Data Mining I Project

Ondrej Krasnansky
Bernardo D'Agostino
Luca Coda-Giorgio

Academic Year 2022/2023

1 Data Understanding and Preparation

1.1 Data Semantics

The dataset analyzed in this study is a reworking of the RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) dataset, which contains audio-visual recordings of 24 actors saying short statements in English. Our dataset contains a total of 2452 instances associated with 38 features concerning synthesis measures and information of the audio files contained in the original dataset.

These last can be divided into categorical and numerical. In the following part, we'll briefly describe such features and the values or classes they assume in the above-mentioned dataset.

1.1.1 Categorical features.

The categorical ones, all of which are also nominal, with their respective classes, are: *'modality'* (audio-only): The format of the recordings, in our case only audio; *'vocal_channel'* (speech, song): The form the *'statement'*s are vocalized in; *'emotion'* (fearful, angry, happy, calm, sad, surprised, disgust, neutral): The emotional expressions the *'statement'* are spoken or sung with; *'emotional_intensity'*: (normal, strong). The intensity of the emotion of the statements. There is no strong intensity for the *emotion_neutral* observations; *'statement'* ("Kids are talking by the door", "Dogs are sitting by the door"): The actors' said and recorded sentences; *'repetition'* (1st, 2nd): Identifies the *'statement'*'s take number that was finally chosen; *'sex'* (M, F); *'actor'* (from 1 to 24): These values represent the ID number of the actor speaking or singing the *'statement's* It's also important to note that the odd-numbered actors are males, while the even ones are females; *'channels'* (1, 2): While 1 stands for mono, 2 identifies stereo audio; *'sample_width'* (1, 2): In this case, the values reveal the number of bytes per sample. 1 means 8-bit and 2 means 16-bit. It's important to note that all observations in our dataset have 16 bits; *'frame_width'* (2, 4): The frame width conveys the number of bytes for each frame, and one frame contains a sample for each channel.

1.1.2 Numerical features.

The majority of the dataset's variables are continuous numerical features. These are: *'frame_rate'*: Such feature describes the frequency, expressed in Hertz, of the samples used. It's important to note that all the samples in the dataset have the same frame rate; *'length_ms'*: This feature indicates the audio file's length, in milliseconds; *'zero_crossings_sum'*: Zero crossings is a property of the audio signal that allows a rough estimation of its dominant frequency and Spectral Centroid (SC). Their rate and, in this case their sum are commonly used features for speech-music discrimination; *'frame_count'*: Expresses the number of frames from the sample; *'intensity'*: Such attribute is a measure of loudness, in dBFS (decibels relative to full scale; the level of 0 dBFS is assigned to the maximum possible digital level, this explains why, in the dataset, only values in a negative interval are assumed); *'mean', 'std', 'min', 'max', 'kur', 'skew'*: Statistics of the original audio signal; *'skew'* stands for Skewness, which is a measure of the lack of symmetry in distributions; *'kur'*, Kurtosis, is a measure of the extremity of deviations in the data, relative to a normal distribution; *'mfcc_mean', 'mfcc_std', 'mfcc_min', 'mfcc_max'*: Statistics of the Mel-Frequency Cepstral Coefficients; *'sc_mean', 'sc_std', 'sc_min', 'sc_max', 'sc_kur', 'sc_skew'*: Statistics of the Spectral Centroid (SC); *'stft_mean', 'stft_std', 'stft_min', 'stft_max', 'stft_kur', 'stft_skew'*: Statistics of the STFT (Short-Time Fourier Transform) Chromagram.

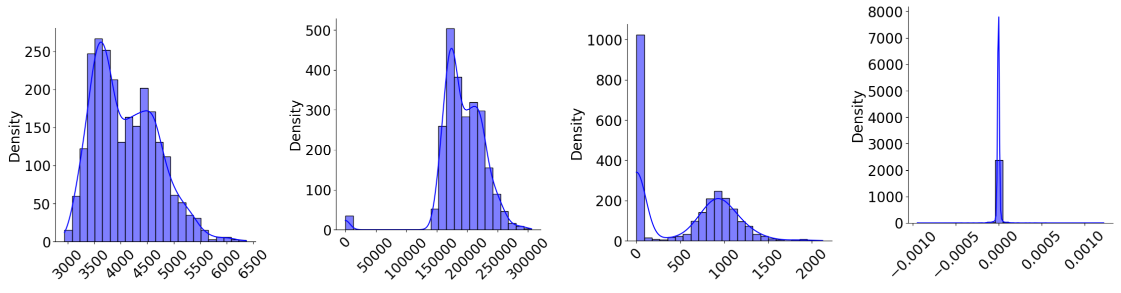
1.2 Distribution of the variables and Statistics

In order to thoroughly understand the features, their properties, and their characteristics, before leaping into the data-cleaning phase of our analysis, we must study the distribution of each feature and examine some insightful descriptive statistics related to the dataset at hand.

Analyzing these features we found that the variables *'modality', 'sample_width', 'frame_rate'* and *'stft_max'* all only had a single value, so we chose to eliminate them since they didn't carry any information. We also found that the variables *'channels'* and *'frame_width'* were the same variables since a stereo audio file has double the bytes of a mono audio file in each frame, and since there were only 6 observations that were recorded in stereo mode, we chose to treat them as outliers and to eliminate them from the dataset, we then proceeded to eliminate both *'channels'* and *'frame_width'*.

1.2.1 Continuous features' distribution

Observing the continuous variables' distribution we note that, while most of them present Gaussian-like distributions, some are definitely skewed (negatively or positively), and other ones, instead, have some eye-catching peculiarities (Figure 1).



(a) *'length_ms'* (b) *'frame_count'* (c) *'sc_min'* (d) *'mean'*

Figure 1: Distribution of peculiar continuous variables.

The *'length_ms'* feature has a bimodal distribution, as shown in Figure 1a. This is because songs and speech recordings follow two distinct normal distributions. On average, songs are 950 milliseconds longer than speech recordings, as shown in Figure 2. We also found the variable *'min'*

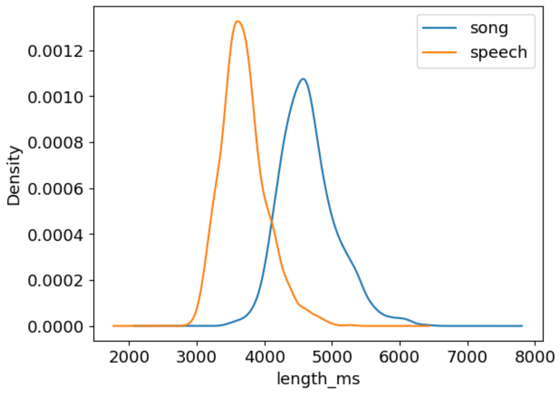


Figure 2: Distribution of *'length_ms'* by *'vocal_channel'*

to be negatively skewed, and the variables *'sc_kur', 'stft_min', 'stft_kur', 'mean', 'std', 'max'* and *'kur'* to be positively skewed.

A deeper examination is needed for the attributes *'frame_count', 'sc_min'* and *'mean'*.

Looking at their distribution, we notice that *'sc_min'* (Figure 1c) has a relevant number of instances assuming the value zero, followed by a shorter, but normal, distribution. The same can be said, although with the value -1 and for only 35 observations, for the *'frame_count'* variable (Figure 1b). We chose to interpret these observations as missing values since we think that this phenomenon can be explained by the fact that at times datasets are filled with specific values to indicate missing data instead of leaving empty observations.

The *'mean'* variable (Figure 1d) is also worthy of discussion. The feature has a distribution that is undeniably very concentrated on the value zero. At the same time, it possesses a very low standard deviation and a lot of outliers having 516 observations being outside the interquartile range times 1.5. We believe that this is caused by the oscillating nature of the original audio signal, which being in the form of a wave tends to have a mean value close to zero as the signal oscillates between positive and negative values of similar intensity.

1.2.2 Categorical features' descriptive statistics.

Examining Table 1, we can note how the *'emotional_intensity', 'statement', 'repetition'* and *'sex'* binary features are very balanced. In fact, the frequency of their assumed classes is exactly (or close to) split in half in terms of instances in the dataset.

The last categorical feature is *'emotion'*. It has 8 different classes, mentioned in the previous section, whose frequencies are shown in Figure 1, surprised and disgust emotions have half the frequency of the majority of emotions since there are no song observations with these emotions, while neutral emotions have half the frequency since there are no observations with this emotion that are said in a strong emotional intensity.

Table 1: Descriptive statistics: Categorical features

Feature	Classes	Frequency
vocal_channel	speech	0.544454
	song	0.375612
	missing values	0.079934
emotion	fearful, angry, happy, calm, sad	0.153344
	surprised, disgust	0.078303
	neutral	0.076672
emotional_intensity	normal	0.538336
	strong	0.461664
statement	“Kids are talking by the door”	0.500000
	“Dogs are sitting by the door”	0.500000
repetition	1st	0.500000
	2nd	0.500000
sex	M	0.508972
	F	0.491028

1.2.3 Pairwise correlations

We proceeded in our analysis by calculating the Pearson correlation coefficient between all the 26 remaining continuous features in our dataset, Figure 3.

First of all, we found that *'length_ms'* and *'frame_count'* had a correlation coefficient of 1. This was obvious since as the frame rate is the same for all observations then *'frame_count'* and *'length_ms'* must be multiples of each other, ($\frac{length\_ms}{frame\_count} = \frac{1}{frame\_rate} = \frac{1}{4800} = 0.020833$) So, we chose to eliminate *'frame_count'* and keep *'length_ms'* as it didn't have missing values.

We then chose to eliminate the mean variable since it didn't have any meaningful correlation with any other variable in the dataset Figure 3. We believe that this was because of its low variance,

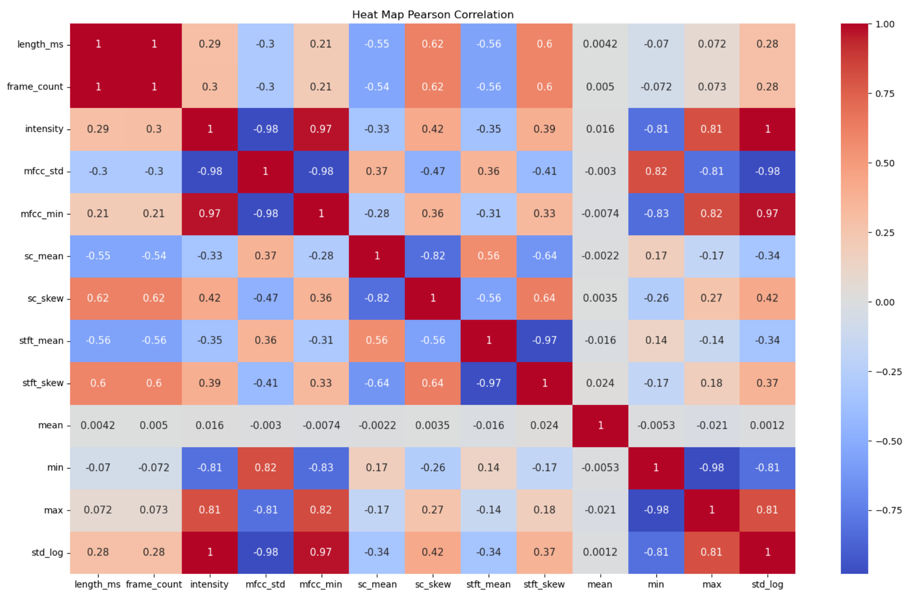
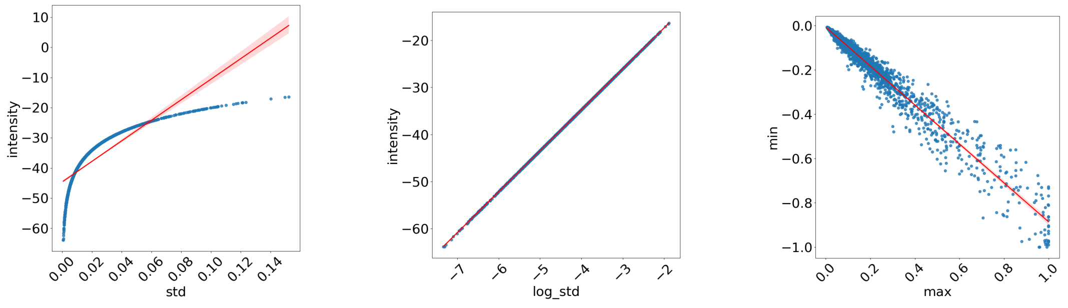


Figure 3: Pairwise correlation heat map of the most correlated variables

caused as we said before by the oscillating nature of audio signals which naturally have similar and close to 0 mean values.

As can be seen from the correlation matrix we found multiple pairs of variables with very high correlation. All 8 pairs of variables with an absolute value of the Pearson coefficient greater than 0.9 had a very strong linear relationship with each other. For reference in Figure 4 it's possible to see the scatter plot of the variables *'min'* and *'max'*. This relation in particular can again be explained by the oscillating nature of the original audio signal, which after each peak in its oscillation has a similar value but of opposite sing, explaining the strong correlation between the two variables.

We then noticed that the remaining pairs of features with a correlation coefficient greater than 0.8 all showed very strong logarithmic relations. For reference, we showed in Figure 4 the clearest relation we found, which was between std and intensity. After applying a log transformation to std the relationship between the two variables became perfectly linear as can be seen from Figure 3. So in order to not repeat the same feature, we chose to eliminate *'intensity'* since it had missing values. This relation is also explained by the nature of audio signals, as they are more intense the farthest they are from their base value of 0. So, since the standard deviation is the root of the mean squared distance from the mean, which as explained before is always very close to 0, *'std'* is exactly a proxy for the intensity of an audio signal.



(a) *'std'* vs *'intensity'* (b) log(*'std'*) vs *'intensity'* (c) *'max'* vs *'min'*

Figure 4: Scatter plot of two pairs of highly correlated variables, with a linear regression for reference

We then chose to eliminate the variables *'min'*, *'max'*, *'mfcc_std'*, *'stft_mean'*, *'sc_skew'* and *'mfcc_min'*, since, as we showed before, we thought their linear and logarithmic relations to be so strong as to make them almost copies of each other, and we thought that this would influence the distance measure used in the following subset distribution and clustering analysis.

The *’min’* and *’max’* features were strongly correlated to *’std’* while *’mfcc_min’* and *’mfcc_std’* were strongly correlated to intensity so they all were a proxy for intensity, while *’stft_mean’* was strongly correlated to *’stft_skew’* and *’sc_mean’* was strongly correlated to *’sc_skew’*.

We believe that all these strong correlations between different features are explainable by the fact that they all are statistics extracted from 4 different representations of the same audio signal and so represent the same characteristics but in different ways.

To conclude, this correlation analysis was a really useful tool to understand the relations between our variables and to allow us to eliminate redundant variables and reduce the dimensionality of our dataset.

1.2.4 Subsets’ distribution.

In the following analysis, we chose to exclude the *’sc_min’* variable since it contained many missing values and would have led to an excessive information loss.

We proceeded with our analysis of the data by subdividing it according to the different categories each observation could be a part of and trying to find the continuous feature over which these categories differed the most. To accurately compare the differences between the classes we first scaled all the continuous features in our dataset as will be discussed in paragraph 1.3.2.

For each category, we split the dataset accordingly and calculated the mean values for each continuous feature, and we considered their absolute difference.

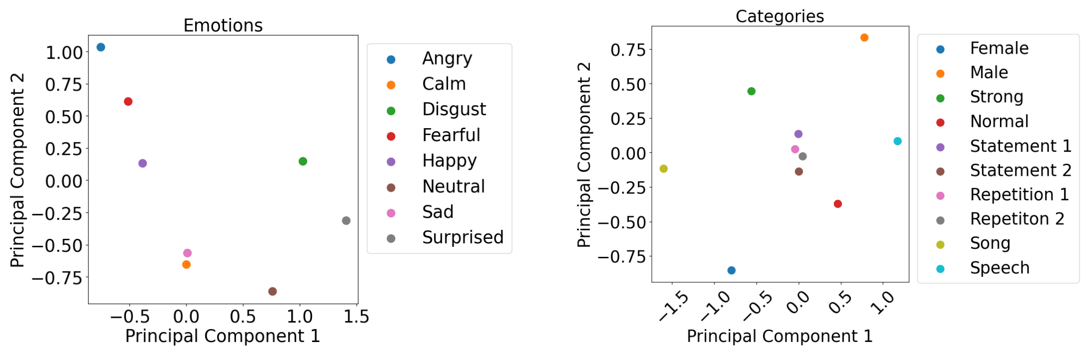


Figure 5: Centroid of each emotion and category represented in the principal component plane.

We discovered that female observations have on average a lower *’mfcc_mean’* -1.08, *’stft_min’* -0.84 and *’mfcc_max’* -1.09, and a higher *’stft_std’* 1.15 and *’stft_skew’* 0.92 than male observations. We also found that song observations are longer on average, as shown before in paragraph 1.2.1, and have a lower *’sc_mean’* -1.07, *’kur’* -1.06 and *’mfcc_max’* -0.58 while having a higher *’stft_skew’* 1.06 and *’stft_std’* 0.83. Observations said in a strong emotional tone have on average a higher *’std’* 0.8, which is a proxy of intensity as said in the previous paragraph. While observations didn’t differ much between repetitions and the two statements.

We then compared categories in a multidimensional analysis, so we calculated a distance matrix of the mean point for each category and then plotted these points in 2 dimensions, which explain 30% and 20% of variability each, using PCA dimension reduction (*PCA* algorithm from *sklearn*) as shown in Figure 5. From the plot is possible to see that the observations differ the most by *’vocal_channel’* with songs and speech observations having a mean euclidean distance of 2.79, then the second category of most importance is sex with a distance of 2.51 and after it emotional intensity with a distance of 1.48.

We then moved our analysis to emotions, we initially did a univariate comparison between the observations belonging to each emotion, finding the biggest differences over the *’std’* and *’length’* variables. Angry observations, as was predictable, are by far the loudest with a much higher *’std’* 1.37 than all the other emotions, while we found the calm emotions to be the quietest ones -0.66.

We also found the calm, sad and angry observations to be the longest and surprised observations to be the shortest with an average difference between calm and surprised observations of 971 milliseconds.

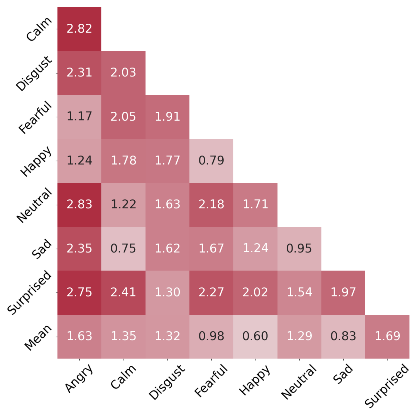


Figure 6: Distance matrix of the emotion centroids.

We then compared emotions in a multidimensional analysis in the same way we did for the binary categorical features, the results are shown in Figure 5 while Figure 6 represents the distance matrix between the mean points of each emotion.

From both the matrix and the plot it's possible to note that the angry and surprised observations are the farthest away from the mean of the entire dataset, while angry observations have the largest absolute distances from the calm, sad and surprised emotions. We can see also that Happy observations are the closest to the mean, and are also very close to fearful observations, while Sad and calm observations are the closest to each other.

1.3 Assessing data quality and Variable Transformation

1.3.1 Data Quality

In our opinion, this dataset has two important shortcomings, the first is redundancy caused by the fact that most of its continuous features are statistics extracted from 4 different representations of the same audio signal, with many of the features in our dataset being so correlated to others that they are almost copies of each other. The second is the high presence of missing values, indicated in 3 different ways in 5 different features.

1.3.2 Feature transformation

From our previous analysis, we found the variables *'sc_ kur', 'stft_ min', 'stft_ kur', 'mean', 'std', 'max'* and *'kur'* to be positively skewed and the variable *'min'* to be negatively skewed. Since we have already eliminated *'max', 'min'* and *'mean'*, we only have to address the remaining 5 features.

We chose to correct them since we thought that the strong asymmetry present in these variables could influence our clustering and classification task.

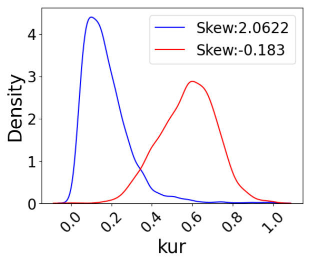
We used a logarithmic transformation on our positively skewed variables and after this transformation, the distribution of all the variables approached a normal distribution see Figure 7.

After this, we chose to apply a Z-Score standardization (*StandadScaler* function from *sklearn*) to all the continuous features in the dataset to reduce scale differences that would affect the distance measures used for the clustering in the next chapter.

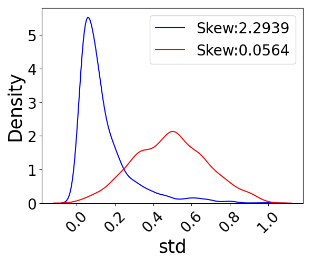
1.3.3 Missing Values

There are 2559 missing values in the dataset, 196 in *'vocal_ channel'*, 1126 in actor, 816 in *'intensity'*, 1016 in *'sc_ min'*, and 35 in *'frame_ count'*, and we have already eliminated the *'intensity'* and *'frame_ count'* variables so we only need to handle the remaining three features.

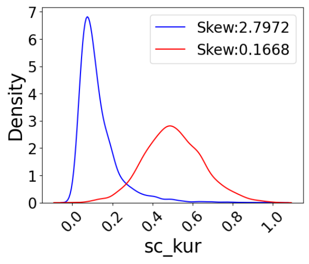
We chose to eliminate the *'actor'* feature since it was missing 46% of its observations, and we thought that an imputation on the missing values would be too imprecise, also given the fact that it would have been a 20-class classification problem. We instead chose to impute the missing values for the *'vocal_ channel'* and *'sc_ min'* features to avoid information loss.



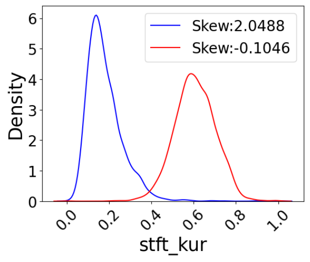
(a) *'kur'*



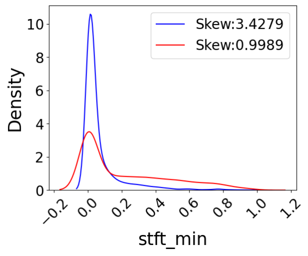
(b) *'std'*



(c) *'sc_ kur'*



(d) *'stft_ kur'*



(e) *'stft_ min'*

Figure 7: Distribution of skewed variables before (Blue) and after (Red) log transformation.

For the imputation of *'vocal_ channel'*, we first imputed all disgusted and surprising observations as speech, as there are no observations with such emotions that are also songs, we then used a mixed distance K-NN model which will be described in more detail in chapter 3.

For the imputation of *'sc_ min'*, we used a Linear regression model with an L2 (Ridge) penalty, which will be described in detail in chapter 4.

We used the imputed values in both the clustering and pattern-mining chapters of our work.

1.3.4 Outliers

In this part, we are going to comment on and analyze outliers found in our data set. We initially started by analyzing mono-dimensional outliers finding all the observations that had values outside the interquartile range times 1.5. This method found 546 outliers, mainly in the *'sc_ max'* feature which contained 435 outliers in itself. So we chose to eliminate the *'sc_ max'* feature and proceed to multidimensional outlier analysis.

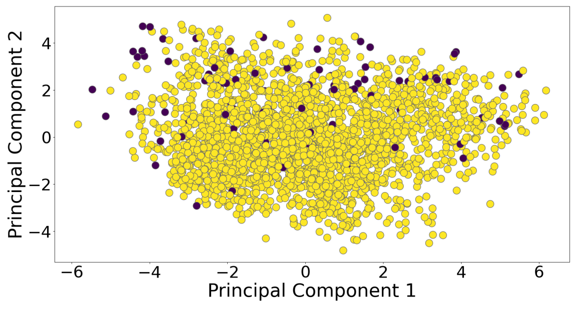


Figure 8: Distribution of the outliers found with *DBSCAN* clustering over the first 2 principal components.

So we proceeded by trying to find multi-dimensional outliers with the DBSCAN clustering algorithm (*DBSCAN* from *sklearn*) that will be described in more detail in paragraph 2.1. After running the algorithm we found 89 observations not belonging to the single main cluster, shown in Figure 8, which are robust to multiple values of the clustering parameters. The fact that only so few observations were external to only a single cluster indicates the dense and compact nature of our data.

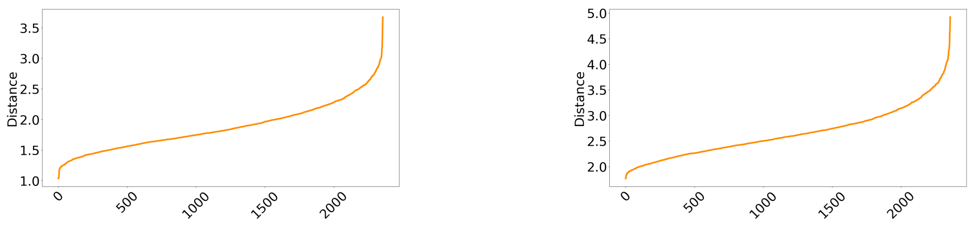
70% of the outliers found in this way have a strong emotional intensity, only 20% of these are songs and also fearful observations are disproportionally represented being 30% of the outliers. We chose to eliminate these outliers as we believe they could have affected our clustering.

2 Clustering

In our cluster analysis, we chose to use the Euclidean distance as a distance measure and to only use the continuous standardized features of the dataset pre-processed as described in the previous chapter, since the main aim of our analysis is to find whether the clusters we will find will fit the classes present in the dataset.

2.1 DBSCAN

We applied the DBSCAN clustering algorithm (*DBSCAN* from *sklearn*) to our dataset preprocessed as described in the previous paragraphs. To do so we had to find the right values for the *eps* (the maximum distance of the neighborhood of a point) and *minSamples* (the minimum number of neighbors to be a core point) parameters, so we calculated the distance to the *k-th* neighbor for all points and plotted them in ascending order, we found that the shape of the generated curve didn't significantly change for values of k between 2 and 10, or for k set to 50, the curves relative to k=6 and k=50 are shown in Figure 9.



(a) *6-th* neighbour (b) *50-th* neighbour

Figure 9: Ordered distance from the 6*th* and 50*th* neighbour for each point.

We chose to set the number of neighbors equal to 6, and we found the corresponding value of the *eps* parameter over the second inflection point to be 2.5. We ran the *DBSCAN* algorithm with these parameters over the continuous features in our dataset, the result, shown in Figure 8, is a single continuous cluster with 2357 observations, with 89 individual points left outside, a description of these points can be found in paragraph 1.3.4. These external points are robust to changes in the *minSamples* parameter and its respective optimal *eps* value. After the removal of the 89 individual points, we ran the clustering again and obtained a single cluster with all 2357 observations in it.

The results of this clustering analysis suggest that we are working with a dense dataset without particular aggregations of points external to the main corpus of the data. This can also be inferred visually by looking at the representation of the data over the first 2 principal components, Figure 8.

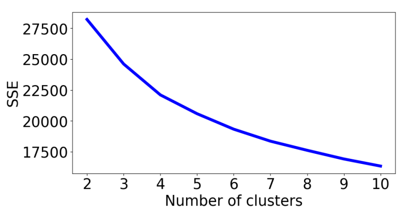
2.2 K-Means

We then proceeded in our work applying the *K-Means* algorithm to the dataset pre-processed and without multidimensional outliers. To determine the optimal number of clusters for the algorithm we computed both the *SSE* and the *Silhouette* measures associated with all numbers of clusters from 2 to 10. The results didn't meaningfully change over multiple reinitializations of the cluster centroids and are shown in Figure 10.

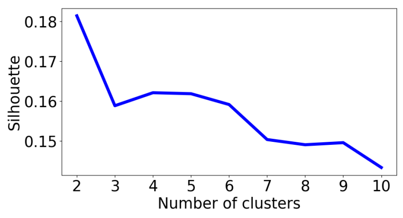
We couldn't find a significant 'elbow' in the SSE plot, but from the Silhouette analysis, we found the best result for 2 clusters 0.18, so we proceeded with our analysis with this parameter.

The two clusters we found are almost exactly split by a threshold on the value 0 of the first principal component (which explains 30% of the variability) see Figure 16a. And their centroids also have similar values for all the remaining principal components except the first.

The 2 clusters we found, which will be called 0 and 1, are similar in size having 1314 and 1043 observations each. We found that while the *'repetition'* and *'statement'* categories are balanced

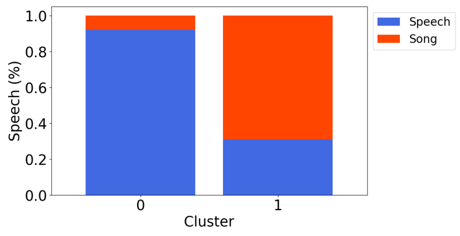


(a) SSE

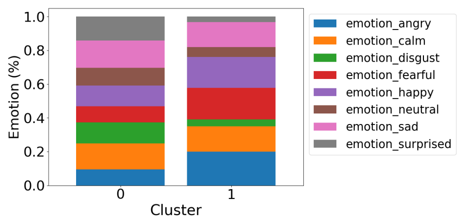


(b) Silhouette

Figure 10: Plot of the *SSE* and *Silhouette* measures over different numbers of clusters.



(a) *'vocal_ channel'*



(b) *'emotion'*

Figure 11: Percentage of Songs and Emotions in each cluster.

between the two clusters, female samples are more present in cluster 1 56% while males are in cluster 0 59%, and also emotionally strong observations are more present in cluster 1 55% than in cluster 0 32%. But the two clusters are mainly divided over the *'vocal_ cannel'* category, with speech observations being much more present in cluster 0, where they represent 92% of the population than in cluster 0 where they are only 30% as shown in Figure 11. These differences in the classes strongly follow the differences represented in Figure 5, where it can be seen that the mean points for song and speech observations strongly differ over the first principal component in the same way as the two clusters we found.

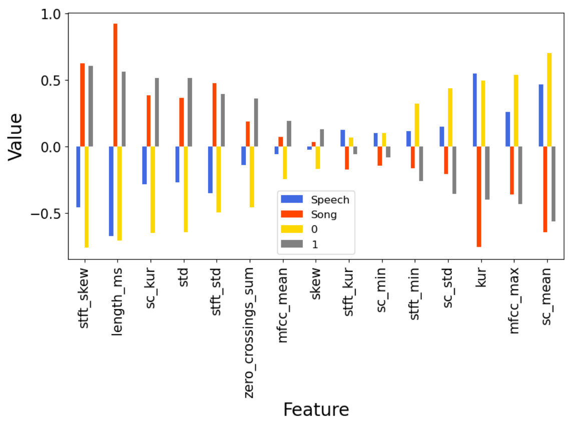


Figure 12: Mean values of each feature grouped by cluster and by *'vocal_ channel'*.

The 2 centroids of the clusters have opposite values for almost all features. These differences closely follow the difference between the mean values of song and speech observations as shown in Figure 12, reiterating how central the *'vocal_ channel'*, split is to the division of the 2 clusters. Observations in cluster 1 have a higher *'stft_ skew'* by 1.36 and are 756 milliseconds longer on average, while observations in cluster 0 have a higher *'sc_ mean'* by 1.26. The two clusters have instead very similar values for the *'stft_ kur'* and *'sc_ min'* features having between them an absolute difference of less than 0.2.

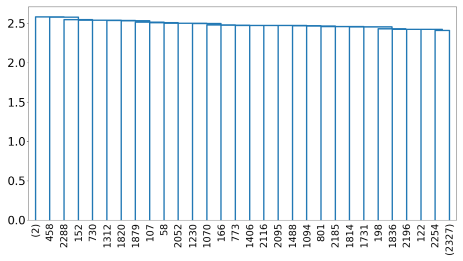
We also found that cluster 0 had more surprised, disgusted, and neutral observations and cluster 1, instead, had more angry, fearful, and happy observations, while sad and calm observations were equally split between the 2 clusters. These findings are consistent with the analysis previously done in paragraph 1.2.4, about the distances between emotions, with sad and calm observations being very close to each other and being in the center of the first principal component, while angry, fearful, and happy observations were in the left side of the plot and so are more present in cluster 1 with the opposite being true for surprised, disgusted, and neutral observations. This was also caused by the fact that surprise and disgust observations are never songs and so are mainly present in cluster 0.

2.3 Hierarchical

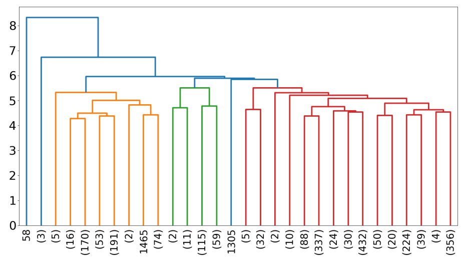
We then proceeded to apply the *Hierarchical Clustering* algorithm to our dataset. This allowed us to seize a different perspective not having to fix the number of clusters upfront.

We tested the *Single Link* method (which takes the minimum of the computed pair distances between all the instances in the two clusters), the *Complete Linkage* method (which takes, instead, the maximum distance), the *Group Average* method (in which the inter-cluster distance is taken as the average of the distances between each pair of observations, one from the first and one from the second cluster) and *Ward's* method (which merges the clusters taking into account the additional SSE resulting from the merge itself).

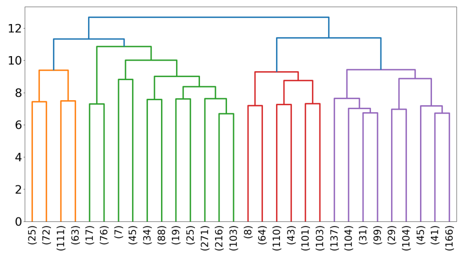
In order to do so, we applied the *linkage* and *dendrogram* functions from the *scipy* library and we evaluated the results given by the implementation of each different aforementioned criterion. To detect the finest linkage criteria we also exploited the quantitative metric *CPCC* (Cophenetic Correlation Coefficient) to assess the dendrograms' ability to keep the pairwise distances between the original points stable. The obtained dendrograms are shown in Figure 13.



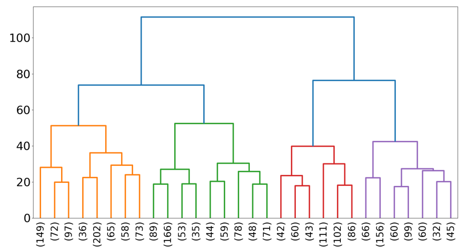
(a) Single Link
CPCC = 0.3478



(b) Group Average
CPCC = 0.5242



(c) Complete Linkage
CPCC = 0.3911



(d) Ward's linkage
CPCC = 0.4274

Figure 13: Dendrograms resulting from the implementation of different merge criterion.

As Figure 13a reveals, the *Single Link* method provides the worst results. In fact, starting from a cluster inclusive of practically every instance of the dataset (2327 out of 2452), this approach makes a point-by-point merge of the remaining ones to such cluster, while also having the lowest CPCC. This stems from the fact that the clusters are very close to each other.

Looking at the *Group Average* approach (Figure 13b), we can notice how we can actually recognize three different significant (in terms of dimension) clusters by "cutting" the dendrogram at the displayed level. Even though the CPCC is the highest among the four approaches, the potential clusters aren't well-defined and separated enough to make any assumptions. A slightly higher threshold would group almost all data points together.

In contrast, we get some interesting results evaluating the *Complete Linkage* and *Ward*'s methods. Both have relatively high coefficient values and both visually suggest, in our opinion, the presence of four distinct clusters, also 2 in the *Ward* method. As we can notice, each of the clusters is quite well distinguished from the others (especially in *Ward*'s linkage).

Ward's method is the closest, by properties and efficiency, to K-Means clustering, in fact, both have the objective function of minimizing the SSE. It's possible to see how both the silhouette plot for the K-Means algorithm and the dendrogram for the *ward* method suggest the presence of 2 or 4 clusters. Analyzing the results of the two algorithms with the number of clusters set to two we found that 87% of the observations were assigned to the same cluster by the two methods (*Ward*'s and K-Means).

Since we already found interesting results with K-Means in the previous paragraph, chosing *Ward's* as a linkage for this type of clustering felt redundant. Therefore we chose to consider the *Complete Linkage* criteria instead.

We then proceeded with the *Compleate*'s with four clusters, the resulting clustering achieved a silhouette score of 0.09. Next, to understand the differences between the clusters and their characteristics, we compared the mean values assumed by each variable among the clusters, while also looking at the distribution of the categorical ones. The clusters, for simplicity, will be named Cluster 0, 1, 2, and 3 (Figure 16c).

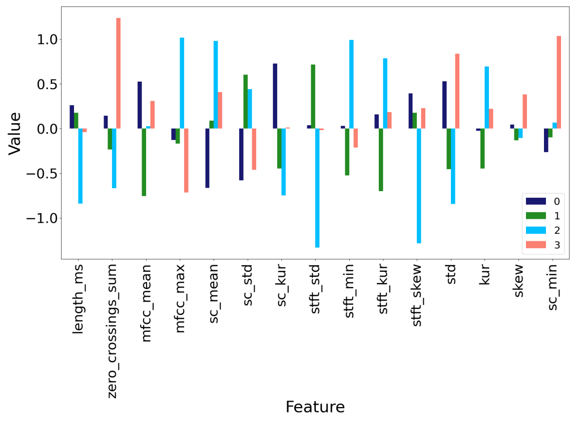


Figure 14: Comparison of the continuous variables' mean values by cluster

Clusters Comparison. The distribution of emotions in each cluster and their differences over the countinuose variables can be seen in Figures 14 and 15

We didn't find any significant difference among the clusters over the distribution of the *'statement'* and *'repetition'* variables, as was for the K-Means analysis, reiterating the irrelevance of these classes for the clustering.

Cluster 0 is the largest among the 4 having 901 observations. These observations are, on average, longer and have the highest intensity among all. There is a prevalence of strong *'emotional_ intensity'* observations while they are balanced over the *voacl_ channel* and *'sex'* features. The *'mfcc_ max', 'sc_ kur'* and *'stft_ skew'* variables assume high values, while *'sc_ mean', 'sc_ min', 'sc_ std'* and *'sc_ min'* are low. We find that the angry, fearful, and happy emotions are the most present (cumulatively 64%), while about 20% are calm or sad. This coincides with the fact that usually, observations with these emotions are lauder as the observations in this cluster. These results also match the ones discussed in paragraph 1.2.4 about the multidimensional analysis of each emotion's centroid, as this cluster centroid, Figure 16 is close to the centroids of the happy, angry and fearful emotions Figure 5.

Cluster 1, is made of 756 instances and is characterized by observations shorter than the ones in the first cluster, but longer than the other two. This cluster is also composed of quiet observations having a low mean *'std'* and 70% of observations with a normal *'emotional_ intensity'*. We can also notice a preponderance of female recordings (70%), while *'vocal_ channel'* is balanced between songs and speeches. 46% of the files have a calm or sad tone, and fearful, happy, and neutral emotions combined represent around 36% of the records inside this cluster. Additionally, *'sc_ std'* and *'stft_ std'* are relatively high and *'mfcc_ mean', 'sc_ kur', 'stft_ min', 'stft_ kur'* and *'kur'* are low, also these results are coherent with the analysis done in Figure 5, with female, calm and sad observations being located close to the centroid of cluster 1.

Cluster 2 (composed of 429 instances) has the shortest and least loud observations. These also have low *'zero_ crossings_ sum', 'sc_ kur', 'stft_ std'* and *'stft_ skew'* values. In contrast, they assume very high values for the *'mfcc_ max', 'sc_ mean', 'sc_ std', 'stft_ min', 'stft_ kur'* and *'kur'* variables. The recordings are nearly exclusively speeches (97%) of male actors (89%) and only 30% of them have a strong emotional intensity. All the emotions, except fearful and angry (which are not significantly present), are split quite evenly in this cluster, with frequencies from the lowest, 12% for happy and neutral, to the highest, 17% for calm, this result also can be tied to Figure 5 since the centroids for male and speech observations are located in the top left of the plot close to the centroid of cluster 2.

Finally, the last cluster (Cluster 3), which is the smallest having 271 instances, has high *'zero_ crossings_ sum', 'mfcc_ mean', 'sc_ mean', 'skew'* and *'sc_ min'* and the maximum intensity of the whole set of clusters, while having, instead, low *'mfcc_ max'* and *'sc_ std'*. The dominant sex is female (75%) and the statements are being said with strong emotional intensity in 71% of the cases. In this cluster we also find that nearly 70% of the recordings are spoken and not sung by the actors and they are mainly vocalized with the angry (28%), fearful (21%), happy and sad (respectively 16% and 14%) emotions, it's interesting to see that it's centroid is close to the one of cluster 0 for the first two P.C. Figure 16c.

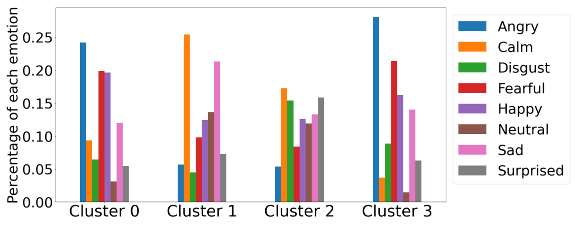


Figure 15: Distribution of *'emotion'* classes among the clusters.

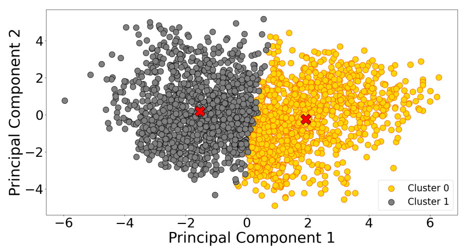
2.4 General clustering discussion

The results of the 3 clustering techniques we applied to the dataset can be visualized in Figure 16. The density-based algorithm DBSCAN didn't give any fruitful results as it found a single cluster containing the vast majority of all observations. But from this and the visualizations of our data, we can infer our dataset's dense and globular shape.

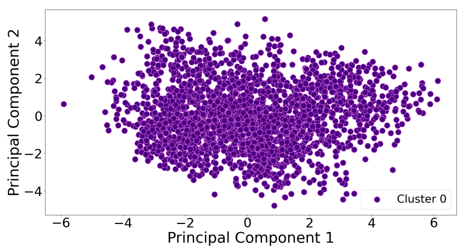
Regarding the other two more interesting results we achieved, we decided to compare them analyzing the correspondence between their cluster labels and the categorical variables contained in our dataset. To do so we initially calculated the weighted average entropy for each category in each clustering and on the original dataset. The K-Means clustering has a very low entropy value of 0.29 on the *'vocal_ channel'* class compared to the hierarchical clustering 0.4 and the original dataset 0.48 while having comparable entropy values for all other categories with respect to the original dataset. While the hierarchical method has lower values of entropy for the *'vocal_ channel'* class as indicated before, and also for the *'sex'* class with an entropy of 0.38 compared to 0.48 and 0.49 of the K-Means and original dataset.

We continued our analysis by calculating the purity for each category over the two sets of clusters, the results of this analysis confirm what we discovered before, with the K-Means clustering having a very high purity of 0.8 for the *'vocal_ channel'* category and the hierarchical method having high purity for the *'sex'*, 0.72 and *'vocal_ channel'* 0.69 categories, and with a purity of 0.67 for the *'emotional_ intensity'* category.

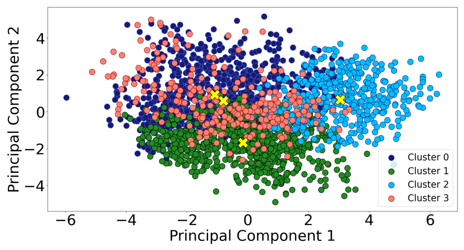
The results of these clustering techniques suggest the presence of one single dense cluster of points in our dataset that can be naturally subdivided following the *'vocal_ channel'* and *'sex'* categories. The K-Means clustering technique mainly captured the speech song separation while the hierarchical clustering managed to capture both.



(a) K-Means



(b) DBSCAN



(c) Hierarchical

Figure 16: Representation over the 2 first principal components of the result of the clusterings.

3 Classification

For our classification task, we chose as target the binary *'vocal_channel'* category as our target variable, since it was the most influential category in our cluster analysis and we wanted to test its predictability and also because we could use the classification task to impute it's missing values.

We decided to use the unprocessed dataset for our model evaluation, as the pre-processing methods we used would have biased the results if applied to the entire dataset, rather than only the training set

As a first step, we eliminated all variables with a single value and removed the variables *'sc_min'*, *'actor'*, *'intensity'*, and *'frame_count'* due to missing values. Next, we removed the remaining missing values in the *'vocal_channel'* category, which we later classified using the best-performing model. We then removed all observations of the surprised and disgust categories, as they are only spoken, leaving us with 2063 observations, 44% of which were songs and 56% speech. We split our data into a training set (70% of the observations) and a test set (30% of the observations).

3.1 Naive Bayes

The first classification algorithm we chose to implement was the naive Bayes classifier, we chose to estimate the probability of the continuous features in our dataset with a normal distribution, and since we had a mixed-type dataset with both categorical and continuous features we employed the *mixed_naive_bayes* algorithm from *MixedNB* library.

The Naive Bayes algorithm is not sensitive to the scale of the features because it relies on probability estimation rather than distance measures. Therefore, we did not scale the continuous features in our dataset. However, the classifier is not robust to skewness, outliers, and high correlations between predictors. To address this, we tested the accuracy of the classifier on two versions of the same training and testing sets: one without any modifications, and one with log-transformed variables (as discussed paragraph 1.3.2) and the removal of highly correlated features and uni-variate outliers from the training set using the interquartile method (as discussed in paragraph 1.2.3 and 1.3.4).

From now on for simplicity, the model fitted on the original data will be called Original NB and the other Treated NB. To account for the class imbalance in the training set (where there are 24% more speech observations than songs) we trained a third model, called Balanced NB by under-sampling an equal number of songs and speech observations from the training set.

We fitted the three versions of the Naive Bayes model (Original NB, Treated NB, and Balanced NB) to the corresponding training sets and used them to predict the class labels on the testing set. To evaluate the performance of the models, we calculated their accuracy on the testing set and the recall, precision, and F-1 index for both classes. The recall indicates the percentage of correctly classified observations of a class, the precision indicates the percentage of observations of a class that are correctly classified, and the F-1 index is the harmonic mean of precision and recall.

To thoroughly evaluate the accuracy of our model's predictions, we conducted a cross-validation study on the three models. We trained and tested each model ten times, using different splits of the data for training and testing each time. For each trial, we calculated the values of our chosen performance measures: accuracy, precision, recall, and F1 score. We then compared the models based on these average values across the ten trials.

The results of the classification are shown in Table 2. The Balanced NB model had the highest average accuracy of 88.1% across the 10 trials. The table also shows that all of the classifiers had a higher recall for songs, particularly the Original NB model which had the largest difference in recall between the two classes. This result is significant because it remained consistent even when the train set for the Balanced NB classifier was undersampled, suggesting that the difference in recall between the two classes is not due to class imbalance.

3.2 K-NN

We decided to use the K-NN classification method on our dataset using the *KNeighborsClassifier* algorithm from *sklearn*. We used the Euclidean distance measure only on the continuous features in our dataset and trained the model on the original dataset and on a modified version of the dataset. The modified version did not include highly correlated variables and had variables that were adjusted for skewness and without outliers as described in the previous chapter. For simplicity, we will refer to these two models as the Original K-NN and Treated K-NN

To optimize the performance of our K-NN model, we standardized the observations in the train set and applied the same standardization to the observations in the test set using the mean and variance parameters calculated from the train set. We also undersampled the train set to address class imbalance. To determine the optimal number of neighbors for each train set, we used five-fold cross-validation and tested all values of k between 1 and 30. We then evaluated the accuracy of each model with its optimal number of neighbors on the test set. As with the previous models, we repeated this process nine more times to get a more reliable estimate of the model's prediction capabilities.

The treated K-NN model outperformed the other on every measure, having a higher mean accuracy of more than 2% and a higher precision, recall, and f-1 score for both classes as can be seen in Table 3. The mean optimal number of neighbors over the 10 trials was 7.8 for the original K-NN and 10.1 for the treated K-NN.

We then chose to try a mixed distance measure with the model, for it we took a weighted average of the Jaccard distance and Euclidean distance. The Jaccard distance, which is equal to 1 minus the Jaccard similarity, was used for the categorical features, and the Euclidean distance was used for the numerical features. The weight of each distance measure was determined based on the number of features. We trained this model on the same dataset as the Treated K-NN and for simplicity, we will refer to it as Mixed K-NN. From Table 3 we can see that while this model outperformed both of the previous ones in every measure and with a mean accuracy on the test set of 0.91% it did so only slightly when compared to the Treated K-NN model, so the mixed distance measure only slightly improved the predictive capabilities of the Treated K-NN model.

The mean optimal number of neighbors for the model over the 10 trials was 13.3 the highest among the three models. It's possible to see how all three models have a higher recall for song

Table 2: Classification report for the Naive Bayes model over the 10 repetitions

Model	**Target**	**Precision**	**Recall**	**F-1 score**
Original NB	Song	0.782	0.928	0.894
Accuracy: 0.856	Speech	0.935	0.800	0.862
Treated NB	Song	0.827	0.922	0.872
Accuracy: 0.879	Speech	0.931	0.844	0.885
Balanced NB	Song	0.830	0.922	0.873
Accuracy: 0.881	Speech	0.931	0.847	0.887

Table 3: Classification report for the K-NN model over the 10 repetitions

Model	Target	Precision	Recall	F-1 score
Original K-NN	Song	0.841	0.946	0.890
Accuracy: 0.896	Speech	0.952	0.855	0.901
Treated K-NN	Song	0.847	0.965	0.902
Accuracy: 0.907	Speech	0.969	0.860	0.911
Mixed K-NN	Song	0.850	0.969	0.905
Accuracy: 0.911	Speech	0.972	0.865	0.915

observations, as seen previously in the Naive Bayes classifiers, so we can conclude that this is not an issue relative only to the Naive Bayes classifier.

As can be seen in Figure 17 the Treated K-NN and Mixed K-NN models had a higher train accuracy for every value of neighbors tested than the Original K-NN model and the Mixed K-NN slightly outperformed the Treated K-NN for numbers of neighbors higher than 15 and performed worst for lower numbers.

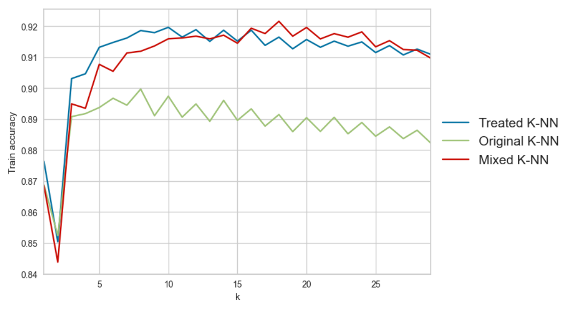


Figure 17: Cross-validation accuracy of the K-NN models for each number of neighbors tested

Table 4: Classification report for the K-NN model over the 10 repetitions

Model	Target	Precision	Recall	F-1 score
Decision Tree	Song	0.827	0.930	0.876
Accuracy: 0.880	speech	0.938	0.844	0.888

3.3 Decision Tree

We implemented a Decision Tree classifier using the *DecisionTreeClassifier* algorithm from *sklearn*. Since this model is robust to outliers and skewness and does not require scaling, we applied it directly to our dataset, only undersampling the train set to address the class imbalance. We then searched for the optimal combination of hyperparameters for our model The hyper-parameters we examined are *criterion* the function used to measure the quality of a split, choosing between Gini or Entropy, *max_ depth* the maximum depth of the tree, choosing in the range of values from 1 to 25, *min_ samples_ split* the minimum number of samples required to split an internal node, choosing in the range of values from 2 to 100 with intervals of 5, *min_ samples_ leaf* the minimum number of samples required to be at a leaf node, choosing in the range of values from 1 to 100 with intervals of 5, *min_ impurity_ decrease*, the minimum impurity decrease to generate a split, choosing values between 0 and 0.7 spaced out by 0.1.

To evaluate the accuracy of different combinations of hyperparameters for our Decision Tree model, we used five-fold cross-validation on the train set with the *GridSearch* algorithm from *sklearn*. We selected the combination with the highest validation accuracy and tested that model on the test set. As before, we repeated this process nine more times, and each time we conducted this process, we found the optimal combination of hyperparameters for that specific train-test split. This allowed us to properly evaluate the performance of the Decision Tree model on each iteration and obtain a more accurate estimate of its prediction capabilities.

The decision tree classifier achieved an average accuracy of 0.88% over the 10 trials done for model assessment, as can be seen from Table 4. As with all the previous models also this one has a higher recall for songs.

The best-performing model out of the 10 trials had Entropy as its splitting criterion, a *max_ depth* of 3, a *min_ impurity_ decrease* of 0, a *min_ samples_ leaf* of 31, and a *min_ samples_ split* of 2. The maximum depth of the Decision Tree model was low, which made the other hyperparameters less important. This is because it was unlikely that the tree would reach a node with fewer than 31 samples or a leaf with fewer than 2 samples after two splits when starting from a train set of more than 1000 observations. This model can be visualized in Figure 18. To visualize our model, we used the *dtreeviz* library. This library allows us to see the train data's splitting process and the distribution of samples in each node based on class for each splitting criterion, along with the chosen threshold.

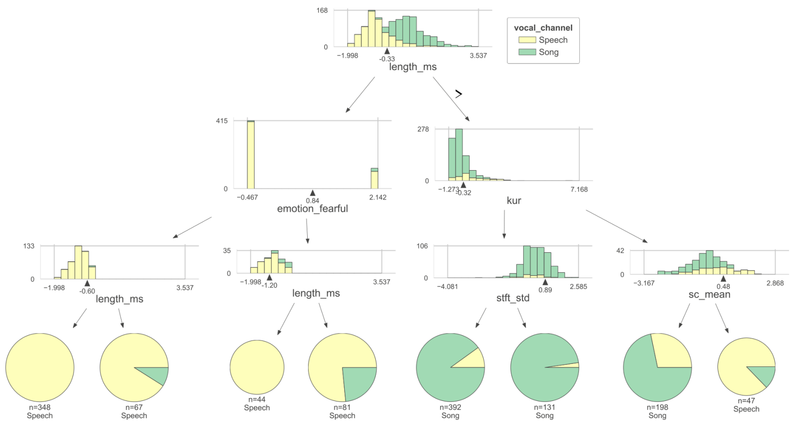


Figure 18: Representation of the training for the decision tree model

Figure 18 shows that the first split of the Decision Tree model was based on the variable *'length_ ms'*, which has a bimodal distribution for songs and speech observations. This split is significant because all observations with a length lower than the threshold are classified as speech due to the majority of speech observations in the four leaf nodes on the left side of the diagram. Only one of the four leaf nodes on the right side has a majority of speech observations. This means that the tree could have just four leaf nodes in total, which would give it a low degree of complexity. The *ìlength_ ms'* feature is important in distinguishing speech observations as the entropy of the node resulting from observations with a length lower than the threshold in the first split was 0.263, a significant decrease compared to the entropy of 1 for the first node. This demonstrates the effectiveness of this feature in separating the two classes.

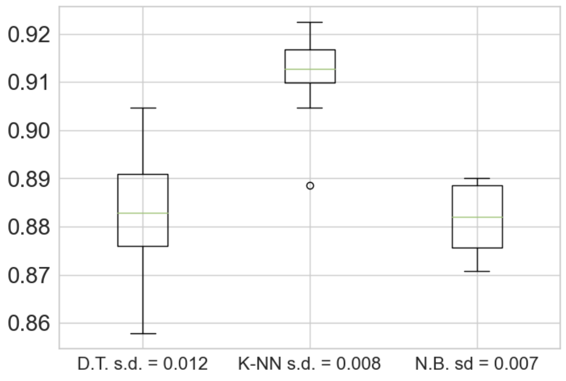


Figure 19: Box plot of the mean accuracy on the test set over the 10 trials and its standard deviation for the best version of each model

3.4 Model comparison

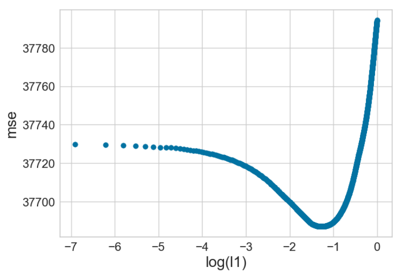
We then proceeded by comparing the accuracy of the Decision Tree classifier with the best-performing K-NN and Naive Bayes models. As we have seen in the previous paragraphs the Mixed K-NN model achieved the highest average accuracy over the 10 trials with a score of 91.1%, while the Decision Tree and Balanced NB achieved two very close average accuracy values of 88% and 88.1%.

From Figure 19 we can see that the Decision Tree model has by far the highest standard deviation in the results, while the Balanced Naive Bayes and Mixed K-NN have similar lower values, this indicates that while the Decision Tree and Naive Bayes models have a similar average accuracy the Naive Bayes Model is more consistent in its results.

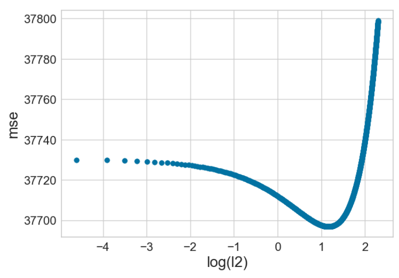
In the end, the Mixed K-NN model outperformed the other models in accuracy, recall precision, and F-1 score being, in our opinion, the best model for this classification task and so we chose it for the imputation of the missing values for the *'vocal_ channel'* feature. From our results, it's also possible to see that all classifiers had a higher recall for Songs.

4 Regression

For our regression task, we chose as target the *'sc_ min'* variable since we wanted to use the best model to impute its missing values. For this task we pre-processed our data with the same method we used for the classification task, eliminating highly correlated features, correcting for skewness, scaling our dataset, and eliminating interquartile outliers from the train set. We chose to compare the performance of the *LinearRegression, Lasso* and *Ridge* models from the *sklearn* library trying to find the optimal regularization parameter.



(a) Lasso



(b) Ridge

Figure 20: Corss-validated MSE of each L1 and L2 parameter tired for the Lasso and Ridge penalisation

We performed on the train set a five-fold Cross Validation testing, 1000 equally spaced possible candidates in the range from 10 to 0.01 for the L2 (Ridge) penalty parameter and in the range from 1 to 0.001 for the L1 (Lasso) penalty parameter. From Figure 20 we can see the cross-validated MSE over the different values of the penalization parameters, from this we can see that the shape of the validation curve is very similar for both models but their optimal regularization parameters strongly differ in magnitude being 3.21 for the Ridge model and 0.276 for the Lasso.

The three models achieved similar results on the validation set, with an MSE of 37686, 37696, and 37730 for the Lasso Ridge and Linear models. While from Table 5 we can see that the Ridge model significantly outperformed the other three on the test set achieving a higher R2̂ score and a lower MSE and Mean absolute error. So we chose this model with 3.21 as the regularization parameter for the imputation of the missing values of the *sc_ min* feature.

Model	**MSE**	**MAE**	**R^2**
Least Squares	33963.53	141.77	0.388
Ridge	33687.52	140.91	0.392
Lasso	33745.62	141.00	0.391

Table 5: Mean squared error, Mean absolute error and R2̂ of each model over the test set

5 Pattern and AR Mining

In this section, we elaborate and discuss the *Pattern Mining* and *Association analysis* techniques we used to discover interesting relationships hidden inside our dataset.

We started by discretizing all the continuous features in our original dataset without missing values and without the highly correlated variables as they would have always come up in the same patterns.

We chose to divide the continuous variables into 4 bins each containing an equal number of observations, to ensure a conservative fragmentation, as a higher number would have significantly reduced the number of frequent patterns. To be precise, the *'stft_ min'* variable had two bins containing observations within the same range of values, as it had 1016 observations with the value 0, so we decided to unite them in a single bin (resulting in a 3-bin variable). We renamed the bins for each feature Low, Medium, High, and very high, while for *'stft_ min'* we named them Low, Medium, and High.

5.1 Frequent patterns

To find the frequent patterns in our data, we applied the *apriori* method from the *pyfim* library, we iterated this algorithm giving it as parameter minimum support values ranging from 2% to 40%.

We found the ('F', 'Low *stft_ min*') set to be the most frequent one having a support value of 38%, this is understandable as the 'Low *stft_ min*' and 'F' items are amongst the most common in the dataset and female observations have lower values of this variable than males (Paragraph 1.2.4), while the most frequent itemsets composed of 3 items are: ('Low stft_std', 'M', 'speech') and ('VeryHigh stft_std', 'F', 'Low stft_min'), with support of 21% each, this finding is interesting as it reiterates the fact that female and male observations vary the most over the *'stft_ std'* variable as said in paragraph 1.2.4.

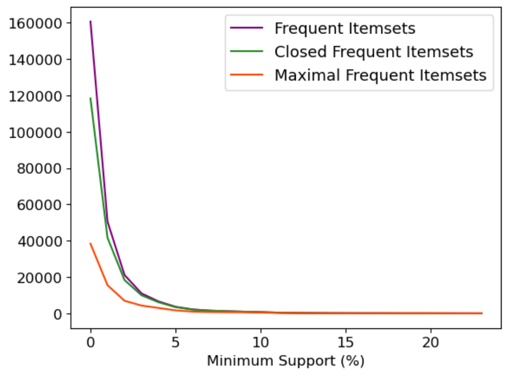


Figure 21: Variation of the number of *Frequent*, *Closed frequent* and *Maximal frequent* itemsets compared to different thresholds of *minimum support.*

Observing Figure 21, besides the fact that we can confirm the expected convergence of the number of frequent itemsets with the closed and maximal ones with the increase of the minimum support value, we noticed that from the support threshold of 9% onwards, the ratio of the number of closed itemsets to the number of frequent itemsets is more than 97%, so nearly all frequent itemsets are also closed and from the 16% cutoff onwards they are all closed (133 itemsets). The results also show that for the specific threshold of 22%, we find that there are 56 itemsets that are also maximal frequent.

These results led us to choose, for the subsequent evaluations, a minimum support cutoff of 16%. At this threshold, while the pairs represent 87.2% of the total frequent itemsets (which are also closed), the frequent triplets are only 12.8% (but all also maximal). We Then chose to focus our analysis on the itemsets that contained the *'vocal_ channel'* feature as we want to use them to generate association rules for a classification task.

In Table 6 we chose to showcase the 5 most interesting itemsets we found, from it we can once again see how the item *'song'* is correlated with high values of *'length_ ms'* while shorter audio files are frequently associated with the *'speech'* class, and also with the *'normal'* emotional intensity.

Table 6: Sample of 5 frequent itemsets and relative support

Itemset	**Support**
('VeryHigh length_ms', 'song')	22.853%
('Low length_ms', 'speech')	26.247%
('Low std', 'speech')	21.709%
('Low length_ms', 'normal', 'speech')	18.438%
('VeryHigh std', 'strong')	20.155%

From the third and fourth pair in Table 6 we see that *'speech'*es are also weaker in terms of intensity (in Decibels), while audio files with high intensity are also associated with a *'strong'* emotional intensity.

5.2 Association rules

In order to find the most interesting association rules and to minimize the noise, we varied the confidence(from 50% to 90%) and support (from 5% to 20%) parameters.

In order to better see the behavior of these changes we plotted in Figure 22 the number of rules resulting from each combination of the *confidence* and *support* parameters. We chose to consider rules with at least 70% confidence and 16% support as this seemed to us a good compromise between quantity and quality of rules.

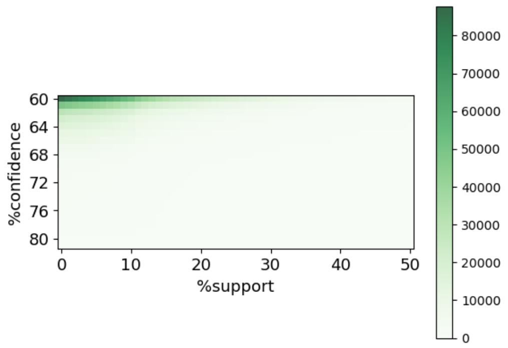


Figure 22: Variation of the number of rules relative to the confidence and support parameters

The last parameter needed to compute the algorithm is the minimum number of elements *z-min* in each rule. We chose to set this parameter to 3, which permitted us to find 84 rules, as a value of 4 would have given us only 9.

From Table 7 we can see the 5 rules we found with the highest lift values. The first 3 rules all have the item *Low_ stft_ std* as Consequent, we can interpret these rules following what previously said in paragraph 1.2.4 since male and speech observations have lower values of *'stft_ std'*, and since it's negatively correlated -0.65 and -0.58 with the *'stft_ min'* and *'stft_ kur'* variables. The other two rules with the highest lift have a Consequent the *'Low_ mfcc_ max'* item, the rules can be explained by the fact that as said in paragraph 1.2.4 female and song observations have lower values of the *'mfcc_ max'* feature, which is also positively correlated 0.38 with the *'stft_ min'* variable.

Consequent	**Antecedent**	**Support**	**Confidence**	**Lift**
Low_stft_std	(High_stft_min, M, speech)	14.595	0.908	3.630
Low_stft_std	(High_stft_min, speech)	14.840	0.894	3.573
Low_stft_std	(VeryHigh_stft_kur, speech)	13.736	0.837	3.348
Low_mfcc_max	(song, F)	16.067	0.813	3.251
Low_mfcc_max	(song, F, Low_stft_min)	14.145	0.808	3.231

Table 7: Association rules with the highest lift with *z-min* $= 3$, *conf* $= 70$, *supp* $= 16$

5.3 Classification with association rules

For the task of classification using association chose to focus our analysis on the rules which had as a consequent the variable *vocal_ channel* as it was the target for our classification task, in order to compare the results of the two techniques. To do so we divided our dataset putting 20% of the observations in the test set, we then searched for all the rules that had song or speech as a consequent and with minimum *z-min, confidence* and *support* of 3, 70 and 16. we applied all of the discovered rules to the test set, classifying each observation as the Consequent when the rule could be applied, and as the other class when it couldn't. We then calculated the accuracy of each rule on the test set.

Consequent	**Antecedent**	**Accuracy**	**Support**	**Confidence**	**Lift**
Song	(Low_kur, normal)	75.1%	15.184	0.939	2.222
Song	(VeryHigh_length_ms, Low_stft_min)	71.8%	15.541	0.962	2.275
Song	(Low_sc_mean, VeryHigh_sc_kur)	67.7%	17.229	0.758	1.794
Song	(Low_mfcc_max, F)	67.5%	17.229	0.723	1.710

Table 8: Song association rules with the highest accuracy with *z-min* = 3, *conf* = 70, *supp* = 16

Consequent	**Antecedent**	**Accuracy**	**Support**	**Confidence**	**Lift**
Speech	(Low_stft_std, M)	58.1%	20.449	0.923	1.600
Speech	(Low_length_ms, normal)	57.1%	18.098	0.997	1.727
Speech	(Low_stft_skew, M)	56.9%	18.967	0.973	1.687
Speech	(Low_std, normal)	53.0%	15.388	0.862	1.494

Table 9: Speech association rules with the highest accuracy with *z-min* = 3, *conf* = 70, *supp* = 16

From Table 8 and Table 9, we see that the rules regarding songs greatly outperformed the ones regarding speech, with the highest accuracy of 75.1% achieved by the *Low_ kur, normal* → *song* association rule. The speech association rules in contrast only achieved an accuracy of 58.1% with the *Low_ stft_ std, M* → *speech.* It's also interesting to see that the most accurate association rule used the *'kur'* feature which was also employed by the most successful Decision Tree model. But all association rule couldn't reach the accuracy of any of the classifiers built in the previous chapter.

From the confusion matrix in Table 10, we can see that the most successful rule had an accuracy of 75.1%. This rule only predicted 15% of observations as songs and the rest as speech, resulting in an accuracy of 61% for correctly predicting almost all speech observations and an additional 14% for correctly predicting 37% of song observations. This is because even the most precise rules are only applied to a small percentage of the population, as most observations do not meet the requirements of the rule. Therefore, rules that apply to the minority class (songs in this case) can achieve higher accuracy by correctly predicting the majority class by default.

		Predicted	
		Song	Speech
Actual	Song	68	115
	Speech	7	300

Table 10: Confusion matrix over the Test set for the best performing Association Rule

We can also see that the *length_ ms* variable is found again within song and speech association rules, which confirms once again the relation found between it and the *vocal_ channel* feature. We confirm also the results found in the previous paragraph regarding the relation between *mfcc_ max* and songs and *Low_ stft_ std* and speech.

It's also interesting to see how the emotion feature didn't appear once in the most relevant association rules for either *'song'* or *'speech'* observations and also in none of the 5 most interesting rules, as each emotion isn't represented enough in the data to surpass the support threshold. Also, the fact that rules regarding *'songs'* have a higher lift and accuracy than the ones regarding *'speech'* could explain why all classifiers had a higher recall when predicting Song observations with respect to speech.