Data Mining report

Emad Chelhi, Chiara Tomaiuolo

December 2023

1 Introduction to data objects

1.1 Dataset

Data objects were taken from Spotify servers, in a overall number of 20k, everyone with 24 attributes.
The data provided was sorted into two files, 15k in the first one, called a training file; 5k in the other, called test file. A `.txt` file with a brief description of every attributes was provided.
The type and a brief human-readable description of data object attributes are listed in Table 1 [1].

Table 1: Summary table of all the attributes of the raw data objects

Attribute name	Attribute `type`	Brief description
`name`	`object`	name of the track
`duration_ms`	`int64`	track lenght in milliseconds [ms]
`explicit`	`bool`	`True` if there are explicit lyrics, `False` otherwise
`popularity`	`int64`	popularity of the song from 0 to 100
`artists`	`object`	name(s) of the artist(s).
`album_name`	`object`	album name
`danceability`	`float64`	how suitable a track is for dancing from 0% to 100%
`energy`	`float64`	measure of intensity and activity of a song from 0% to 100%
`key`	`int64`	key of the song expressed in Pitch Class Notation (see **Data undestanding**)
`loudness`	`float64`	loudness of a track in decibels [dB]
`mode`	`float64`	mode of the song, minor (0.0) or major (1.0)
`speechiness`	`float64`	Detects the presence of spoken words in a track from 0% to 100%
`acousticness`	`float64`	measure from 0% to 100% of whether the track is acoustic
`instrumentalness`	`float64`	Predicts whether a track contains no vocals, from 0% to 100%
`liveness`	`float64`	Detects the presence of an audience in the recording from 0% to 100%
`valence`	`float64`	Measure from 0% to 100% describing the musical positiveness conveyed by a track
`tempo`	`float64`	tempo in beats per minute [BPM] of the track
`features_duration_ms`	`int64`	Duration of the track in milliseconds [ms]
`time_signature`	`float64`	Estimated time signature of the track in quarter notes
`n_beats`	`float64`	The total number of time intervals of beats throughout the track
`n_bars`	`float64`	The total number of time intervals of the bars throughout the track
`popularity_confidence`	`float64`	The confidence, from 0.0 to 1.0, of the popularity of the song
`processing`	`float64`	-
`genre`	`object`	The genre in which the track belongs

1.2 Data analysis software

Data objects have been analyzed using the `Python` and its libraries, in particular `pandas` for handling data objects with multiple attributes, `sklearn` and `fim` for algorithms routines, `scipy` for statistics quantities and measures, `matplotlib` and `seaborn` for plots and graphics.

2 Data understanding: comprehension and preparation

The first glance to data was not driven by a specific goal, except for data comprehension, data reduction and data quality assess purposes. After the understanding of attributes' characteristics, analysis of the correlations and pmfs(pdfs) of the attributes, *elimination of redundant features* and *filling of* `NaN` *values* in order to prepare data for a task-dependent routine has been performed.

2.1 Data reduction - elimination of redundant features

In order to reduce the dimensionality of the data, the first attributes dropped were `name`, `artists` and `album`, that are basically ID numbers for the dataset. After that, attributes cross relationships have been considered for improving data reduction.

The first quantity that defines whether an attribute contains a considerable amount of information is the correlation between it and the other features: highly correlated attributes contain the same information (because it is possible to obtain one from the another with good precision) and so keeping only one of the two is useful.
In Figure 1 it is shown the correlation matrix containing Pearson correlation coefficients.
In order to assess that there are no considerable differences, also the Spearman correlation matrix has been computed; having no substantial differences, it has not been reported.

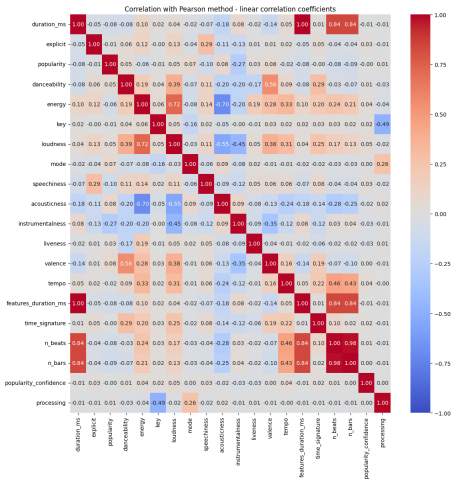


Figure 1: Pearson correlation matrix between all attributes of the raw data objects

2.1.1 Highly correlated parameters elimination

- `features_duration_ms`, is correlated with `duration_ms` with $\rho = 1.0$.
  For assessing that its elimination does not implicate information loss, the mean difference between the two attribute arrays has been computed: $\mu_{\Delta t} = 2 \cdot 10^{-4}$ min (less than a second); the number of data objects that differ for more than 5 s is 4. Those informations confirm this attribute's redundancy.

- `n_beats` and `n_bars`, correlated with `duration_ms` with $\rho = 0.88$ and 0.89 . Considering their definition in musical field, this correlation is reasonable, being `n_beats` and `n_bars` divisions of track time. Because of their definition and this correlation values, they have been considered as redundants.

- `loudness` is correlated to `energy` with $\rho = 0.72$. Being `energy` adimensional as the major part of the attributes in the dataset, it has been kept and the dimensional feature `loudness` has been eliminated because of its redundancy.

2.1.2 The attribute `processing`

Being the `processing` attribute given with no description, it has been inspected deeply in order to understand its meaning. The values of this feature are not human-readable, as shown in Figure 2:

```
In [39]: df['processing'].value_counts()

Out[39]: processing
         4.067086    1800
         3.349057    1741
         2.367412    1568
         3.700483    1481
         1.343558    1431
         1.279305    1209
         0.748116    1205
         1.170953    1089
         0.916010    1081
         0.757389    1063
         2.725904     905
         1.738916     427
         Name: count, dtype: int64
```

Figure 2: `processing` feature values list

Because of the lack of readability, a relationship with the `key` attribute has been seek, having the two a Pearson correlation coefficient $\rho = -0.49$, the highest for the `processing` column (see Figure 1) and being the number of possible values the same as the `key` coefficient: 12. In Figure3 is shown the scatter plot of `processing` vs `key`:

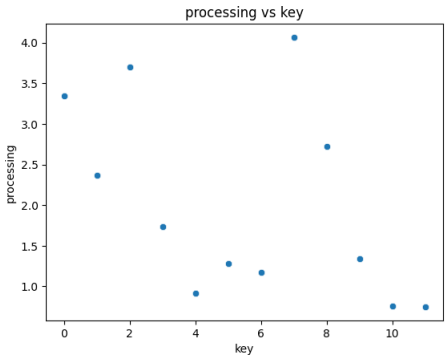


Figure 3: Scatter plot of `processing` as a function of `key`. It is possible to count twelve points, this means that for every `key` value, there is a specific value of `processing`, as further assessed using pandas filtering

Looking at the plot, it is possible to assure that, even if the map is not intuitive, there exists an univocal value of `processing` for every `key`; this classifies the former parameter as redundant and so it has been eliminated.

2.1.3 The attribute `key`

The `key` attribute is expressed as Pitch Class Notation, the standard music field.
Being the musical scale circular, as suggested by Figure 4, for this attribute two kind of distances have been considered:

- a **custom distance** that considers the distance between different values in the circle, not the raw difference between attributes value (in absolute value); that classifies the attribute as *categorical ordinal*;
- a **standard distance**, that considers the raw difference between attributes using Jaccard distance; this classifies the attribute as *categorical not ordinal*.

It has to be noted that for all the other categorical attributes Jaccard distance has been used in both custom and standard distance.

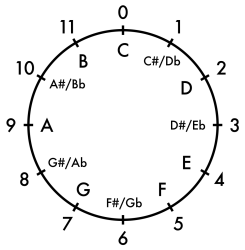


Figure 4: Pitch Class Notation depiction

2.2 Transformation of data attributes

In order to enhance human readability and standardization of some features, some rescaling has been performed.

`duration_ms` to `duration_min`

Being minutes the most human readable unit of measure for track lenght instead of ms, `duration_ms` was rescaled in order to change its unit of measure, in this way difference between tracks became more readable, in fact a difference of thousands of milliseconds is not important in the framework of listening, instead a difference in minutes is considerable.
The rescale factor is: $c_{\text{ms}\to\text{min}} = 6 \cdot 10^{-4}$.

`popularity` to `popularity_percent`

Being all the other attributes expressed in percentages, `popularity` has been rescaled to a decimal value as well.
The rescale factor is: $c_{\%} = \frac{1}{100}$

2.3 Filling of missing values

Using the `pandas` function `df.isnull().sum()`, we have found that 3 features of the dataset contained at least one `NaN` value:

- `mode` attribute contained 4450 missing values;
- `time_signature` attribute contained 2062 missing values;
- `popularity_confidence` attribute contained 12783 missing values;

The number of not-`NaN` values of `popularity_confidence` are not enough for obtaining reasonable statistics of the attribute. For this reason, we have decided to eliminate the attribute from the analysis.

2.3.1 Filling criterion

Our criterion of replacing was to maintain as invariant as possible the overall pmf of the attributes to fill; in order to do that, replacing values has been sampled from the distribution of the two attributes separately. Being both attributes discrete, the probability of every outcome wrt the total has been computed.

- `mode`: having only two possible outcomes, p_0 and p_1 has been computed, resulting in $p_1 > p_0$ with $p_0 = 0.37$ and $p_1 = 0.63$. For assuring that the gap between the two values wasn't due a problem of reading of value 0, an online research has been done, finding an article that affirms the human preference of major mode in musical tracks, (see [3] for example).
  A random number $\xi \in [0, 1)$ has been sampled, if $\xi > p_1$, then the replacing value was 0.0 (minor), else 1.0 (major).
- `time_signature`: having multiple outcomes, after having computed p_i with $i \in \{0, 1, 2, 3, 4, 5\}$, the replacing values has been sampled from a multinomial pmf (using `scipy.stats.multinomial`).

After sampling, a cross check has been done, recomputing the probabilities of all outcomes for both attributes, confirming that they remained the same.

3 Clustering

The first task of the project was to try several types of clustering methods on the dataset, in order to search for groups in an unsupervised way.
Having no supervision, most of the times is not possible to do an external verification of the results, so the clusters has been evaluated with proper internal measures. For a single case, an external verification has been tried: cluster the data objects by genra using, as subgroup of data, all the technical musical features.

3.1 Clustering data preparation - choice of data subsets

Being the K-means algorithm based on Euclidean distance, only numeric attributes could have been used, where this definition of distance makes really sense. A first data subgroup (I) has then picked with only continuous attributes. As second data subgroup (II), for which two kind of distances has been defined where the difference is given by the computation for `key` feature as previous described, the attributes listed in Table 2 has been picked:

Table 2: Data subgroups - (in II `valence` attribute has been forgotten by mistake)

I data subgroup	II data subgroup	attribute type
`duration_min`	`duration_min`	continuous
`popularity_percent`	-	continuous
`danceability`	`danceability`	continuous
`energy`	`energy`	continuous
`speechiness`	`speechiness`	continuous
`acousticness`	`acousticness`	continuous
`instrumentalness`	`instrumentalness`	continuous
`liveness`	`liveness`	continuous
`tempo`	`tempo`	continuous
`valence`	-	continuous
-	`mode`	categorical not ordered
-	`time_signature`	categorical not ordered
-	`key`	categorical (order depends on distance chosen)

3.1.1 Outlier selection criterion

The outlier criterion for the following clustering procedures consists in the following filter:

```
time_signature > 1.0 or genre == sleep
```

From data understanding, we have discovered that tracks having `time_signature` ≤ 1.0 are mainly those containing noise sounds and natural sounds indeed almost of them belong to the genre `sleep`. This cut has been chosen in order to try to select only *music tracks*, where the technical features chosen are all well-defined; actually, all those features have values even in non-music tracks but are subject to detection errors. For example, it does not really makes sense to measure the acousticness of tracks containing forest sounds: the algorithm could assign to it a number different from zero for any reason and this would be incorrect.

3.1.2 Normalization

Before applying the clustering routines, attribute values has been normalized in order to make them weight the same.
The normalization chosen is the *minmax normalization*, that rescaled the interval of all features to $[0, 1]$. This normalization has been chosen because it left the majority of feature values unchanged, being them already percentual values and so in the right range; this guaranteed a good human-readibility in phase of development of most of the attributes.

3.1.3 Internal measures for clustering tuning and validation

Three internal measures for clustering validation has been considered:

- SSE for tuning of $K-means$ clustering, because the algorithm is based on its minimization and so was useful to see its trend also for clustering validation;
- Sil coefficient, that is the *Silhouette coefficient*. This measure is in general more interesting than SSE, because it considers both the *cohesion* and *separation* of clusters between each other;
- $CVNN$ index as described in [4]. This internal measure would require some tuning, because it takes as input parameter the number of nearest neighbour to be considered for the computation of Separation and Cohesion. For computational time issues, a standard case using as number of neares neighbours $NN = 10$ has been considered, the implementation provided by the repository linked in the references ([5], see `internal_validation.py`) has been used. This measure has been computed for only few cases for cross checks with other measures.

3.2 Centroid-based clustering

The first algorithm taken into consideration for clustering is the *K-means* algorithm. This method required the tuning of its unique parameter K, the number of clusters (that is, the number of centroids) and uses the Euclidean measure, so only the I data subset has been clustered with this method.
In `sklearn` routine, it is possible to choose the method of initial centroid computation, the standard one follows the *K-means++* routine [2, Chapter 5, p.326], that consists in the following steps:

1. the first centroid is picked randomly;
2. the following ones are picked from the dataset points, every dataset point has a probability $p \propto d^2$, where $d =$ distance from the closest centroid.

The K-means++ routine is robust with respect to outliers, because the probability of picking an outlier as centroid is strongly suppressed, being them ideally unlikely.

3.2.1 K parameter choice - I data subset

In order to tune the number of clusters K for the algorithm, the routine has been executed for every K in $[2, 200]$ and SSE, Sil and $CVNN$ with $NN = 10$ computed. In the ideal case, the best K is the lowest K where the elbow of SSE, a local maximum for Sil and a local minimum of $CVNN$ index appears.
In Figure 5, 6 the result of the procedure. Based on the exploited criteria, $K = 5$ has been chosen as best parameter.

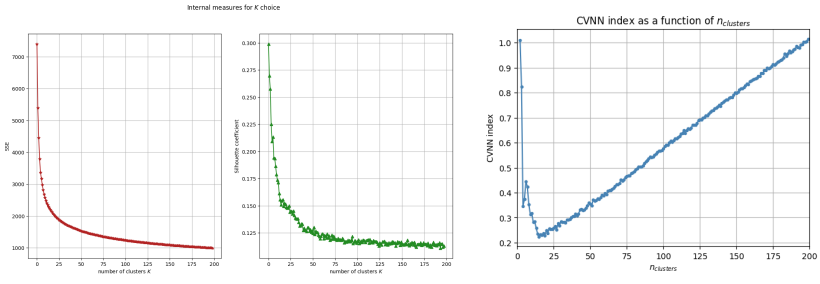
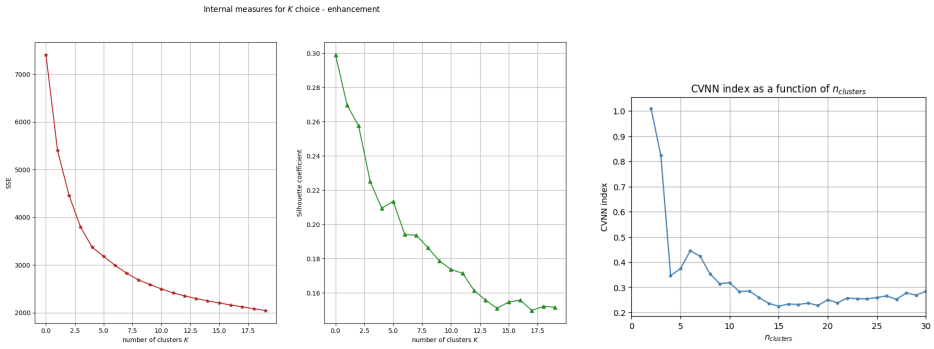


Figure 5: Complete trend of internal measures SSE, Sil and $CVNN$ used for tuning the K parameter.



(a) Enhanced trend

Figure 6: Enhanced trend of the internal measures SSE, Sil and $CVNN$ used for tuning the K parameter. The values of the metrics for the chosen $K = 5$ are: $SSE = 3376$, $Sil = 0.217$, $CVNN = 0.37$

3.2.2 External measure attempt - genre attribute vs K-means results - I data subset

Being the starting hypothesis that the tracks could be regrouped considering `genre` labels using the musical attributes, a clustering setting $K = 19$, that is the number of genra in the data subgroup, has been performed. In Figure 7 the comparison between `genre` and the clustering results, reassumed in the column `kmeans_labels`.

It is evident that this attempt has not given the expected result: in every cluster appears almost

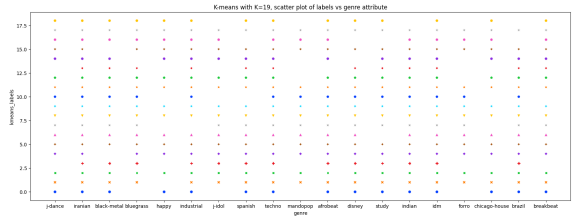


Figure 7: Trial of K-means clustering with $K = 19$, scatter plot of `kmeans_label` as a function of `genre`

every genre of music, concluding that K-means clustering procedure could not classify the dataset by `genre`.

3.2.3 K-means clustering results with $K = 5$ - I data subset

After having fixed the number of clusters to 5, the resulting groups has been studied: in particular, correlations between `kmeans_labels` and the other attributes has been computed, those quantities could be interpretated as a sort of weight in the choice of cluster assignement. In Figure 8 the Spearman correlation matrix, that showed stronger correlations coefficients than the Pearson one, pointing out non-linear trends.

For the sake of clustering understanding, the distance between centroids in divided by attribute is shown in Figure 10. Being the attributes `duration_min` and `tempo` out of a percentual scale, they have been cut from the first plot. The attribute `duration_min` has been plot separately. The latter has not been reported because not relevant: the distance between centroids, in terms of it, was not considerable.

Figure 10 is interesting because advices which could be the best parameters for localizing a specific cluster. In particular, some centroids (and so, hopefully, clusters) are divided surely better by `acousticness` with respect to `speechiness`. In particular the figure regarding `duration_min` advices that 3 clusters recollect tracks of mean lenght; long tracks (> 4 min) are recollected in other clusters.

By looking at the plot, two good attributes for dividing properly the clusters are `acousticness` and `valence`, whereas for example `liveness` and `speechiness` aren't:

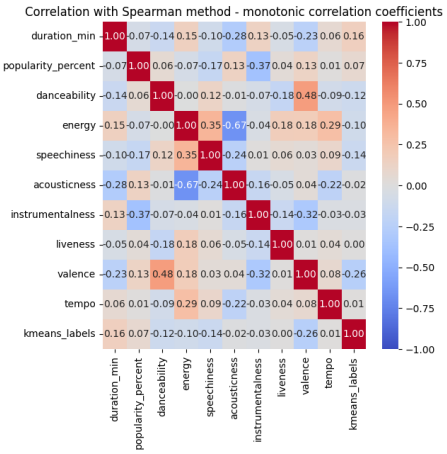
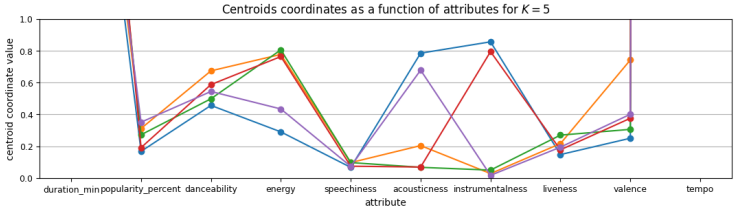
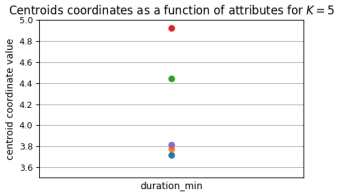


Figure 8: Spearman correlation matrix between attributes. The interesting column is the last one: the correlations between `kmeans_label` and the other attributes

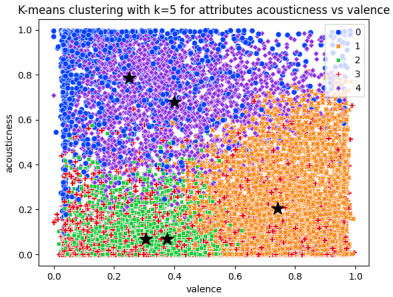


(a) Distance between centroids for percent coordinates

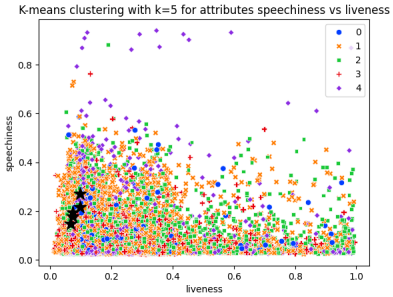


(b) Distance between centroids for track lenght

Figure 9: Centroid coordinates divided by attribute



(a) Good visual division



(b) Bad visual division

Figure 10: Visual rappresentation of clusters in two possible 2D planes.

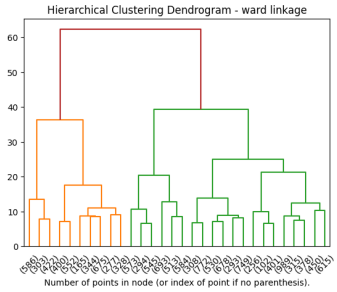
3.2.4 Cluster tendency

In order to assure that the clustering performed by the routine was not random, a consistent random data were created and clustered $N = 1000$ times, the SSE of this clusters computed and the pdf of the measure constructed. The value of SSE of non-random points cluster has been compared to the distribution, showing that the results are significant and not random, because the SSE is an extreme outlier with respect to the distribution of random measures: $SSE_{\text{effective}} = 3376$ and $SSE_{\text{random clusters}} = 9541 \pm 24$.

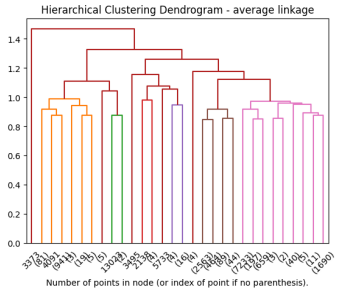
3.3 Hierarchical clustering - I data subset

Without changing data subsets, it has been done also a hierarchical clustering, considering for I the four standard methods of linkage provided by the `sklearn` package [6], that consists in: *maximum linkage, minimum linkage, Ward's linkage, group average linkage*. In Figure 11 the dendrograms relative to those procedures.

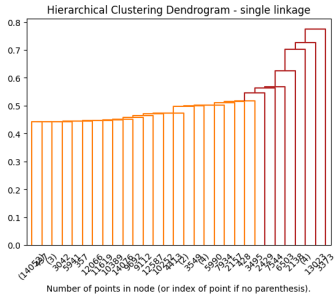
It is possible at first glance to detect several properties of the linkage methods:



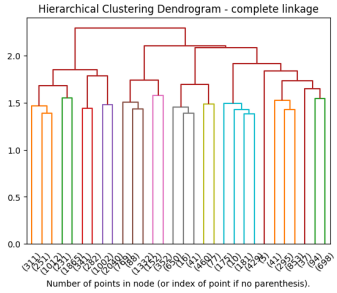
(a) Ward's linkage method dendrogram



(b) Group average linkage method dendrogram



(c) Single linkage method dendrogram



(d) Complete linkage method dendrogram

Figure 11: Dendrograms depicting the various methods of hierarchical clustering linkage methods

- **complete linkage** tends to form more clusters and in general to divide big clusters, in fact, it creates more clusters than the other methods;
- in contrast to complete linkage, **single linkage** actually creates a single cluster regrouping all points (see first point in x-axis, the cluster contains 14053 points out of 14090), leaving only few points outside first regroupment. This method will be analysed for the sake of understanding it properly but then discarded because not informative;
- **Ward's method** provides a different type of computation, because is a minimization method, calling recursively the distance between points, that's why its y-axis has a different scale [7].

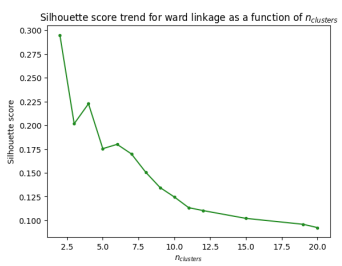
3.3.1 Number of clusters $n_{clusters}$

Being the best number of clusters $n_{clusters}$ in a hierarchical clustering a decision of the user, we have considered as in K-means technique an internal measure for the evaluation. In I, we have considered the Sil coefficent and the $CVNN$ index with $NN = 10$.

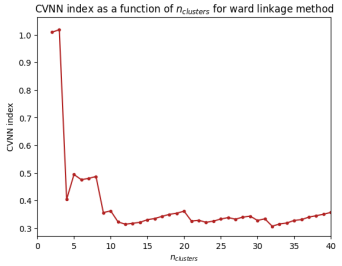
As shown in Figure 12, single linkage approaches negative values for Sil as the number of clusters increases, representing lack of separation. Being the value of Sil higher for Ward and average linkage, those seemed to us the two best methods of clustering for the data subset.
$CVNN$ index is not stable for average and single measure, in fact, it would be interesting to exploit this internal measure as a function of its parameter NN in order to seek for the minimum value of the index as suggested in [4] but this analysis has not been done for time issues, so $CVNN$ index value was reported only for a cross check with Sil score value when its trend was considered stable. In Table 3 the best values for the parameters and its associate $n_{clusters}$ are summarised and the routine with the best characteristics highlited.

Table 3: Summary table of best values of internal measures for I in hierarchical clustering routines. In red the number of cluster and its relative measure(s) that we assumed the best of the set

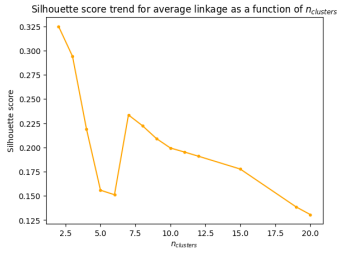
linkage	$n_{clusters}$ (Sil / $CVNN$)	Sil	$CVNN$ ($NN = 10$)
Ward	4/4	0.225	0.4
avg	6/6	0.15	0.11
complete	10/7	0.11	0.45
single	7/-	0.07	-



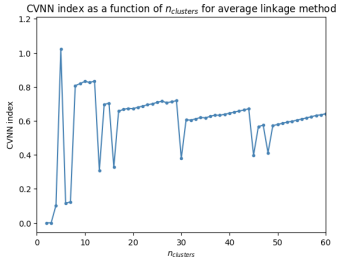
(a) Ward's linkage method silhouette coefficient trend



(b) Ward's linkage method CVNN index trend

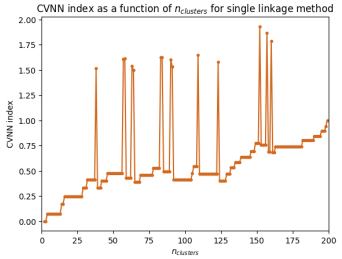


(c) Group average linkage method silhouette coefficient trend

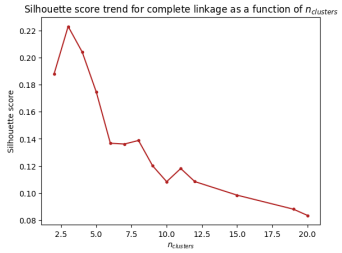


(d) Group average linkage method silhouette CVNN index trend

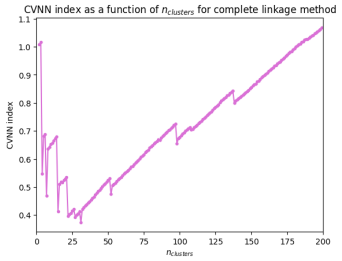
(e) Single linkage method silhouette coefficient trend



(f) Single linkage method silhouette CVNN index trend



(g) Complete linkage method silhouette coefficient trend



(h) Complete linkage method silhouette CVNN index trend

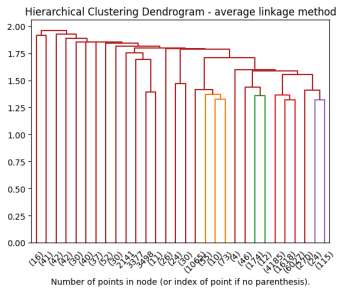
Figure 12: Silhouette coefficient and CVNN index trends for the exploited linkage methods

In particular for average linkage method, the *Sil* coefficient seems more stable and so weighted more in our decision, anyway, in both the methods selected, the two measures give the same optimal number of clusters.

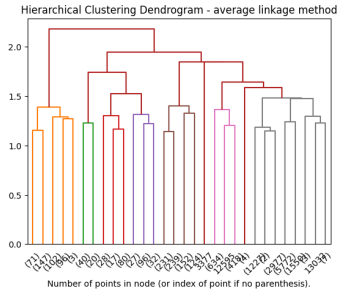
3.4 Hierarchical clustering - II data subset

The previous analysis has been repeated for II, using both the distances introduced in 2.1.3. We have reported in Figure 13 the dendrograms relative to the procedures and the summary of the internal measures in Table 4. It has to be noticed that Ward's method cannot be used in this cases because uses the Euclidean measure.

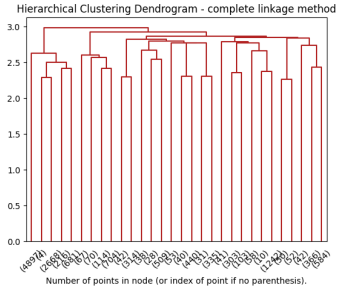
It is interesting to see that, as expected, the two different distances gave different dendrograms, even if the overall scale of distance differs only few decimals.



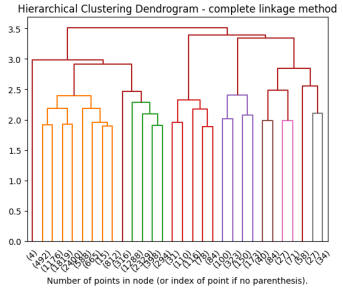
(a) Standard distance with group average linkage method



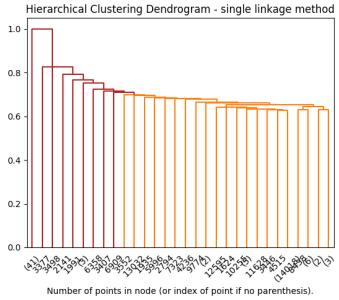
(b) Custom distance with group average linkage method



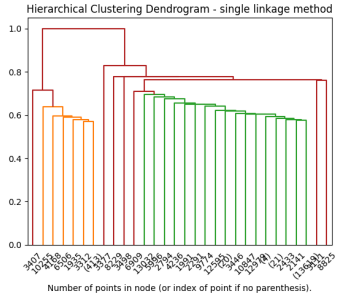
(c) Standard distance with complete linkage method



(d) Custom distance with complete linkage method



(e) Standard distance with single linkage method



(f) Custom distance with single linkage method

Figure 13: Dendrograms for the various kind of distances and linkage methods

- Even if less pathological than the first data subset, **single linkage** method still formed a single macro cluster and left only few single points at last stages of regrouping;
- In **complete linkage**, clusters appear more distinct with the usage of the custom distance, same in **average linkage**. It can be seen in Table 3 that custom measure tends to create less clusters and with a greater (and so ideally better) *Sil*.

Table 4: Summary table of best values of *Sil* for II in hierarchical clustering routines. In red the number of clusters and its relative measure that we assumed the best of the set.

linkage	distance	$n_{clusters}$	*Sil*
single	std	1	no max
single	custom	75	0.05
complete	std	17	0.23
complete	custom	7	0.3
average	std	16	0.19
average	custom	12	0.26

3.4.1 Cluster tendency

For the three combinations highlighted in Table 3 and 4, the same test, as in K-means case for assuring the un-randomness of resulting clusters, were computed. This time, instead of SSE pdf, the Sil pdf was constructed with $N = 200$ for I and $N = 50$ for II.

dataset, distance, linkage	Sil_{signal}	$\mu_{random} \pm \sigma_{random}$
I, euclidean, ward	0.225	0.031 ± 0.002
II, custom dist, average	0.26	0.027 ± 0.004
II, custom dist, complete	0.3	0.029 ± 0.003

Table 5: Mean and standard deviation of Sil score for randomized data compared to Sil score resulted from clustering on effecting data.

Being the Sil coming from the real data and so having several standard deviations away from the centre of the distribution of the Sil given by random data, our clusters can be considered as valid and not random.

3.5 Density-based clustering: DBSCAN

The DBSCAN algorithm needs the tuning of two parameters, they have been determined for every trial of density-based clustering:

- Eps: the radius of the area for the determination of core, border and noise points;
- $MinPts$: minimum number of points within the area of radius Eps in order to label the points as core, border or noise.

3.5.1 Tuning of Eps and $MinPts$

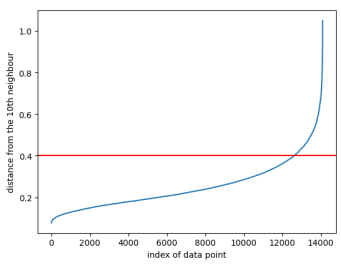
To determine the parameter Eps, the heuristic method has been used based on the trend of the k-th nearest neighbour distance for each single point; with $k \geq 1$ and for datasets of 2 dimensions it is suggested in literature to set $k = 4$. Considered the higher dimensionality of the datasets, we decided to try setting $k = 10$. According to [8, p.181–183], the choice of the k is not very crucial for the algorithm. After having compared the results with the two different k, the ones with $k = 10$ have been kept, indeed as expected the results did not differ very much and an higher k provided a lower computational time.

It has been computed the k-distance for all the data points, sorted them in increasing order and plotted them in function of the points. The trend had an elbow shape and the optimal value of Eps was given by the point of greatest slope. The elbow shape is interpretable as a threshold: all points with a higher k-distance value to the right of the threshold will be labeled as noise points, all the others will be assigned to some cluster.
Having chosen the Eps value, it was time to determine the optimal $MinPts$. In general, it is good practice to compute an internal measure for different values of $MinPts$ and observe its trend. Considering the computational cost, it is desirable the lowest reasonable value of $MinPts$ possible, where $MinPts = 1$ is not considered in order to avoid single-link effect. We exploited values from 2 to 51 of $MinPts$ considering the Sil score. In Figure 14 the described plots for the II data subset analysed with the custom distance are shown.

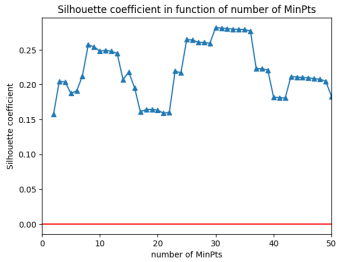
3.5.2 DBSCAN routines

The algorithm has been tuned for both data subsets I and II previously described. For II it has been repeated the routine for both custom and standard distances for feature `key` in order to see the differences, if any. Therefore, three different density-based clustering analysis have been done. The internal measures for assuring clustering quality were the following:

- *Sil coefficient*;
- the *number of noise points*, ideally to be kept as low as possible (we assume to have no outliers, every point should be labeled);



(a) k-th nearest neighbor distance for II data subset computed using custom distance



(b) Silhouette score as a function of $MinPts$. Eps has been fixed by previous procedures. The plot is referred to the II data subset, using custom distance

Figure 14: Plots used for the choice of the density-based clustering procedure Eps and $MinPts$.

- the *number of clusters and their size*. For certain values of $MinPts$ (higher ones), most of data objects were all labeled in the same macrocluster, giving no group formation.

In Table 6, are reassumed the quantities introduced above for all the three considered routines.

Data subset	Eps_{opt}	$MinPts_{opt}$	$n_{clusters}$	$n_{elements}$	n_{noise}	Sil
I	0.35	5	1	1: 13892	196	0.18
II (std dist)	0.4	20	5	1: 11743, others < 50	1976	-0.002
II (custom dist)	0.44	8	8	1: 7952, 2: 4611, 3: 558, 4: 334, 5: 138, others < 100	320	0.26

Table 6: Summary table of results with DBSCAN routines, reporting the optimized parameters, the number of clusters (cluster with noise excluded) obtained and their size, the number of noise points and the corresponding Silhouette score. In red the clustering procedure with the best results

As shown in Table 6, The only combination that did not return a single macrocluster, that gives us no information about data objects, is the third one: II data subset with usage of the custom distance, even if the number of relevant cluster is in fact less than 8, because three of them contain only few points and having the 85% of data contained in the first two clusters; Sil score anyway is the best one obtained. The following analysis made us assuring that the dataset is not really suitable for this kind of clustering; that's why we didn't validate cluster tendency for this kind of algorithm.

3.6 Conclusions

Considering as parameters for inferring the goodness of a clustering its silhouette coefficient and also the size of the resulting clusters (that means, macroclusters and clusters with few points are not interesting), the more interesting ways for clustering the dataset taken into consideration are the K-means method and the hierarchical method with complete measure; the latter in particular is more interesting when the custom distance is used and in general because contains more clusters and uses more attributes, so it has been considered the best one.

4 Classification

In the following *supervised task*, the goal is the classification of tracks using as labels the `genre` attribute.
The whole dataset labeled as `train.csv` has been used as training set and the whole dataset labeled as `test.csv` has been used as test set.
The algorithm for tree construction is provided by `sklearn`.

4.1 `genre` maps for classification

Two maps for the target label `genre` has been built:

- **A label for every genre**: in this case, the number of classes is 20. This labeling will be called *complete map*.
- **A label for every macrogenre**: in order to reduce the number of classes, genra has been encorporated by macrogenre groups. In this case, the first regroupement has been done considering a list found on the internet [9], and improved after the first tree classification performed below. In this case the number of classes is reduced to 7. This labeling will be called *macrogenre map*.

For completeness, in Table 7 and 8 the complete and macrogenre maps labels are shown.

Table 7: Complete map labels

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

Label	genre list
0	breakbeat, chicago house, techno, afrobeat, idm , industrial
1	mandopop,j-idol,happy,disney,j-dance, indian
2	forro,brazil,spanish
3	bluegrass
4	black metal
5	iranian
6	sleep, study

Table 8: Labels assigned in macrogenre map

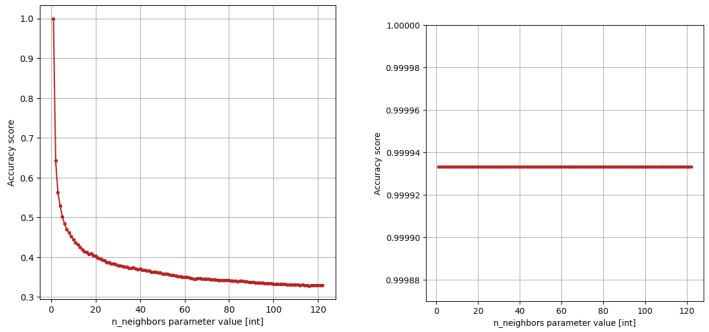
4.2 Data preparation for classification

Data preparation is equivalent to the one done for clustering, except for outlier choice. Here, no outlier criterion has been used, because also non-musical tracks would be interesting to classify.

4.3 KNN classificators

The KNN classification has been performed for the *complete map*.
In order to perform a KNN classification, the following parameters have to be tuned: *distance metric*, *number of nearest neighbours*, *weights*. Being interesting the use of both categorical and continuous attributes, the *custom distance* has been used.
`sklearn` KNN classificator takes as input a precomputed distance matrix that has been computed separately. The main issue has been the validation, the standard cross validation does not perform cleverly with precomputed matrix: changing data objects in position inside the dataset does not imply the change in distance matrix elements position; this results in associating the wrong computed distance between points; because of that, the only validation performed in train set has been a 3-times repeated holdout, recoputing the matrix everytime.
Being impossible the use of `GridSearchCV()`, a manual tuning has been performed, varying the parameters `n_neighbours` and `weights`.
When setting `n_neighbours = 1`, the classificator has shown barely perfect behaviour with an accuracy close to one for a single classificator accuracy = 0.99997 and also for the mean of the repeated holdout accuracy.

The `weights` parameter introduces a stability wrt the `n_neighbours` parameter, fixing the accuracy to its maximum value. Being in this case the value of the nearest neighbour completely



(a) Case with `weights=uniform`. (b) Case with `weights=distance`.

Figure 15: Tuning of a KNN classifier parameter `n_neighbors` for the two standard values of `weights`: `uniform` and `distance`. It is evident the stability of accuracy when `weights = distance`.

ininfluent, it has been fixed to 1 as in the previous case.
The accuracies for the two described cases are shown in Figure 15:

This analysis suggests that every data object has the nearest neighbour of the same class (that probably has almost all features equal and so 0 distance, probably the only exceptions would be dictated by continuous attributes; this behaviour explains the stability seen when setting `weight = 'distance'`).

Perfect performances have been obtained also for test set, because of that, ROC curves are not shown, because of their trivial shape.

It is possible to conclude that for this dataset, the attributes used are exhaustive for the task of `genre` classification using a KNN classificator.

4.4 Decision tree classificators

Two decision trees have been built and tuned, one for *complete map* and another for *macrogenre map*.
For evaluating decision trees performances we have used the standard metrics provided by `sklearn`: accuracy and F1-score. For final evaluation, the whole confusion matrix has been plotted and commented.
In every tree the procedure used is the following:

1. preparation of both training and test datasets;
2. definition of the tree and training using the whole training set;
3. first evaluation: untuned tree performances using prediction on training set, both without and with cross validation;
4. use of `RandomizedSearchCV()` [10] for doing a random search in order to tune the tree's structure parameters: `criterion`, `max_depth`, `min_samples_split`, `min_samples_leaf`;
5. second evaluation: tuned tree performances using prediction on training set, both without and with cross validation;
6. first tuning of cost complexity coefficient on the tuned tree;
7. using pruned tree for predictions on test set;
8. final pruning: choice of cost complexity coefficient comparing accuracy on training and test set;
9. final performance evaluation of the pruned tree on test set and evaluation of the confusion matrix for completeness.

4.4.1 Decision tree - complete map

In Table 9 the size measures of the tree before tuning are shown.

max_depth	n_leaves	node_count
32	5140	10279

Table 9: Size measures of the untuned tree for the complete map.

The performances and importances of attributes of the untuned tree show strange behaviour: the precision and F1 scores are "perfect", always equal to 1. The attribute importance is never 0 for any attribute, and always lower than 15%.

Untuned tree The "perfect" performances exhibit a strong overfitting of the tree: it adapts perfectly to training data, as can be also seen by the number of final leaves `n_leaves`: they are around $\frac{1}{3}$ of the total number of data objects, that means that on average every leaf contains only three tracks.
The overfitting theory is again confirmed by evaluating the classification performance with a cross validation that splits the training set into an effective training set and a validation set, used for testing. It has been obtained computing the accuracy with a cross validation with 5 folds, $\mu_{\text{accuracy}} = 0.43$, tremendously lower than 1. As last evidence of overfitting, all features are used for tree classification and all have little weights.

Tuned tree results The results of the `RandomizedSearchCV()` are the following:

```
min_samples_split: 20, 'min_samples_leaf': 5, 'max_depth': 11, 'criterion': 'gini'
                                accuracy = 0.48
```

Looking at the accuracy value it is clear that the tuning is a good start for avoiding the overfitting issue. In Table 10 size measures and performance metrics of the tree are shown. Performances exhibit great improvement regarding the overfitting issue. Feature importances of attributes are still pathological but a single attribute weight, that is `time_signature`'s one, has been set to 0.
It has to be noted that in Table 10 the accuracy is higher than the one in `RandomizedSearchCV()` because the cross validation has not been done.

max_depth	n_leaves	node_count
11	572	1143

	precision	recall	f1-score	support
0	0.57	0.60	0.59	750
1	0.74	0.77	0.75	750
2	0.47	0.46	0.47	750
3	0.71	0.66	0.68	750
4	0.65	0.68	0.67	750
5	0.56	0.50	0.53	750
6	0.47	0.50	0.48	750
7	0.80	0.86	0.83	750
8	0.71	0.57	0.63	750
9	0.44	0.40	0.42	750
10	0.49	0.57	0.53	750
11	0.66	0.52	0.58	750
12	0.88	0.85	0.86	750
13	0.43	0.44	0.44	750
14	0.73	0.54	0.62	750
15	0.43	0.51	0.46	750
16	0.53	0.62	0.57	750
17	0.49	0.64	0.56	750
18	0.82	0.79	0.80	750
19	0.53	0.47	0.50	750
accuracy			0.60	15000
macro avg	0.60	0.60	0.60	15000
weighted avg	0.60	0.60	0.60	15000

Table 10: Size, performance metrics of the classification tree after training on training set and after tuning.

Pruning After tuning, the tree has been used for prediction on the test set and, before evaluating the final performances, the `ccp_alpha` parameter, that is the *cost-complexity parameter*, has been chosen in order to prune the tree properly. The tree has been pruned trying to choose the `ccp_alpha` that maximized the test accuracy *and* minimized the gap between test accuracy and train accuracy. In Figure 16 is shown the plot used for the evaluation. The chosen value is `ccp_alpha = 0.0025` .

Final results In Table 11 and Figure 17 the final results are shown:
The effect of pruning is evident, the size of the tree is dramatically smaller and some attribute's importance has drop to 0 weight but this is payed in accuracy score on train set, that was still overfitting after tuning. It is possible to see that some labels are easier to classify than others:

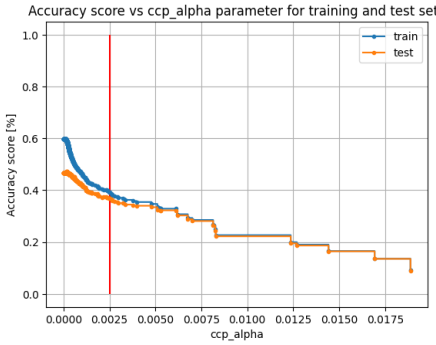


Figure 16: Accuracy score for both training and test datasets for the choice of the `ccp_alpha` parameter. In red, a line that shows the chosen parameter value.

max_depth	n_leaves	node_count
7	29	57

```
Train Accuracy 0.3926
Train F1-score [0.50139276 0.56664006 0.          0.48879837 0.46461372 0.28521435
 0.         0.67001886 0.32952253 0.          0.34331984 0.4024864
 0.74793388 0.21844226 0.54914197 0.          0.38751346 0.38449491
 0.59254948 0.27315542]

Test Accuracy 0.369
Test F1-score [0.48625793 0.51941748 0.          0.5115304  0.45806452 0.24146982
 0.         0.66536965 0.29166667 0.          0.3280543  0.39719626
 0.70833333 0.18571429 0.46226415 0.          0.38184664 0.37440758
 0.5380531  0.22680412]
```

```
popularity_percent 0.2810686824933053
danceability 0.17412841449934147
duration_min 0.13705822566882386
loudness 0.11196405892520342
instrumentalness 0.09368484274591707
acousticness 0.09018049809505584
speechiness 0.04111389861802162
valence 0.030697238178044636
energy 0.027220149463695748
tempo 0.012883991312591167
explicit 0.0
key 0.0
mode 0.0
liveness 0.0
time_signature 0.0
```

Table 11: Size, performance metrics and feature importance results of the classification tree after training on training set and after tuning.

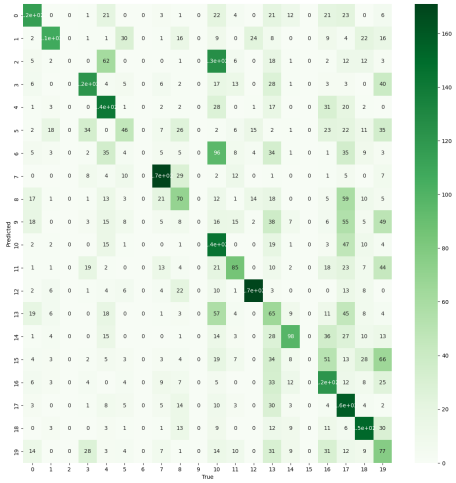


Figure 17: Confusion matrix for the pruned tree on the test set. It contains the complete information about the classificator. It has to be remembered that in test set, there are 250 data objects for every `genre`.

- `j-dance`, `chicago-house`, `iranian` reach around 50% of accuracy;
- `spanish` has a 19% accuracy;
- `brazil`, `indian`, `afrobeat`, `industrial` are *never* identified.

The zero accuracy on some genra suggests to look at the confusion matrix in order to search for the genra for which the misclassified values are often confused. This could be done in order to find macrogroups of genra; this helped in finding the second map used for classifying genra with classification tree.

4.4.2 Decision tree - macrogenre map

The genra has been regrouped following the list in [9] and in case of ambiguity or absence of the specific genre in the list, the genre has been assigned to the macrogenre of the one for which has been more often confused by the previous tree. Table 8 shows the labels of the map.

Untuned tree The untuned tree presented the same problems as first case: complete ovefitting, fake 'perfect' behaviour with maximum accuracy and F1 score and usage of all features for classification. In order to compare this stage to the following ones, in Table 12 the size measures of the untuned tree are shown:

max_depth	n_leaves	node_count
26	3333	6665

Table 12: Size measured of the untuned tree for the macrogenre map

Tuned tree results The results of the `RandomizedSearchCV()` are the following:

`'min_samples_split': 50, 'min_samples_leaf': 10, 'max_depth': 18, 'criterion': 'entropy'`
`accuracy = 0.64`

In Figure 18 all the interesting metrics are shown. It can be seen that, even if the overfitting issue is less pathological, its resolution is still a matter of uncertainty: all features are still, most of them, with few weight, used.

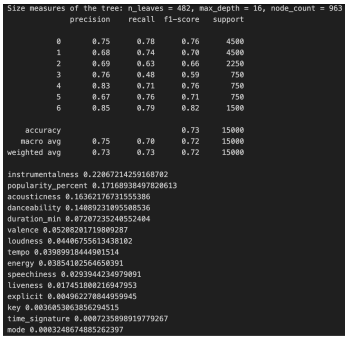
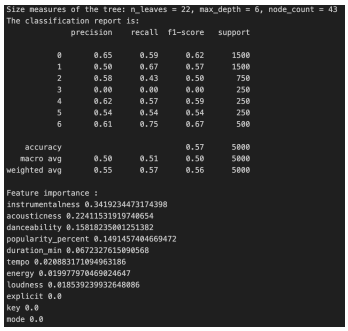
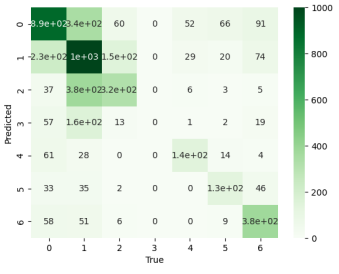


Figure 18: Size metrics and performances of the tuned tree for the macrogenre map

Pruning and final results After prediction on test set and pruning parameter choice, that has set `ccp_alpha = 0.01`, the final performance evaluation and confusion matrix are shown in Figure 19:



(a) Size, performance evaluation and feature importances (the features not in the image have 0 weight.)



(b) Confusion matrix for the pruned tree

Figure 19: Final metrics of evaluation of the pruned tree for *macrogenre map*.

4.4.3 Final comment

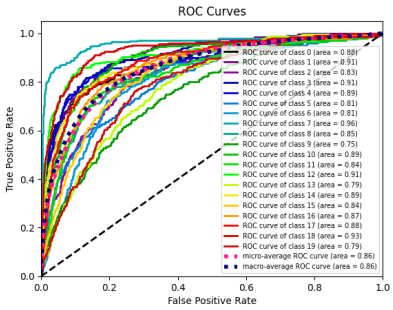
As expected, having the *macrogenre map* less categories, performances are better than *complete map*. In this latter case, performances are good except for `genre = bluegrass`, that is never predicted and so decreases the overall scores. Tree could be improved encorporating this genre to another macrogroup but hardly performances of KNN classificator would be accomplished.

4.5 Naïve Bayes Classificator

Lastly, we performed the genre classification through the Naïve Bayes classificator.
We performed 2 different analysis: one using the *complete map* and one for the *macrogenre map*. For the preparation of the training and test set we removed all the redundant, because highly correlated, features that could degrade the performances because of the Bayes assumption of conditional independence; and all the categorical ones. Then we had to manage the different estimation procedure of the conditional probabilities between discrete and continuous attributes. After deciding to use both, we solved this issue recalling the Naïve Bayes assumption, above mentioned, that states that *"the joint conditional probability of X1 and X2 given Y can be factored as the product of conditional probabilities of X1 and X2 considered separately"* [2]. So, we simply calculated the continuous features with a Gaussian NB model, the discrete ones with a Multinomial NB model and then we multiplied them. After the normalization step we computed different values for the threshold parameter of the classification but the default one, equal to 0.5, gave us the best results. In Figure 20, 21 the test set prediction performances are shown:

	precision	recall	f1-score	support
0	0.55	0.50	0.52	250
1	0.53	0.42	0.47	250
2	0.52	0.18	0.27	250
3	0.28	0.76	0.41	250
4	0.37	0.64	0.47	250
5	0.38	0.11	0.17	250
6	0.21	0.04	0.07	250
7	0.77	0.77	0.77	250
8	0.32	0.28	0.30	250
9	0.48	0.04	0.08	250
10	0.28	0.68	0.40	250
11	0.31	0.10	0.15	250
12	0.54	0.66	0.59	250
13	0.20	0.15	0.17	250
14	0.26	0.74	0.38	250
15	0.32	0.12	0.17	250
16	0.36	0.38	0.37	250
17	0.41	0.21	0.28	250
18	0.62	0.53	0.57	250
19	0.19	0.15	0.17	250
accuracy			0.37	5000
macro avg	0.40	0.37	0.34	5000
weighted avg	0.40	0.37	0.34	5000

(a) Classification report for Bayes classificator on *complete map*

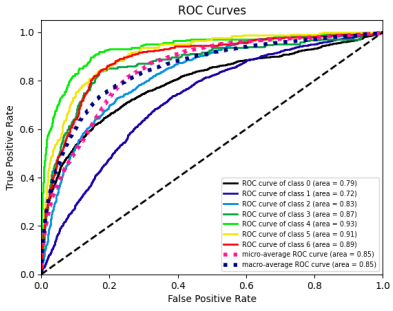


(b) ROC curves for Bayes classificator on *complete map*

Figure 20: Performances evaluation on test set for Bayes classificator on *complete map*.

	precision	recall	f1-score	support
0	0.63	0.60	0.61	1500
1	0.57	0.18	0.27	1500
2	0.33	0.77	0.47	750
3	0.33	0.48	0.39	250
4	0.55	0.60	0.57	250
5	0.43	0.50	0.46	250
6	0.52	0.49	0.51	500
accuracy			0.48	5000
macro avg	0.48	0.52	0.47	5000
weighted avg	0.53	0.48	0.46	5000

(a) Classification report for Bayes classificator on *macrogenre map*



(b) ROC curves for Bayes classificator on *macrogenre map*

Figure 21: Performances evaluation on test set for Bayes classificator on *macrogenre map*.

Despite the good results of the area under the curve (AUC) resulting from the plot of the ROC curves, mean classification metrics, overall, are not good as expected because some genra aren't well predicted, i.e. `genre = afrobeat` has recall and f1-score under 10%; instead the majority of classes have metric values over 50%. This discrepancy is softened in *macrogenre map* classification but still not satisfactory as KNN classification.

5 Pattern mining

In the following section, the *apriori* algorithm has been used for performing the search of rules inside the Spotify dataset. Every track has been considered a *transaction*, the items of the transaction were the data attributes.

5.1 Data preparation for pattern mining

The choice of attributes and outliers is the same as for classification, with the difference that, initially, the `genre` attribute has not been targeted.
The other difference between the other tasks, is that for the use of *apriori* algorithm, all continuous attributes have been discretized into intervals labels. The binning choice here has been a standard one: for every continuous attribute, 4 bins have been created using the `pandas.qcut()` function, that divides based on quartiles distribution; this guarantees every bin to have a substantial support count wrt all the others (every bin should contain $\sim 1/4$ of the data objects).

5.2 Pattern mining tasks

The tasks taken into consideration are the following:

1. Search of interesting rules involving specific *antecedents* or *consequents*;
2. Search of interesting rules for the attributes containing `NaN` values, in order to find specific filling methods based on the other attributes of the data object;
3. Search of interesting rules for classification. In particular it has been considered the *macro-genre map* of the attribute `genre`.

5.2.1 Preliminary trends - number of itemsets and rules

The crucial parameters of the algorithm `apriori` implemented by the library `fim` are the `supp`, that is the *minimum support* required for a dataset for being considered frequent, and the `conf`, that is the *minimum confidence* for a rule constructed from a frequent itemset for being considered valid.
The other metric of interest that has been considered for the evaluation of rules validity is the *lift*, that evaluates the dependence between the antecedent itemset and the consequent itemset. In particular, the interesting rules are the ones having *lift* considerably greater than 1.

Number of closed and maximal itemsets As preliminary, the number of closed and maximal itemsets has been studied as a function of `supp` parameter (confidence is defined for rules only). In Figure 22 the result is shown:

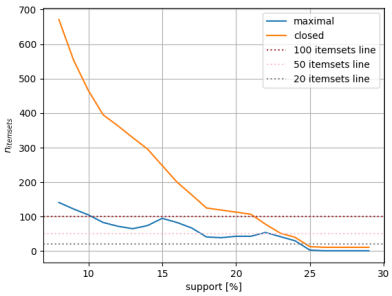


Figure 22: Number of itemsets as a function of the `supp` parameter of the `apriori` algorithm.

Max support rule and its meaning In order to exploit more parameters, the itemset having the maximum support has been found to be: `(time_signature = 4.0, explicit = False)`, with a support of the 84%. This itemset shows the most frequent values in the dataset. This itemset is not useful for implications, it only shows which are the most frequent values in the dataset and no more.

Rules with `supp = 20`, `conf = 60` and interpretation

After some trials of manual tuning, some rules having good lift has been found. Some of them are reported and interpretated below (the `-->` indicates the implication direction).

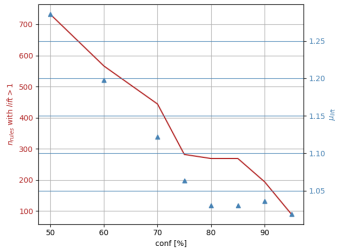
```
                    energy = (-0.001, 0.48] -->
--> (acoustic = (0.573, 0.996], explicit = False), lift = 2.7
```

The rule is the following: low energy implies high acousticness. It has to be noted that the value `explicit = False` is some kind of nuisance parameter, because of its high frequency and so it is contained in many rules but probably without a real implication meaning. As a confirm, the same rule with similar lift exists without `explicit = False` inside the itemsets.
The reversed rule exists and has similar lift, so it can be concluded that the algorithm with the lift measure is not able to discern the cause/effect, as expected, but has found anyway a good implication, whatever the direction.

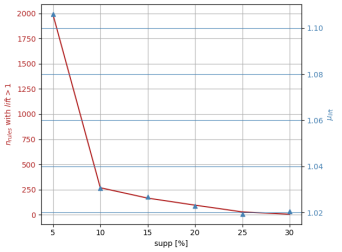
The following rule has as antecedent the interesting attribute `popularity`:

```
(popularity = (0.42, 0.94], time_signature = 4.0) -->
--> instrumentalness = (-0.001, 0.00313], lift = 1.4.
```

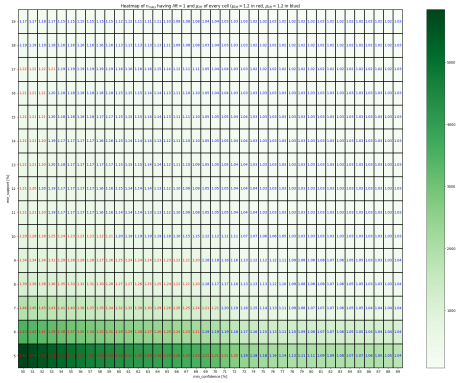
This rule does not show the reversed implication rule. It can be read as: *"if a song is popular, it is more likely to have low instrumentalness, that is, brief or no parts with only musical instruments"*.
In Figure 23 are shown the trends of the number of rules with the constraint `lift > 1.` as a function `conf` and `supp`.



(a) Number of rules as a function of minimum confidence



(b) Number of rules as a function of minimum support



(c) Number of rules and associated mean lift in a grid of minimum confidence and support parameters. In red, lifts higher than 1.2, in blue, lower than 1.2 .

Figure 23: Trend of the number of rules having lift higher than one as a function of the two parameters `supp` and `conf`.

Those trends can be used for tuning parameters, choosing a lift threshold for the rules.

5.2.2 Classification and missing values filling rules

Filling of missing values rules The first attribute of interest, being its dependance from the others not caught by the other techniques considered in this report, is `popularity_percent`. The interest of the rules having this attribute as antecedent is evident: it would identify the feature of

songs that are more likely of being famous; because of this, we have seek for rules having as consequent the bin with higher value of `popularity_percent`. Unfortunately, even with low support and consequence, there are zero rules with the wanted property.
Other interesting rules are the ones involving the attributes that in dataset contain missing values. Instead of using the pmf for filling, with rules containing as consequent implication `mode` or `time_signature`, it could be possible to create a more adaptive method.
For this task, every data attribute containing at least a missing value has been deleted (without considering `popularity_confidence`, having too much low support for finding any rule.) and the remaining counts observed. `time_signature` attribute has very inhomogeneous counts, requiring very low minimum support for the rules involving it and even so, no rules has been found for filling. The most realistic attribute to fill using this technique is `mode`, having counts of the same order of magnitude and having only two values. Setting `supp = 10, conf = 70`, a rule has been found having `mode = 1.0` as consequent:

```
key = 7 --> mode = 1.0, lift = 1.25
```

being the lift over 1 but not satisfying (data objects with `key = 7` have only 25% probability of having `mode = 1.0` wrt the data objects with other features), we have decided not to use this rule for `NaN` filling.

Classification Considering *macrogenre map*, pattern mining can be used for seeking rules for `genre` classification. Setting as consequent the specific values of `genre`, a sort of custom decision tree can be built, using rules as single splitting.
Only few macrogenra has shown the presence of rules that can be used for classification:

```
            (acoustic = (-0.001, 0.00974], dance = (0.695, 0.98]) -->
                  --> genre = 0, confidence = 0.89, lift = 2.3
(popularity = (0.14, 0.24], instrumentalness = (-0.001, 0.00313], mode = 1.0) -->
                  --> genre = 1, confidence = 0.64, lift = 1.70
(instrumentalness = (0.744, 1.0], duration_min =  (0.142, 3.0], energy = (-0.001, 0.48]) -->
                  --> genre = 6, confidence = 0.65, lift = 4.0
```

It has been constructed a toy classificator for the 3 macrogenra having specific rules for classification in order to evaluate its performance:

- It has been considered a single rule for every genre (highest lift one, shown above);
- uncategorized data object have been set to an unexistent category `genre = -1` in order to count the misclassified data objects.

In Figure 24 the resulting confusion matrix is shown:

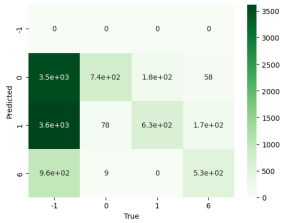


Figure 24: Confusion matrix for the toy classificator. It is possible to see that the most misclassified class is `genre = 1`, that corresponds to the rule with lower lift. The accuracy score is 18% even if the number of class is not high.

Clearly, this toy classificator is not performing, in particular, the biggest problem is the handling of misclassified values, that can be reassigned for example using more rules. Further improvements of this classifier has not been done in this project for space and time issues.

6 Regression

For the task of the regression, the most interesting analysis to carry outis the prediction of the feature `popularity`. From the correlation matrices, this attribute is not correlated substantially with any of the others. However, different regression analysis searching for relationships have been computed.

6.0.1 Linear regressor

The first regression performed used as predictor the feature `danceability`. In Figure 25, the blue

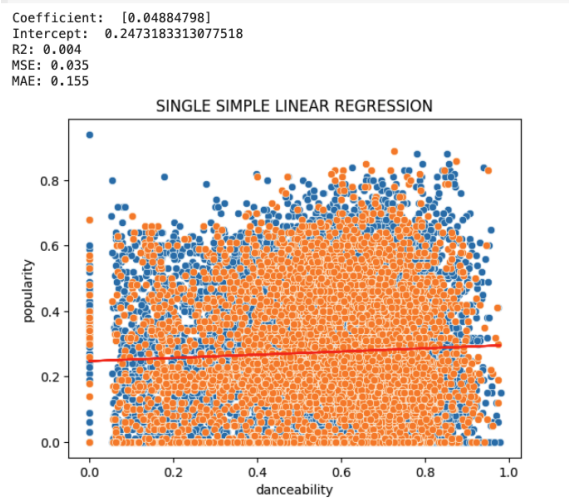


Figure 25: Single simple linear regression using `danceability` as independent variable. Above plot, the results of the regression: coefficient value and validation metrics.

dots represent the training patterns while the orange ones form the scatterplot of the test set. The regression line is increasing and the coefficient is positive, therefore a positive relationship between the two exists: an increase of 1 point percentage of "danceability" induces to a rise of the "popularity" of 5 point percentages approximately. But, as the scatterplots and the R_2 demonstrate, this correlation is very low: `danceability` explains only the 0.4% of the total variance of the popularity, a really low goodness index for this analysis. Indeed, the regression line, despite minimizing the errors or residuals, is in evident underfitting. A linear model, in this case, is too simple.

6.0.2 Multiple regressor

The next analysis performed was a single multiple linear regression, always for predicting the `popularity`, but this time using all the continuous variables at disposal. All the qualitative and discrete features have been dropped. While all the considered redundant features from data preparation have been kept to observe the property of regularization of the Ridge and Lasso regressors in this evident case of multicollinearity. Using the linear regression model, the R2 obtained was equal to 11.8%. Three variables affect the most the `popularity`: `duration_min` with a negative coefficient of -7.37, `danceability` with +4.02 and `energy` with a -4.18 . The MSE and MAE surprisingly didn't change from the simple regression despite a bigger number of dimensionalities. This could be explained again by underfitting.

6.0.3 Ridge and Lasso regressors

Ridge and Lasso regressors have been explored. For both the value of alpha, the regularization parameter, have been chosen. It has been tuned through a generalized cross validation on the training set. For the Ridge regression the chosen range of values was (1, 100). In Figure 26 the power of Ridge regularization is shown: as alpha increases, the coefficients decreases from their original value, or better, they shrunk toward zero and each other. This limits the multicollinearity between predictive features. The alpha obtained from the tuning was equal to the superior limit set in the cross validation.
The metrics obtained on the test set are shown in Table 13:
The R_2 didn't change from the multiple linear regression but the errors indices increased signifi-

R_2	MSE	MAE
0.118	0.882	0.771

Table 13: Validation metrics for Ridge regression on test set

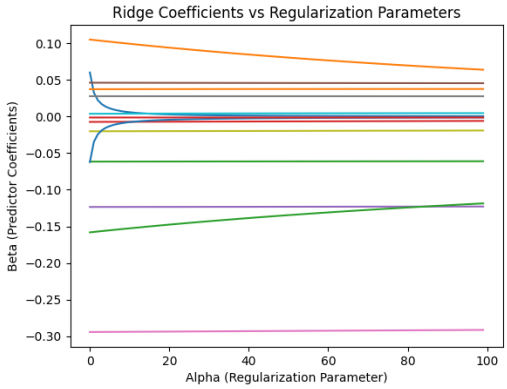


Figure 26: The regularization property of the Ridge classificator: as alpha increases, the value of the regressor all the coefficients beta tend to zero, expecially the multicollinear quantities ones.

cantly. The Lasso regressor model returned the same results but what's interesting is its property of feature selection: with a small alpha (alpha=0.01) 7 out of the 14 features were put equal to zero leaving only the most significant ones: `danceability`, `energy`, `speechiness`, `acousticness`, `instrumentalness`, `liveness`, `n_bars`.
Being the validation metrics not satisfactory for neither Ridge or Lasso regressor and being their values lower than 1%, coefficients has not been reported.

6.0.4 Final comment

Although several regression methods have been tried, no considerable results were obtained. It can be inferred that a linear model is not proper for `popularity` prediction.

References

[1] `https://www.spotify-song-stats.com/about`

[2] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar (2019). *Introduction to Data Mining - SECOND EDITION*

[3] Lele Fang, Junchen Shang, Nan Chen (2017). *Perception of Western Musical Modes: A Chinese Study*, `https://www.frontiersin.org/articles/10.3389/fpsyg.2017.01905/full#:~:text=The%20major%2Dmode%20music%20induced,not%20influence%20the%20dominance%20rating.`

[4] `https://www.mdpi.com/1999-4893/11/11/177`

[5] `https://github.com/clslabMSU/clustGUI`

[6] `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html`

[7] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html#scipy.cluster.hierarchy.linkage`

[8] Sander et al. (1998) *Data Mining and Knowledge Discovery 2*, Kluwer Academic Publishers

[9] `https://en.wikipedia.org/wiki/List_of_music_genres_and_styles`

[10] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html`