warping. In both cases a Randomized Search with Repeated Stratified K-Fold as the cross-validation generator [11] was used to optimize the number of neighbors, running from $k = 2$ to $k = \sqrt{n}$, since this is the convention for determining a reasonable number of neighbors [9]. In the case of the Euclidean distance, the best results were at $k = 89$.

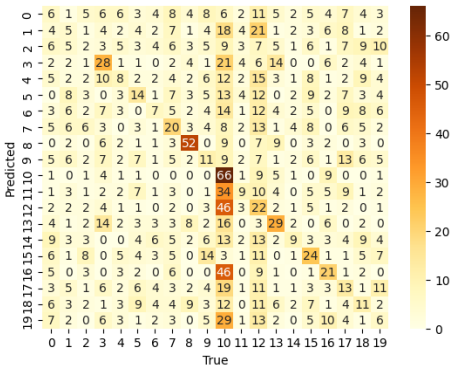


Figure 18: Confusion matrix for KNN using DTW distance.

The confusion matrix shows that no category predicted more than 20% of the observations correctly. In fact, the category with the best classification is *new_age*, with 16.5% of correctly labeled observations, followed by *minimal_techno* with 13% of true positives and -much more distantly- by *progressive_house* with 7.25%, a value that is close to the classification rate of a third of the genres. The remaining ones have less than 3% classification rate, and there are even genres such as *goth* where only one of the observations was correctly labeled.

The results observed in the confusion matrix lead us to conclude that the genres most similar to

electronics are those with the most accurate predictions, and that there is also a misclassification bias towards *new_age* since in all categories more false labels are generated for *new_age* than for the other genres, indicating that its high classification rate is probably due to this bias.

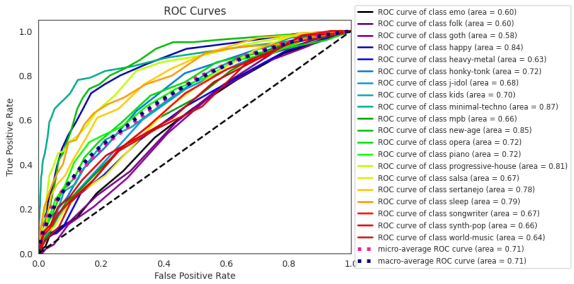


Figure 19: ROC curve for KNN using DTW distance.

ROC Curve analysis is also useful in classification scenarios such as this one, where the data is balanced and stratified. For the KNN using DTW, it can be observed that all genre had a better tradeoff than a base classifier, with areas between 0.6 and 0.85 and a mean of 0.71. Supporting the results observed in the confusion matrix, the Area Under the Curve (AUC) of this technique is better in *minimal_techno*, *new_age*, *happy* and *progressive_house*, being greater than 0.8 in all cases. These genres share several common characteristics, the most remarkable being their high instrumentalness and danceability, as well as being in a similar range of popularity.

	Precision	**Recall**	**F1-Score**
emo	0.07	0.06	0.07
folk	0.08	0.05	0.06
goth	0.04	0.02	0.03
happy	0.25	0.28	0.27
heavy-metal	0.15	0.08	0.10
honky-tonk	0.18	0.14	0.16
j-idol	0.15	0.07	0.10
kids	0.20	0.20	0.20
minimal-techno	0.54	0.52	0.53
mpb	0.13	0.11	0.12
new-age	0.16	0.66	0.26
opera	0.21	0.09	0.13
piano	0.10	0.22	0.14
progressive-house	0.31	0.29	0.30
salsa	0.27	0.09	0.14
sertanejo	0.23	0.24	0.23
sleep	0.25	0.21	0.23
songwriter	0.13	0.13	0.13
synth-pop	0.13	0.11	0.12
world-music	0.08	0.06	0.07

Table 2: Performance metrics for KNN using DTW distance.

Finally, the analysis is complemented by performance metrics. The average accuracy is 0.1815, which for the number of categories to be classified is a relatively good index. However, the generalization capacity of the model is not the same in all categories, where the more synthetic rhythms such as *minimal_techno* and *progressive_house* present better results and the alternative and melancholic rhythms such as *goth* and *emo* have problems to correctly classify the observations. With the exception of *new_age* -probably due to the bias described above-, a quite similar situation is reflected with recall, where the classes that hav1e the lowest AUC are precisely those that have the lowest recall, indicating that the model has difficulties recovering observations that are actually positive. On the other hand, the F1-Score is slightly better for most genres, indicating a balance between precision and recall and maintaining the best performance in the genera with the highest precision index. Generally speaking, the subgenres of electronic music, the fast and catchy songs, but also the relaxed songs such as sleep songs are easier to classify using this model.

For the second application of the algorithm, the logic is basically the same as the first one, with the only distinction that the distance used is Euclidean. In the same way, a randomized search was used to calculate the optimal number of neighbors, concluding that $k = 87$, with an accuracy of 0.1125 is the configuration with the best results and, consequently, the one that was used.

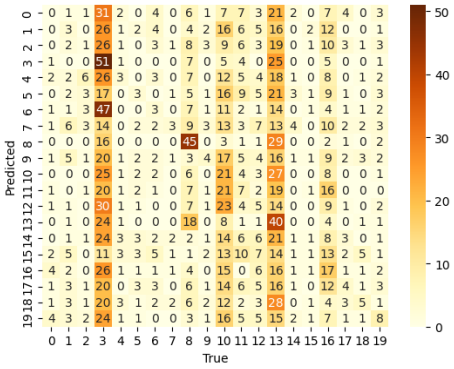


Figure 20: Confusion matrix for KNN using Euclidean distance.

The confusion matrix shows very different results from the previous ones. In this case, a bias is observed in the classifier towards the classes *happy*, *new_age* and *progressive_house*, which can be verified through the sum of the observations of its columns since it is much higher than the others, especially in the case of *happy*. This may happen because part of the series of these classes are in regions of high density, so that the nearest neighbor vote is mostly for these classes and cause an incorrect classification. This may also be caused by the variability of these classes, however the problem is not as noticeable when using DWT, so the notion of distance used is probably a determining factor in the bias. This issue will be discussed further in the comparison of the performance of the algorithms.

The AUC of each of the labels is in all cases better

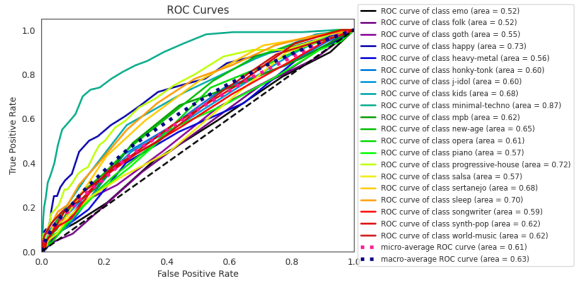


Figure 21: ROC curve for KNN using Euclidean distance.

than that of a random classifier, ranging from 0.52 to 0.87, with most classes between 0.5 and 0.6. The bias found in the confusion matrix is also reflected in Figure 21, where the tradeoff of the four classes mentioned before is above the others.

	Precision	Recall	F1-Score
emo	0.00	0.00	0.00
folk	0.07	0.03	0.04
goth	0.04	0.01	0.02
happy	0.10	0.51	0.17
heavy-metal	0.12	0.03	0.05
honky-tonk	0.12	0.03	0.05
j-idol	0.08	0.03	0.04
kids	0.25	0.03	0.05
minimal-techno	0.28	0.45	0.34
mpb	0.17	0.04	0.06
new-age	0.08	0.21	0.11
opera	0.08	0.07	0.07
piano	0.07	0.05	0.06
progressive-house	0.10	0.40	0.16
salsa	0.06	0.01	0.02
sertanejo	0.08	0.01	0.02
sleep	0.10	0.17	0.12
songwriter	0.14	0.04	0.06
synth-pop	0.23	0.05	0.08
world-music	0.19	0.08	0.11

Table 3: Performance metrics for KNN using Euclidean distance.

The performance measures generally show a not so good performance, where most labels have less than 10% of precision and even in one of them there was not a single correct prediction. Similarly, precision shows that the genres with the best performance were *minimal_techno*, *kids*, *synth_pop* and *world_music*, somewhat different results from the DTW method. As in the previous sections of this analysis, the classifier bias affected the recall measure and therefore also the F1-Score of the favored categories, and given the disagreement between these and the precision we can conclude that it may indicate that the model prioritizes the identification of all positive observations, minimizing false negatives at the expense of false positives.

3.3.2 Shapelets

Another technique used to classify the genre was the extraction of shapelets, which are subseries that represent distinctive features of groups of time series through local patterns [2]. Its discriminative ability makes it suitable for classification purposes, but it can also be used to compute the distances of series from their shapelets and apply them as input data to another classifier. Both scenarios will be explored in this section.

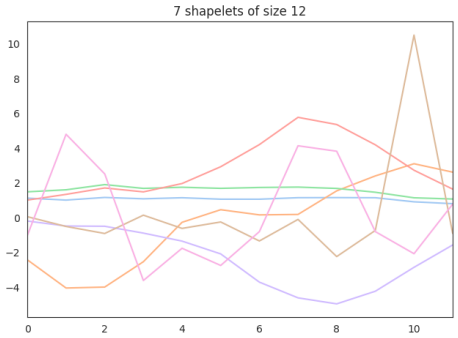


Figure 22: Shapelets.

In shapelet discovery, the dynamic programming method proposed by Grabocka et al. [2] was used to define the size of the optimal shapelets, in this case in the form of a dictionary that maps the length of each shapelet and the number of shapelets corresponding to that length [10]. To do that, the parameters defined were a minimum length of 0.1 which implies the extraction of shapelets that at least describe 10% of the length of each series and a reduction factor of 1 to consider all possible shapelets within the range of the minimum length. From this, 7 possible shapelets of size 12 were defined.

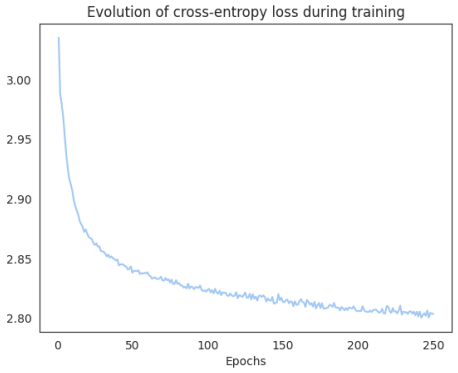


Figure 23: Cross-Entropy loss of shapelets during training phase.

From the resulting dictionary, the LearningShapelets class [10] was used for the learning phase. This model applies a logistic regression layer on a Shapelet Transform, which in our case was optimized with Adam [1] in a L2-penalized cross-entropy loss with a regularization weight of 0.01. In addition, a batch size of 64 and a maximum limit of training iterations of 250 were specified. By using this configuration, the model yielded an accuracy of 0.1435.

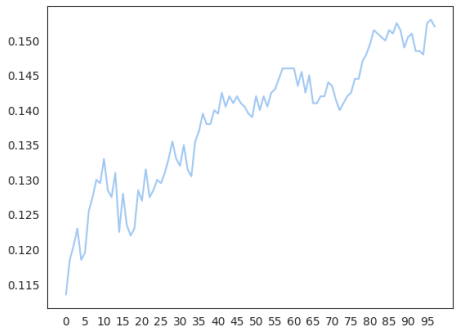


Figure 24: Accuracy of KNN for $k \in [0, 100]$ using shapelets as input data.

Subsequently, the time series of the training set were transformed into distances with respect to the shapelets obtained, in order to use them as input into another classifier, in this case a KNN that was chosen to compare more homogeneously the performance with respect to the rest of the classifiers. To evaluate the number of neighbors that gives the best accuracy, runs were carried out on $k \in [2, 100]$, similarly to what was done in the previous section. In fact, the optimal number of neighbors was $k = 87$, which is close to those obtained previously. For this configuration, the accuracy was 0.2318, which represents a remarkable improvement with respect to classifications using the preprocessed data.

Comparison with Motifs

Once the shapelets are obtained, we can conclude that there are several relationships between shapelets and motif means by gender. In particular, the lavender shapelet is mostly similar to **sertanejo**, indicating that the motif and shapelet share characteristics of cluster 1 obtained from the hierarchical clustering analysis. In addition, the red shapelet has much similarity with *salsa*, which is the main representative of cluster 2; the genre motifs that most closely resemble the pink shapelet are *progressive_house* and *opera*, while the orange shapelet resembles the motif corresponding to *world_music*. On the other hand, the green shapelets and the blue shapelet is very similar to *piano* and *new_age*, which in turn are similar to *sleep*. Both describe cluster 0, which is best described by shapelets and contains songs of long duration and high acousticity. Finally, the brown shapelet could not be associated with any genre because no significant similarities were found with the motifs.

3.3.3 ROCKET

Finally, the ROCKET model uses random convolutional kernels to perform the classification of the dependent variable. To optimize the model, tests were performed with the *rocket* and *minirocket* frameworks. The use of multirocket was avoided because it requires too high computational cost. Thus, the minirocket framework obtained an accuracy of 0.21, with a precision of 0.20, recall of 0.21 and F1-score of 0.20. The results at the genre level are shown below:

	Precision	Recall	F1-Score
emo	0.08	0.07	0.07
folk	0.08	0.07	0.07
goth	0.09	0.07	0.08
happy	0.48	0.66	0.56
heavy-metal	0.10	0.10	0.10
honky-tonk	0.22	0.19	0.20
j-idol	0.16	0.14	0.15
kids	0.13	0.13	0.13
minimal-techno	0.60	0.70	0.65
mpb	0.08	0.06	0.07
new-age	0.25	0.29	0.27
opera	0.17	0.15	0.16
piano	0.12	0.11	0.11
progressive-house	0.39	0.43	0.41
salsa	0.11	0.12	0.11
sertanejo	0.17	0.21	0.19
sleep	0.28	0.28	0.28
songwriter	0.13	0.12	0.12
synth-pop	0.18	0.19	0.19
world-music	0.12	0.12	0.12

Table 4: Performance metrics for ROCKET model using minirocket framework.

It can be noted that the *minimal_techno* and *happy* genres have the best performance results. Specifically, *happy* shows a 0.48 accuracy, 0.66 recall and 0.56 F1-score, while *minimal_techno* has a 0.60 accuracy, 0.70 recall and 0.65 F1-score.

	Precision	Recall	F1-Score
emo	0.07	0.06	0.06
folk	0.08	0.07	0.07
goth	0.04	0.03	0.03
happy	0.49	0.60	0.54
heavy-metal	0.09	0.09	0.09
honky-tonk	0.23	0.25	0.24
j-idol	0.20	0.19	0.19
kids	0.06	0.06	0.06
minimal-techno	0.63	0.72	0.67
mpb	0.08	0.07	0.08
new-age	0.29	0.29	0.29
opera	0.22	0.25	0.24
piano	0.08	0.08	0.08
progressive-house	0.35	0.40	0.37
salsa	0.19	0.20	0.19
sertanejo	0.16	0.16	0.16
sleep	0.35	0.32	0.33
songwriter	0.19	0.17	0.18
synth-pop	0.26	0.27	0.26
world-music	0.13	0.13	0.13

Table 5: Performance metrics for ROCKET model using rocket framework.

On the other hand, when using the rocket framework, an accuracy of 0.22, precision of 0.21, recall of 0.22 and F1-score of 0.21 are obtained. As seen in Figure 7, the *minimal_techno* and *happy* genres present the best classification performance results. In fact, *minimal_techno* has an accuracy of 0.63, recall of 0.72 and F1-score of 0.67, while *happy* has an accuracy of 0.49, recall of 0.60 and

F1-score of 0.54.

In general terms, a better performance is observed in the rocket framework than in the minirocket framework, both at the general level and at the level of prediction by gender.

3.3.4 Algorithms Performance

After applying all the algorithms, we have evaluated the performance of each in time series gender labeling. The evaluation is based on four performance metrics that will be used throughout the project. The results obtained for each method are presented below:

	Accuracy	Precision	Recall	AUC
KNN DTW	0.1815	0.1855	0.1815	0.71
KNN Euclidean	0.1125	0.1180	0.1125	0.63
Shapelets	0.1435	-	-	-
MiniROCKET	0.2100	0.2000	0.21	-
ROCKET	0.2200	0.2100	0.2200	-

Table 6: Classification evaluation metrics.

Based on the results of the 6 table, it can be observed that ROCKET and MiniROCKET, and especially the ROCKET framework, are the best methods for classification in all metrics, which can be attributed to their discriminative ability. On the other hand, the KNN using Dynamic Time Warping shows reasonable performance, especially in terms of AUC, which indicates a good discrimination between classes but not better than that of the ROCKET methods. On the other hand, the KNN using Euclidean distance has the worst performance, which is attributed to the incapacity of the measure to adequately handle the temporal characteristics and variability of the data. Finally, the KNN using as input the distances obtained from shapelets have intermediate performance, despite the fact that DTW was used as the distance measure.

4 Advanced Data Pre-processing

In this section we will discuss various algorithms that allow the detection of outliers throughout the dataset, as well as the classification of originally unbalanced attributes.

4.1 Outliers Detection

Detecting outliers means finding data points that are very different from the norm. This is important for better data analysis. In this section, we aim to find the top 1% of outliers in the data using several methods.

4.1.1 Visual Based Approach

For this approach, the Histogram Based Outlier Score (HBOS) algorithm is used. For this purpose, a contamination percentage of 0.01 was set to detect the top 1% of outliers in the dataset.

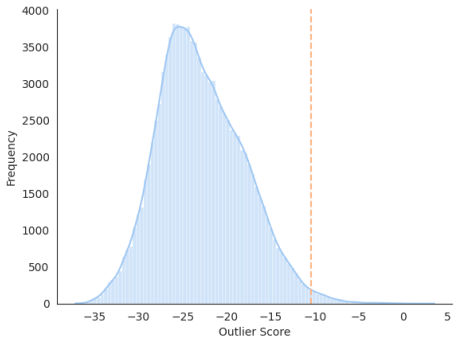
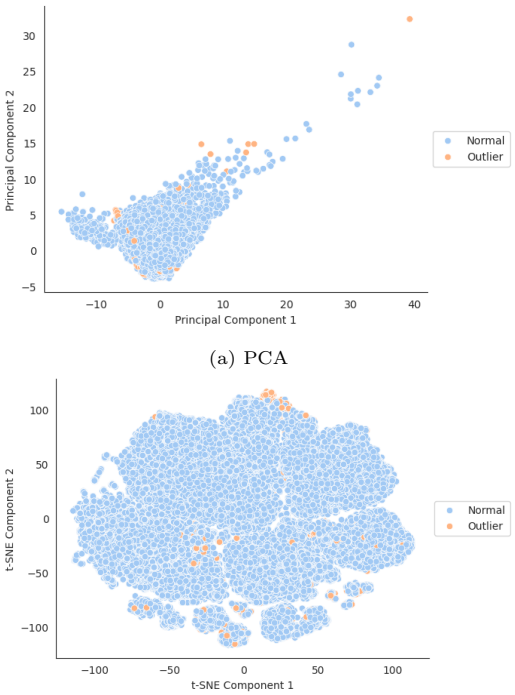


Figure 25: Outliers detection using HBOS.

As can be seen in Figure 25, 1% of outliers are determined approximately at outlier score values greater than -10. It is also worth mentioning that 1091 records considered as outliers were detected.



(a) PCA

(b) t-SNE

Figure 26: Bidimensional representation of outliers with HBOS.

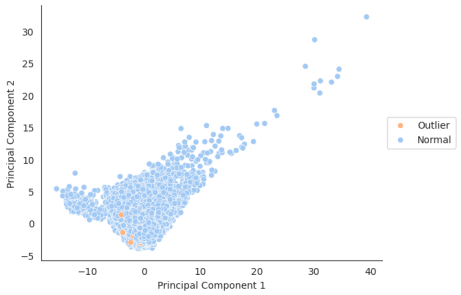
The PCA plot of the data set reflects that the outliers were detected mainly in areas of low point density, moving away from the main density of the data, which means that the model has done a good job of outlier detection.

On the other hand, when analyzing the dimensionality reduction with t-SNE, it is observed that most of the outlier points are distributed

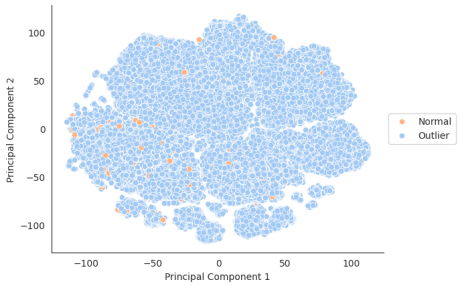
in clusters. Thus, the cluster located in the highest values of the second component and in average values of both components stands out. Other outliers are scattered in low values of the second component.

4.1.2 Density Based Approach

For this approach the Local Outlier Factor algorithm is used, which considers as outliers the data that have a considerably lower density compared to their neighbors [5].



(a) PCA



(b) t-SNE

Figure 27: Bidimensional representation of outliers with LOF.

The scatter plot with dimensionality reduction using PCA distinguishes outliers mainly in the lower part of the data distribution, where the values of the second principal component are lower. On the other hand, dimensionality reduction with t-SNE shows a larger scatter of outliers along the whole data distribution, especially at the lower values of the first component.

4.1.3 High Dimensional Approach

The high dimensional approach chosen is the Angle-Based Outlier Detection (ABOD), which is a technique based on the idea that points that are outliers are in geometrically unusual positions compared to points that are not outliers. This approach involves the angles between the vectors connecting a point to each pair of points in the data set, and their variance is used.

In our project, an ABOD with a contamination rate of 0.01 was run to extract the top 1% of outliers. However, the results were not positive since the outliers were not clearly visible in the two-dimensional representation and were confused with the inliers, so they will not be presented in this report. Some of the reasons why this method may have failed were the high dimensionality of the dataset, or the non-sphericity of the data set.

4.1.4 Model Based Approach

As a model-based approach, the Isolation Forest was used, which detects anomalous events using the concept of isolation, and more specifically, the idea that outliers are easier to isolate than normal points. To apply this technique, a contamination value of 0.01 was considered, so that 1% of the observations are expected to be identified as outliers. In addition, the number of maximum samples to build each tree was defined as $auto = min(256, n\_samples)$, where $n\_samples$ is the number of observations in the dataset [5].

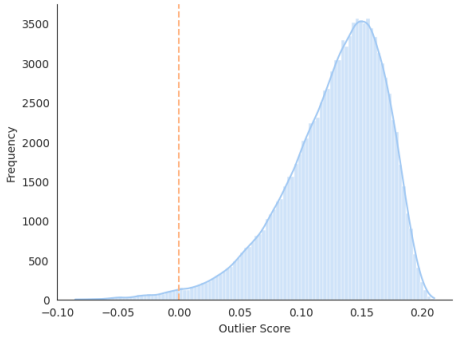


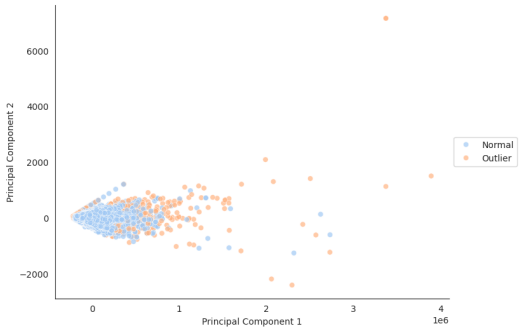
Figure 28: Outliers Detection with Isolation Forest.

Given the threshold at the top 1% of the observations, we can see the distribution of the data, which shows us many more values close to the threshold than at the left end of the distribution. In total, 1091 observations were found to be outliers.

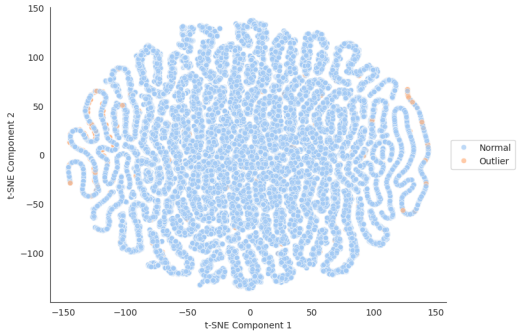
As can be seen in the PCA representation, the outliers are mainly scattered outside the accumulation of inliers. Particularly, these are located to the right and above the main cluster, indicating a deviation from one of the principal components. In addition, several of the points are scattered, which may suggest that there are considerable variations and that therefore, different types of outliers may exist.

On the other hand, the figure containing the t-SNE shows a large number of outliers on the outer edge of the mass of dots, although some are not very visible because they are superimposed by inliers. In addition, there are some anomalous points scattered along the right edge of the structure, suggesting that they are located in less dense regions.

While the PCA shows a clearer separation of the outliers in relation to the maximum variance,



(a) PCA



(b) t-SNE

Figure 29: Bidimensional representation of outliers with ABOD.

they are further away from the main directions of most of the data. On the other hand, in the t-SNE the structure of the data is reflected in more detail, and the outliers are more identifiable since most of them are not integrated with the normal points.

4.1.5 Algorithms Performance

In the HBOS method, the outliers are grouped, indicating the presence of different clusters of anomalies, which can be beneficial. However, some outliers are dispersed within the data structure. Additionally, while the distribution of the scores is reasonable, it is not significantly distant from the mean, making it challenging to identify outliers clearly. Conversely, in the LOF method, there is substantial overlapping, with outliers being very dispersed and lacking a distinct structure. In contrast, the Isolation Forest method shows a well-structured pattern; most outliers are clustered in a specific area, separate from the main data structure, as evidenced in both the PCA and t-SNE visualizations. Therefore, based on these visualizations, we conclude that the Isolation Forest is the most effective method.

4.1.6 Dealing with Outliers

There are several ways to deal with outliers, among which are the data imputation and the removal of these values. In our case, the first intuition was to remove outliers since, as the dataset is large, deleting 1% of the observations should not significantly affect performance when they are outliers. In addition, imputation was not considered as it is a more complex process and involves introducing synthetic data, which could affect the results in unbalanced variables.

First, it is important to note that the outliers to be worked on are those obtained by Isolation Forest. To confirm this intuition, we applied a Decision Tree classifier with a $min\_samples\_leaf = 3$ to predict the gender label, both in the original data, in a new dataset without outliers, and in another where outliers were determined as missing values and imputed with the mean of each variable. The results are shown below:

	Accuracy	**Precision**	**Recall**	**F1-Score**
Original	0.9215	0.9182	0.9223	0.9200
Removed	0.9119	0.9085	0.9165	0.9108
Imputed	0.9119	0.9167	0.9205	0.9183

Table 7: Performance metrics for Outliers Detection using a Decision Tree.

The above table shows that the performance is almost the same between the three datasets, but it is slightly higher in the original dataset. Making the comparison between outlier removal and outlier imputation, the imputation is a little better, but we stick with the decision to remove the outliers because the difference in performance is less than 0.1% and this tradeoff does not justify the possible creation of synthetic variables in case of imputing the outliers.

4.2 Imbalanced Learning

Since the present work is focused on finding the main determinants for the popularity success of songs uploaded to Spotify, it was decided to use *top_track* as a predictor variable to predict which songs are considered mainstream in the dataset, considering that *top_track* is a binary variable with a ratio of 99% and 1%. To avoid using redundant variables in the model, the variable *popularity* was removed from the set of independent variables.

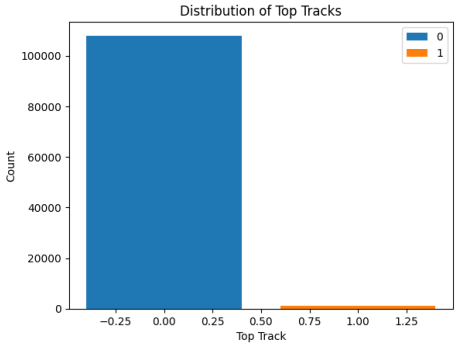


Figure 30: Frequency of top_track variable

To begin with, a preliminary K-Nearest Neighbors model was performed with the originally unbalanced data. Thus, a model with an accuracy of 0.99 was obtained. However, the model had an accuracy, recall and F1-score of 0.00 for category 0 of the *top_track*, which was completely expected due to the distribution of this variable. In fact, as shown in Figure 31, the model did not classify any of the records as 1 in the dependent variable.

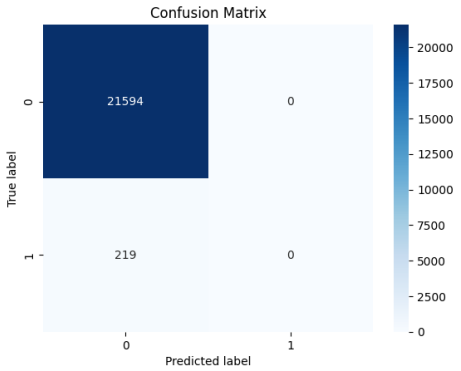


Figure 31: Confusion matrix for the preliminar KNN with imbalanced dataset.

To correct this classification error, oversampling and undersampling techniques were used to make the data set more balanced.

4.2.1 Undersampling

First, the Random Undersampler algorithm was used, which randomly reduces the number of instances of the majority class to balance the dataset. The application of this algorithm resulted in a balanced dataset of 878 observations for each class of the variable to be classified.

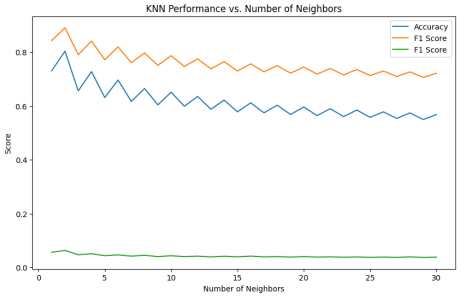


Figure 32: KNN performances vs Number of neighbors by Random Undersampling.

Next, to determine the optimal number of neighbors in the model, several tests were performed to determine which model generated the highest accuracy and precision, especially for the originally unbalanced class. Thus, it was obtained that the model generated better results with 2 Nearest Neighbors.

This configuration of hyperparameters generated a model with an accuracy of 0.803, in addition to an average precision, recall and F1-score of 0.51, 0.73 and 0.48, respectively. The originally unbalanced class obtained an accuracy of 0.03, recall of 0.66 and F1-score of 0.06, suggesting that the use of the Random Undersampler improved performance, especially in the minority class.

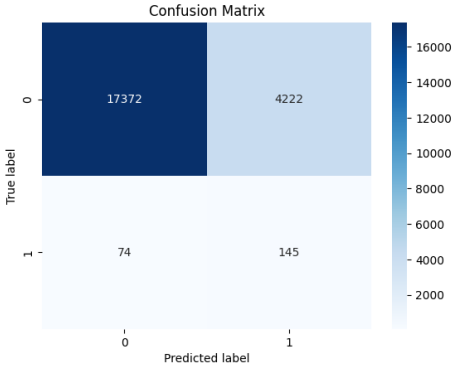


Figure 33: Confusion matrix for KNN with Random Oversampling

As seen in 33, the model correctly predicted 145 observations of the unbalanced class, while it missed 74 of the records, which is a significant improvement compared to the original dataset but does not represent an effective performance.

Later, the same procedure was performed using Tomek Links, an algorithm that identifies and eliminates ambiguous examples from the majority class when they are close to the decision boundary. It generated a new training dataset with 86253 observations for class 0 -no *top_track*- and 878 observations considered mainstream.

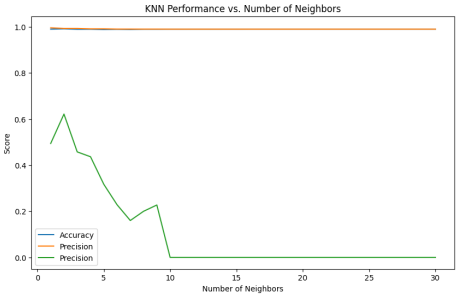


Figure 34: KNN performances vs Number of neighbors - Tomek Links

Figure 34 shows the performance indicators of the KNN model from various neighbor number configurations. It can be seen that the model generates better results when 2 nearest neighbors are used. Therefore, this configuration will be used in the classification model.

Using the previously discussed hyperparameter configuration, the model yielded an accuracy of 0.991, with an average precision, recall and F1-score of 0.81, 0.66 and 0.71, respectively. Specifically, the unbalanced class obtained an accuracy

of 0.62, recall of 0.32 and F1-score of 0.42. As can be seen in Figure 35, the model made classification errors in 192 observations, 42 for no *top_tracks* and 150 for *top_tracks*.

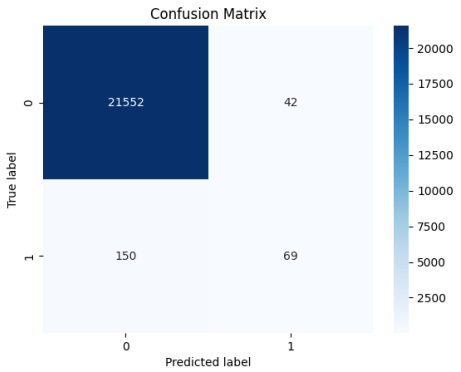


Figure 35: Confusion matrix KNN classification using Tomek Links

4.2.2 Oversampling

The first oversampling algorithm used was SMOTE, a technique that generates new synthetic instances of the minority class to balance the classes. From this algorithm, a data set with 86373 observations was generated for both class 0 and class 1.

Figure 36 shows the performance indicators of the model for different values of nearest neighbors. Thus, it is observed that the best classification results are obtained with 2 nearest neighbors. Therefore, this will be the hyperparameter configuration to be used into the KNN model.

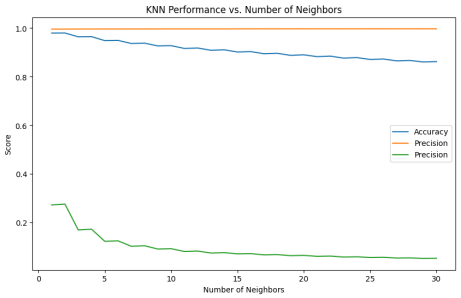


Figure 36: KNN performance vs Number of neighbors - SMOTE

The classification model yielded an accuracy of 0.98, with an average precision, recall and F1-score of 0.64, 0.80 and 0.69, respectively. As for the originally unbalanced class, an accuracy of 0.28, recall of 0.61 and F1-score of 0.38 were obtained. Furthermore, as seen in 37, the model made incorrect classifications on 436 observations, from which 351 were incorrectly classified as class 1 and 85 registries were incorrectly classified as class 0.

Following the same logic as above, the ADASYN

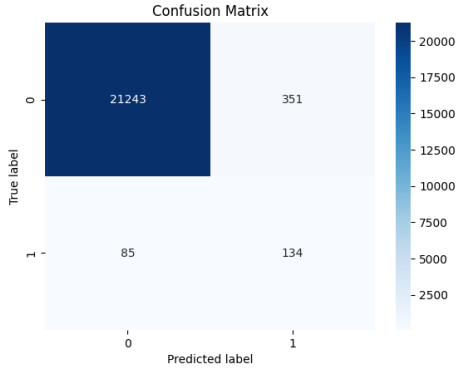


Figure 37: Confusion matrix KNN using SMOTE

method was used, which is an extension of SMOTE that not only generates synthetic samples of the minority class but also focuses on generating samples for difficult decision regions. The application of this algorithm generated a data set of 86490 class 0 observations and 86373 class 1 observations.

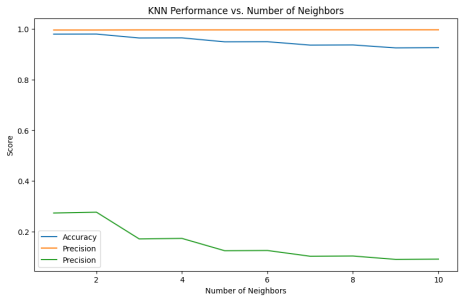


Figure 38: KNN performances vs Number of neighbors - ADASYN

To identify the ideal number of nearest neighbors, the performance indicators of the model with various numbers of nearest neighbors are analyzed. Thus, as shown in figure 38, the best accuracy and precision results are obtained with 2 nearest neighbors. Therefore, this will be the hyperparameter configuration to be used. Once the parameters are defined, the KNN model yields an accuracy of 0.98, with an average precision, recall and F1-score of 0.28, 0.61 and 0.38 respectively. As for the originally unbalanced class, an accuracy of 0.28, recall of 0.61 and F1-score of 0.38 were obtained.

4.2.3 Algorithms Performance

It is evident that the model that has yielded the best results is Tomek Links, using the undersampling approach. Therefore, the dataset obtained from this algorithm will be used in future classification tasks based on the search for mainstream songs within the dataset.

5 Advanced Machine Learning & Explainable Artificial Intelligence

5.1 Advanced Classification

5.1.1 Logistic Regression

The logistic regression model is a very generic model, and its application can be extended from binary classification to multiclass classification, where a multinomial logistic regression [5] is used. Thus, it was decided to use this model under the one-vs-rest (OvR) [5] scheme to classify the musical genres to which the different records of the song table belong.

In order to obtain the best possible results in this classification model, a random search of hyperparameters was used to maximize the accuracy score. The hyperparameters considered in this search are:

- C, to parameterize the regularization level of the model.
- *solver*, which is the optimization algorithm of the model.
- *penalty*, which specifies the penalty rule used to avoid overfitting.

After an hyperparameter search, the optimal configuration of parameters was $C = 1$, $solver = lbfgs$ and $penalty = l_2$. Here, l_2 was used as the penalty parameter because it had compatibility with all available optimization algorithms. In addition, when experimenting with the logistic model, it was observed that adjusting the weight of the genders according to their frequencies using *class_weight* worsens the prediction results, so it was decided to ignore this hyperparameter.

The logistic regression model, with the hyperparameter configuration described above, yielded an accuracy of 0.23, a precision of 0.20, a recall of 0.22 and an F1-score of 0.19. These results reflect a good performance of the model, considering that there are 114 musical genres to be predicted. It should also be noted that the *comedy* and *sleep* genres have the highest accuracy scores -0.70 and 0.66 respectively- and F1-score -0.78 and 0.70-. As for recall, *study* is the genre with the highest score, with 0.80. In contrast, the genera: *blues*, *british*, *edm*, *folk*, *indie_pop* and *swedish* present 0 in all the scores analyzed, implying that the algorithms do not classify any observation correctly for these genres.

As can be seen in Figure 39, the model presents a good ability to discriminate between music genres. Specifically, the model presents an AUC score of 0.891. Moreover, the genres with the highest AUC are *deep_house* with 0.98, *black_metal* with 0.97 and *electro* with 0.98. On the other hand, the genres with the lowest AUC score are *german* and *spanish*, both with 0.74.

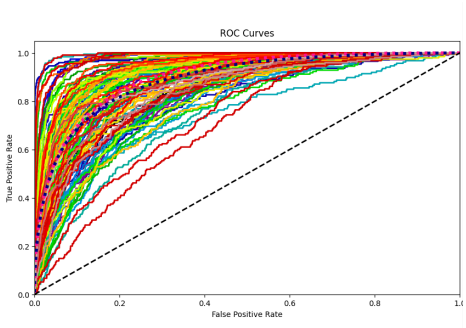


Figure 39: ROC curves for logistic regression model.

5.1.2 Support Vector Machines

Unlike the logistic regression model, Support Vector Machines use a one vs. one scheme for the prediction of multiclass attributes [5]. In this classification algorithm, the parameters to be considered are:

- C, to optimize the regularization strength of the model.
- *kernel*, to define the kernel function to be used.
- *gamma*, to define the influence coefficient of a single training example on the total model.

To define the optimal hyperparameters, a randomized search method was used to analyze the combinations of various parameter values to obtain the choice of hyperparameters that maximizes the model accuracy score. Thus, it was obtained that the hyperparameters that maximize the model results are: $kernel = rbf$, $gamma = 0.1$ and $C = 100$. On the other hand, as in the logistic regression model, it was shown that the weight balancing of the classes to be predicted worsens the classification results.

With the hyperparameter configuration discussed above, we obtained an accuracy, precision, recall and F1-score of 0.26. At the music genre level, it is observed that *comedy* and *sleep* present the highest levels of all the performance indicators of the model. Specifically, *comedy* has a score of 0.83, 0.86 and 0.85, while *sleep* maintains a score of 0.81, 0.78 and 0.80 in precision, recall and F1-score, respectively. In contrast, *songwriter* is the lowest performing genre, with a score of 0.1 on all model performance indicators.

As can be seen in Figure 40, the SVM model does a good job in discriminating the different genders. Specifically, the lowest score is 0.79 for the genre *groove*, while the genres with the highest scores are *study*, has a perfect score, while *comedy* and

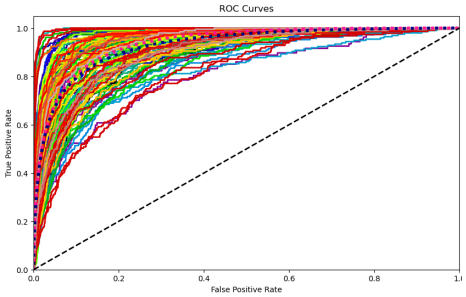


Figure 40: ROC curves for Support Vector Machine.

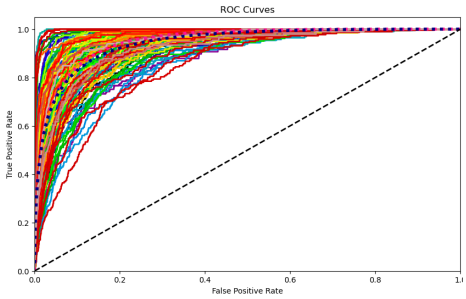


Figure 41: ROC curves for Multi-Layer Perceptron model

chicago_house have both 0.99 as AUC. Overall, the AUC score of the model is 0.928.

5.1.3 Neural Networks

In this section, the Multi-layer Perceptron classifier is implemented to classify the different music genres. For this purpose, the following hyperparameters are considered:

- *hidden_layer_sizes* to determine the neuron number in each hidden layer that the model will have, avoiding overfitting of the model.
- *alpha*, that will be configured to increase the regularization of the model in order to control a possible overfitting.
- The activation function, which will be chosen using the parameter *activation* and the optimization algorithm *solver*.
- *learning_rate* optimization, that will be considered to define the learning rate policy that optimizes the model.

From the above, a random search for the optimal hyperparameters is executed, in order to maximize the accuracy of the model. Thus, the best hyperparameters found are: $solver = adam$, $learning\_rate = adaptive$, $hidden\_layer\_sizes = (100, 100, 100)$, $alpha = 0.01$ and $activation = logistic$. Additionally, the early stopping parameter is activated to avoid the overfitting of the model.

Based on the parameters obtained above, we executed a Multi-Layer Perceptron model, which shows an accuracy of 0.32, precision of 0.31, recall of 0.32 and F1-score of 0.30. It means that the model does a good job of classification, considering the number of classes in the variable to be predicted. Regarding the scoring by genre, it can be observed that *comedy* and *sleep* have the best results in recall, with 0.87 and 0.84, respectively; and F1-score with 0.86 and 0.79, respectively. In contrast, *singer_songwriter* shows the worst results, with a recall of 0.01 and F1-score of 0.02.

As visualized in Figure 41, the model does a great job in discriminating its selection between music genres. In fact, *grindcore*, *sleep* and *study* are the genres with the highest AUC score, with 1.00 for each. On the other hand, *spanish* is the genre with the lowest AUC score with 0.85, that is still better than the AUC for a random model. Overall, this algorithm produces an AUC score of 0.928.

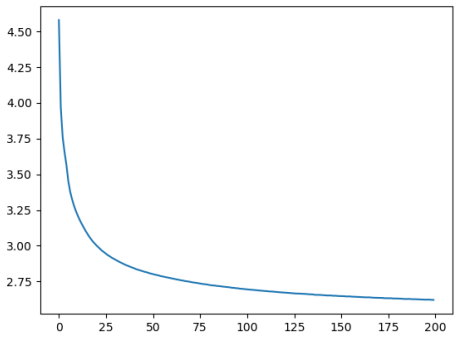


Figure 42: Loss curve for MLP model.

Finally, as seen in Figure 42, the model is learning effectively, as the loss is steadily decreasing. This indicates that the model is being properly trained and is improving its ability to fit the training data. Moreover, it can be observed that there are no overfitting problems in the model, as the results of the model in the training set are not too far from the results in the test set. Specifically, in the training set there is an accuracy of 0.35, while in the test set the accuracy is 0.32.

5.1.4 Ensemble Methods

The ensemble method chosen to classify the musical genres is Random Forest. For this purpose, the following hyperparameters will be taken into account:

- *n_estimators*: number of decision trees used in the forest.
- *criterion*: the function to measure the quality of the decision tree splits.
- *max_depth*: maximum depth of the decision trees. Here, excessive depth could cause an overfit of the overall model.
- *min_samples_split*
- *min_samples_leaf*

- *max_features*

All of the above parameters were inserted into a random search function to determine the hyperparameter configuration that allows the highest accuracy. Thus, it was obtained that with $n\_estimators = 100$, $min\_samples\_split = 10$, $min\_samples\_leaf = 4$, $max\_features = log_2$ and $max\_depth = 15$ and $criterion = gini$ the best results were obtained.

Thus, the previously mentioned hyperparameter configuration obtained an accuracy of 0.31, a precision of 0.30, recall of 0.31 and F1-score of 0.29. This reflects a good prediction job of the model, considering the high number of classes in the variable to be predicted.

Regarding the results by musical genre, it is observed that *comedy* and *sleep* have the best results in precision with 0.88 and 0.85, respectively; and F1-Score with 0.87 and 0.84, respectively. On the other hand, *study* and *comedy* have the best results in recall, with 0.9 and 0.86, respectively. In contrast, the genres *dubstep* and *edm* have 0 in all the performance metrics evaluated above.

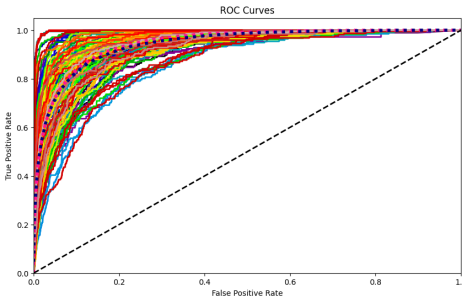


Figure 43: ROC curves for random forests.

Reviewing the ROC curves in Figure 43, it can be seen that the model does a good job in discriminating the different genres. In fact, the AUC scores of all genres are higher than 0.85, even in those genders with 0 in all previous scores. It can also be observed that overall, the model has an AUC score of 0.942.

On the other hand, the analysis of the importance of the variables used in the random forest model shows that the most important variables are, in descendent order:

1. *popularity*: 0.120
2. *acousticness*: 0.082
3. *danceability*: 0.075
4. *speechiness*: 0.072
5. *valence*: 0.066
6. *duration_ms*: 0.065
7. *loudness*: 0.065

From these results it can be concluded that the determination of the musical genre of each song, according to the random forest model constructed, is based mainly on musical and popularity attributes. More specifically, *popularity*, *acousticness*, *speechiness*, *duration* and *loudness* play a determining role in selecting the genre to which each song belongs. Other attributes, such as the *album_type* and the *track_number* have a low contribution to the classification model, which makes sense considering that these attributes do not interfere with the quality or sound of the song, only perhaps with the way in which the song is promoted.

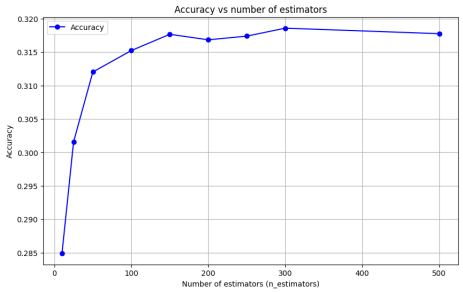


Figure 44: Accuracy vs number of estimators.

Finally, as shown in figure 44, the accuracy shows a constant growth until it reaches 100 estimators. After this figure, this indicator has an irregular trend. This means that, at a certain point, even though the number of estimators to the model continues to increase, the performance of the model will not necessarily improve.

5.1.5 Gradient Boosting Machines

In this section, the XGBoost model is chosen to classify the musical genres to which the different songs in the dataset belong. To optimize the results of this classification model, the parameters to be used are:

- *n_estimators*.
- *learning_rate* to regulate the contribution of each individual model in the algorithm, avoiding possible overfitting.
- *max_depth*.
- *gamma*.
- *reg_alpha*.
- *reg_lambda* to set regularization parameters L_1 and L_2, respectively.
- *objective*, to adjust the objective function.

As it can be observed, some of the hyperparameters are the same as those of the random forest.

Next, a random search for the combination of parameters that optimize the performance of the XGBoost model from 5-fold cross validation is performed. The following values for the previously mentioned hyperpa-

rameters are obtained from this search are: *reg_lambda* = 0, *reg_alpha* = 0.1, *objective* = *ova*, *n_estimators* = 300, *max_depth* = 10, *learning_rate* = 0.1 and *gamma* = 0.

With this configuration, the XGBoost model returns an accuracy of 0.334, as well as a precision, recall and F1-score of 0.33, which represents a good performance considering the large number of classes in the variable to be predicted. Regarding the performance by gender, *comedy* and *sleep* stand out as having the highest performance in accuracy, with 0.89 and 0.91, respectively; and F1-score, with 0.89 and 0.87, respectively. As for recall, *comedy* and *grindcore* have the highest scores, with 0.88 and 0.86, respectively. In contrast, *indie* and *songwriter* are the genres with the lowest performance in these indicators, where each one has 0.02 as accuracy, recall and F1-score.

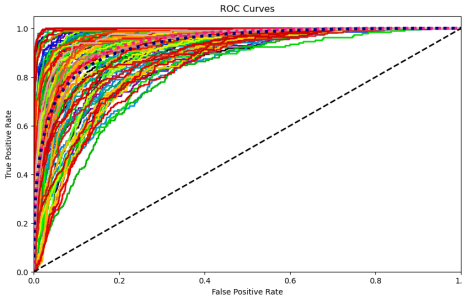


Figure 45: ROC cuves for XGBoost model.

As can be seen in Figure 45, this model does a good job of distinguishing between positive and negative classes. In fact, all genres perform 0.81 or higher on this indicator. The performance of comedy stands out, with a score of 1.0. Overall, the AUC score of the XGBoost model is 0.934.

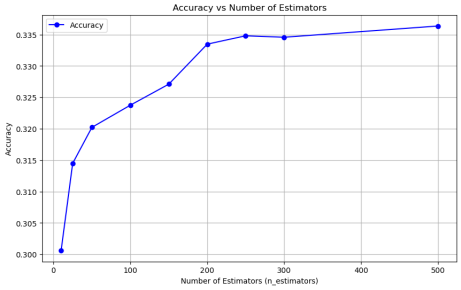


Figure 46: Number of estimators vs accuracy of XGBoost

On the other hand, figure 46 shows that the model shows a relatively high level of accuracy from the first 10 estimators. On the other hand, it is noticeable that up to 250 estimators, the accuracy presents a continuous growth, and then it generates an irregular behavior. It can be concluded that XGBoost tends to learn faster than other ensemble models, such as Random Forest.

5.1.6 Algorithms Performance

Overall, all models do a good job of prediction, considering that there are 114 different genres to be classified. The summary of the performance measures can be found below:

	Accuracy	Precision	Recall	AUC
Logit	0.23	0.20	0.22	0.89
SVM	0.26	0.26	0.26	0.928
ANN	0.32	0.31	0.32	0.928
Random Forest	0.31	0.30	0.31	0.942
XGBoost	0.34	0.33	0.313	0.934

Table 8: Performance metrics for Classification models.

Among all the classification models presented, XGBoost demonstrates the best overall performance. This model has the highest numbers in accuracy, with 0.34, precision, with 0.33, and recall, with 0.313, indicating its ability to correctly predict both true positives and true negatives with high precision.

However, the other models do not differ significantly from the above mentioned results. For example, Random Forest shows a very close performance with an accuracy of 0.31, a precision of 0.30, and a recall of 0.31. What particularly stands out about Random Forest is its AUC Score of 0.942, which the highest among all models, indicating a good capability in distinguishing between classes. The Neural Network (NN) model also demonstrates good performance, being followed to the SVM model. Besides, the Logit model, although with lower performance, still maintains a reasonably good results, but far from the others.

On the other hand, in all classification models the genres with the best prediction results are *comedy* and *sleep*. In contrast, the genre *songwriter* is the most likely to have the lowest metrics.

5.2 Advanced Regression

For the advanced regression algorithms, two different approaches will be used: Support Vector Regressor (SVR) and XGBoost. In this task, we will try to predict the *popularity* variable, so that we can define which variables have the greatest influence on the prediction of the level of popularity. For this purpose, the variable *top_track* is omitted.

5.2.1 Support Vector Regressor

A random search using 5-fold cross validation on several hyperparameters was used for this model. Specifically, the parameters to be optimized were:

- *gamma* and *kernel* to efficiently fit the kernel function and the training coefficient of the

kernel function.
- C to define the optimal regularization level of the model.

Thus, the random search method yielded that the best hyperparameter settings are: $kernel = rbf$, $gamma = scale$ and $C = 0.1$.

Next, using the hyperparameter settings described above, the Support Vector Regressor model was run and the following results were obtained:

1. **Mean Squared Error (MSE):** 0.0310
2. **Mean Absolute Error (MAE):** 0.1385
3. **R-Squared:** 0.4104

From the results, it can be mentioned that the model explains approximately 41.04% of the variability in the output data. In addition, it is notable that the model has a moderate capacity to capture the relationship between the independent variables and *popularity*.

5.2.2 XGBoost

Considering that this model showed the best results in classification, it was decided to use it also for the regression tasks. In this sense, the following hyperparameters were considered to be optimized, so that the model can deliver the best possible results:

- *objective* to set the target function to be used during model training.
- *max_depth* to avoid possible overfitting of the trees used in the model.
- *learning_rate*, *gamma* y *tree_method* to define the algorithm to be used for building the trees.

Thus, a random search function was used to find the best configuration of the hyperparameters described above. As a result, it was obtained that the best configuration is: $tree\_method = exact$, $objective = reg : logistic$, $max\_depth = 8$, $learning\_rate = 0.3$ and $gamma = 0.5$. In addition, RMSE minimization was set as an evaluation criterion. From this configuration, the model yielded:

1. **Mean Squared Error (MSE):** 0.0268
2. **Mean Absolute Error (MAE):** 0.1231
3. **R-Squared:** 0.4903

As can be seen in Figure 47, album type plays an important role in predicting the level of song popularity. Specifically, album type based on song compilations turns out to have a significance coefficient of 0.835. The second most important variable in the model refers to the number of songs on the album in which it appears. Thus, we can

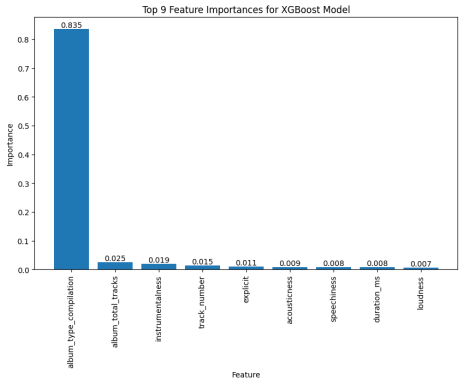


Figure 47: Feature importances of XGBoost regressor.

see that the configuration of the album in which the songs seems to be significant for the model's prediction of popularity.

5.2.3 Algorithms Performance

The following table presents a comparison of the performance of the Support Vector Regressor (SVR) and XGBoost regression models, highlighting the key statistics that reflect each model's effectiveness in correctly modelling an outcome:

	MSE	**MAE**	**R-Squared**
SVR	0.0310	0.1385	0.4104
XGBoost	0.0268	0.1231	0.4903

Table 9: Performance metrics for Regression models.

Having obtained the results of both regression models, we can conclude that the XGBoost model has yielded better results than the Support Vector Regressor model. XGBoost has lower error statistics and a higher R-squared coefficient, which means that this model gives a better indication of the variability of the model with respect to SVR. Thus, XGBoost demonstrates a superior ability to model the relationship between the independent variables and the popularity of songs.

5.3 Explainability

For this section, it was decided to generate a classification model based on random trees to predict the songs considered mainstream, using the variable *top_track*. For this purpose, as discussed in the imbalanced learning section, the dataset generated by the Tomek Links algorithm will be used. The model generated from this data set will be the black box model that will be explained by explainability algorithms.

Specifically, the LORE algorithm was used to identify the reasons that lead the model to correctly identify the songs in the highest popularity percentile. To achieve this goal, a random

sample of 15 records was obtained, which correctly predicted class 1 of the variable *top_track* with different associated musical genres. As for the hyperparameter settings, all were left at the original levels.

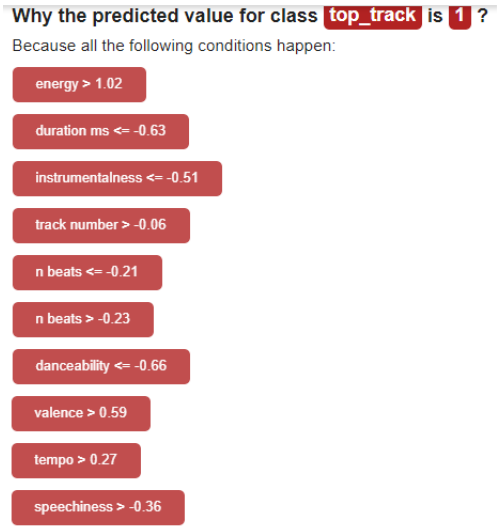


Figure 48: LORE explainability rules.

As shown in Figure 48, the model results show that, according to the black box model used, songs in the highest percentile of popularity are characterized by high levels of *energy*, low *duration_ms* and *instrumentalness*, and finally slightly above average levels of *valence* and *tempo*.

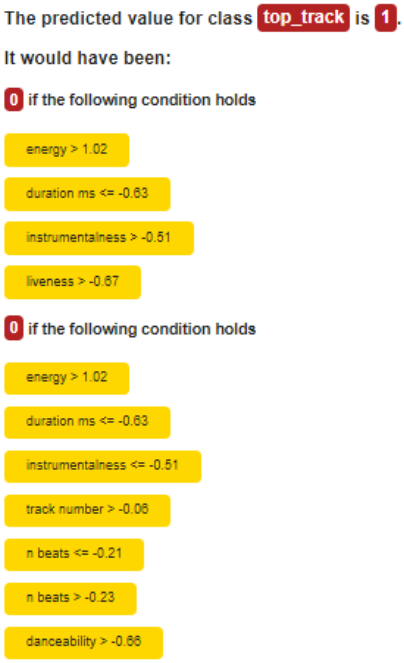


Figure 49: LORE counterfactual rules.

Finally, as shown in Figure 49, average or high levels of *instrumentalness* and *danceability* cause the model to classify the observations as not belonging to the mainstream, despite the above rules being met. Thus, it can be concluded that for a song to be more likely to be successful, it should contain high levels of *energy* but low *danceability*, with a low duration and a faster than average *tempo*. In addition, instrumental and long songs should be avoided.

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), `https://www.tensorflow.org/`, software available from tensorflow.org

[2] Grabocka, J., Schilling, N., Wistuba, M., Schmidt-Thieme, L.: Learning time-series shapelets. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (08 2014)

[3] Kobylin, O., Lyashenko, V.: Time series clustering based on the k-means algorithm. Journal La Multiapp 1, 1–7 (12 2020)

[4] Music, T.: If you're a musician, chances of being totally undiscovered are slim, new study finds (01 2014), `https://www.musictimes.com/articles/3563/20140121/youre-musician-chances-totally-undiscovered-new-study.htm`

[5] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011)

[6] Podolak, B.: How will my music sound on spotify? (2023), `https://www.izotope.com/en/learn/how-will-my-music-sound-on-spotify.html`

[7] Renard, X.: Time series representation for classification: a motif-based approach. Ph.D. thesis, Université Pierre et Marie Curie - Paris VI (2017), ffNNT : 2017PA066593, fftel-01922186

[8] Spotify: About spotify (2023), `https://newsroom.spotify.com/company-info/`

[9] Tan, P.N.: Introduction to Data Mining: Second Edition. Pearson, 2nd edn. (2018)

[10] Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K., Woods, E.: Tslearn, a machine learning toolkit for time series data. Journal of Machine Learning Research 21(118), 1–6 (2020), `http://jmlr.org/papers/v21/20-091.html`

[11] TURING: Different types of cross-validations in machine learning and their explanations (2023), `https://www.turing.com/kb/different-types-of-cross-validations-in-machine-learning-and-their-explanations`

[12] Zhu, Y., Zimmerman, Z., Senobari, N.S., Yeh, C.C.M., Funning, G., Mueen, A., Brisk, P., Keogh, E.: Matrix profile ii: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins. In: 2016 IEEE 16th International Conference on Data Mining (ICDM). pp. 739–748 (2016)