Data Mining

Advanced Topics and Applications

EXAMINATION PROJECT REPORT

Can a song move more than a thousand words?

Exploring the relevance of vocal channel in classifying emotions

Lucrezia Labardi (600163)[1]
Vincenzo Sammartino (599203)[1]
Daniele Borghesi (578406)[2]

[1]*Master's degree in Digital Humanities, Università di Pisa*
[2]*Master's degree in Data Science and Business Informatics, Università di Pisa*

ACADEMIC YEAR 2022-2023

Contents

1 Introduction

Every human can recognize emotions based on several factors. In this project, it was decided to study the features that enable major algorithms to investigate and distinguish emotions like *anger*, *surprise*, *fear*, *happiness*, *disgust*, *neutrality*, *sadness* and *calm*, taking into consideration both spoken and sung sentences. The reference dataset was RAVDESS[1] in its audio-only part.

First of all, in section 2 several operations of data preparation and understanding were performed, including attribute transformation and elimination, succeeded by a subdivision of the data useful for later experiments. Next, in section 3, an outlier detest phase was carried out, where dimensionality reduction techniques were also exploited. Then, in Section 4, it was tested an unbalanced class scenario, where the dataset was artificially unbalanced. It was decided to test two algorithms for oversampling and one for undersampling, then a classification task was performed to show and compare the results. Respectively, in Section 5 and 6, several algorithms of multi-class classification and multiple feature regression were performed. For the classification task, it was decided to classify the eight emotions on three different datasets: the full dataset, a dataset with only speech data, and a dataset with only song data.

This data subdivision allows us to pursue the main task of the project: understanding whether songs can lead a machine-learning model to detect emotions better than speech, or vice versa.

2 Data Understanding and Preparation

This section explains all the main operations for exploring and preparing the dataset for subsequent experiments. Specifically, in subsection 2.1 a general overview of the dataset is given, with a focus on the breakdown of the data among the various categorical features present. Next, in subsection 2.2, the operations performed to reduce the number of attributes, including highly correlated features and irrelevant attributes, are explained. In subsection 2.3, it is illustrated how the data were transformed to improve subsequent machine learning model training. Finally, subsection 2.4 illustrates how the data were partitioned for conducting the subsequent advanced classification and regression experiments.

2.1 Dataset overview

First, it was verified that there were no duplicate records in the dataset, nor missing values in any feature. Thus, there was no need to perform record removal operations or filling of missing values.

The dataset consists of two parts: a *training-set*, of 1828 items, and a *test-set* of 624 items. The main features of the *training-set* are discussed below.

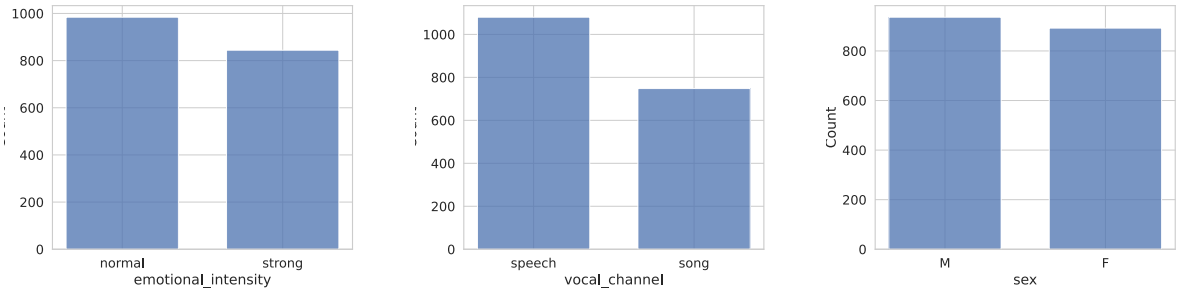

Figure 1: Data distributions for main categorical features

As can be seen from figure 1, the available data are not perfectly balanced across classes. By analyzing the binary categorical features, we can see that:

- Records related to strong emotional intensity are fewer in number than those related to normal emotional intensity (feature *emotional_intensity*)
- Records related to sung sentences are considerably fewer than those related to spoken phrases (feature *vocal_channel*)

[1] Livingstone SR, Russo FA (2018) The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. PLoS ONE 13(5): e0196391. https://doi.org/10.1371/journal.pone.0196391.

- Phrases spoken by male actors are slightly more numerous than those spoken by female actors (feature *sex*)

Figure 2: Data distribution for emotions

Figure 2 shows differences in the distribution of the data in the training set also with regard to the emotions (feature *emotion*): the records for *disgust*, neutrality (*neutral*) and surprise (*surprised*) are half as large as those for the other emotions.

2.2 Correlations and irrelevant attributes

As the first step of the Data Preparation process, all unnecessary attributes were removed, to greatly reduce the size of the dataset.

Starting with the irrelevant attributes, features that had the same value for each record were eliminated. In this way, the number of attributes was reduced from the initial 434 to 382: the dataset had 52 irrelevant features with the same value for each instance. Among the categorical features removed was the feature "modality", which had the value "audio-only" in all instances.

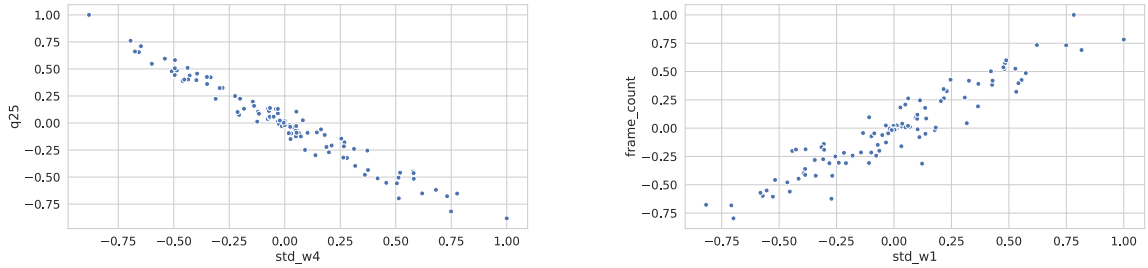

Figure 3: Two examples of highly correlated feature pairs

To further reduce the size of the dataset, correlations between features were considered: the Pearson correlations of the 382 remaining features were calculated and it was noted that many of them had a very strong positive or negative correlation (see figure 3). Therefore, one feature was eliminated for each pair that correlated less than -0.75 or greater than 0.75. This operation eliminated a large number of attributes, reducing the size of the dataset from 382 to 115 features. Highly correlated features provide machine learning models with very similar information, and that can be redundant and counterproductive for classification and regression operations.

At the end of the operations, the dataset was reduced by almost 75% of its dimensionality.

2.3 Attribute transformation

The values of the features in a dataset may have very different characteristics. They may coexist features with very skewed distributions, or with very different ranges of values between them. These characteristics may be obstacles to the training of machine learning models.

It was decided to carry out two types of transformations for each feature:

1. Data were transformed using a *Cube Root transformation*, to smooth out any highly skewed distributions. Cube Root transformation was chosen because several features have both negative values

and values equal to zero, for which a *logarithmic transformation*, for example, would not have been possible

2. Data were scaled using a *Standard Scaler*, in order to have a more normal distribution (e.g., Gaussian with 0 mean and unit variance)

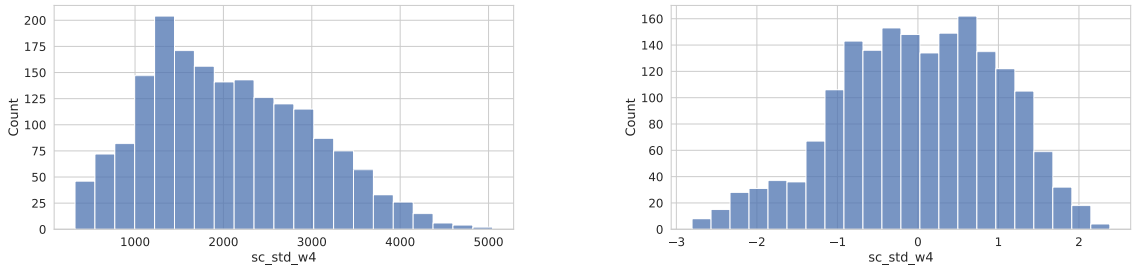

Figure 4: An example of the effects of data transformation on a feature distribution (before the transformations, on the left, and after the transformation, on the right)

As shown in figure 4, at the end of the operations, the data distribution of each feature is made more like a normal one, and its symmetry is improved.

2.4 Subdivision of data

As mentioned, one of the goals of this project is to analyze whether the efficiency of a classifier can vary based on the type of utterance. For this reason, classification experiments were performed using three datasets: the full dataset, a dataset with only speeches, and a dataset with only songs. In this way, it was possible to analyze whether there are differences in emotion classification among these three datasets, and thus whether songs are better able to convey emotion than a speech, or vice versa.

3 Outlier Detection and Dimensionality Reduction

This section discusses the detection and handling of outliers. Specifically, subsection 3.1 illustrates the results obtained by running four different types of algorithms for detecting outliers, while subsection 3.2 illustrates how outliers were eliminated or replaced for preparing the dataset for training the machine learning models.

3.1 Top 1% outliers

Four different outlier detection algorithms were chosen, belonging to four different families. Specifically, they were tested:

- *ABOD*, an *high-dimensional* approach
- *Isolation Forest*, an *ensemble-based* approach
- *Likelihood*, a *statistical* approach
- The *Deviation-based* approach

Other types of approaches, such as *distance-based* or *density-based* algorithms were discarded, as distance and density become significantly less meaningful when used in high-dimensional datasets. At the same time, *depth-based* approaches such as *Elliptic Envelope* were also discarded, as the number of samples in the dataset used is not sufficiently large compared to the number of attributes (the number of samples should be larger than the number of attributes at the second so that the Elliptic Envelope algorithm provides reliable results)[2].

Each algorithm provides an outlier score for each point in the dataset. Accordingly, the 1% of outliers with the highest score was extracted for each algorithm used.

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.covariance.EllipticEnvelope.html`

Table 1: Top 1% outliers detected by tested algorithms

ABOD		*Deviation*		*Likelihood*		*Isolation Forest*	
Outlier	*Score*	*Outlier*	*Score*	*Outlier*	*Score*	*Outlier*	*Score*
942	-4.16e-12	942	1.07	942	1.27e+20	1704	0.56
382	-4.45e-08	382	0.90	382	4.65e+02	1702	0.55
1702	-1.08e-07	589	0.73	589	3.20e+02	656	0.55
1765	-1.46e-07	1704	0.37	1704	1.22e+02	1756	0.54
292	-1.51e-07	1367	0.36	311	1.00e+02	1703	0.54
251	-1.64e-07	314	0.36	308	9.90e+01	1707	0.54
826	-1.71e-07	315	0.35	1702	9.79e+01	308	0.54
294	-1.72e-07	324	0.35	1367	9.61e+01	292	0.53
1752	-1.73e-07	1597	0.35	324	9.59e+01	1367	0.53
771	-1.78e-07	840	0.31	314	9.52e+01	1481	0.53
458	-1.86e-07	1702	0.30	1597	9.49e+01	314	0.53
832	-1.88e-07	771	0.30	840	9.48e+01	772	0.53

In table 1, it can be seen that the four algorithms tested show similar results in identifying the top 1% outliers. Some data points, such as 942, 382, 1702, 1704, 589, and 314 recur in almost all rankings. Data points 942 and 382 turn out to be the two most outlier points for three out of four algorithms, while point 589 turns out to be the third most outlier point for both the *Deviation-based* and *Likelihood* algorithm. Upon initial observation, the *Isolation Forest* algorithm appears to show the greatest differences in identifying the top outliers compared to the other algorithms.

A graphical visualization can help to better compare the results of the tested algorithms. In this regard, Figure 5 also shows how the top 1% outliers detected by *ABOD* differ from what was identified by the other algorithms, which have very similar representations.

Overall, the results suggest that the different outlier detection models may have different strengths and weaknesses in identifying outliers. Therefore, it is important to carefully consider the characteristics of the data and the goals of the analysis when choosing an appropriate outlier detection method. In this regard, the similarity (overlap) between the top 10% outliers detected by each algorithm was analyzed. This was compared with the top 10% outliers detected by an algorithm that tended to be unsuitable for our dataset: *Elliptic Envelope*. In this way, it is possible to see which algorithm, among the four tested, differs most in its results from *Elliptic Envelope*.

Table 2: Overlap of the top 10% outliers detected by the tested algorithms

Algorithm	*Iso. Forest*	*Deviation*	*Likelihood*	*ABOD*	*Elliptic Envelope*
Iso. Forest	/	75%	65%	49%	68%
Deviation	75%	/	87%	53%	74%
Likelihood	65%	87%	/	47%	69%
ABOD	49%	53%	47%	/	52%

From table 2, it is possible to see that the *ABOD* algorithm differs the most from the results obtained by *Elliptic Envelope*. At the same time, it is also possible to see that there are very strong similarities (87%) between the top 10% outliers detected by the Deviation-based algorithm and by *Likelihood* and *Isolation Forest*. In contrast, *ABOD* shows no more than 53% overlap with the results from the other algorithms. From these results, we can infer how *ABOD* seems to establish itself as the most reliable algorithm when compared with the results of an inadequate algorithm such as *Elliptic Envelope*.

3.2 Outliers removal

Given the results obtained, it was decided to handle outliers in two stages: in the first stage, all the outliers detected by the ABOD algorithm (which was shown to be the most reliable) were removed. In the second stage, a substitution of outliers detected by interquartile range methodology was performed for each feature: on each attribute, items above the 75 percentile, and below the 25 percentile were detected and substituted with the median of the feature. The median was calculated by grouping data

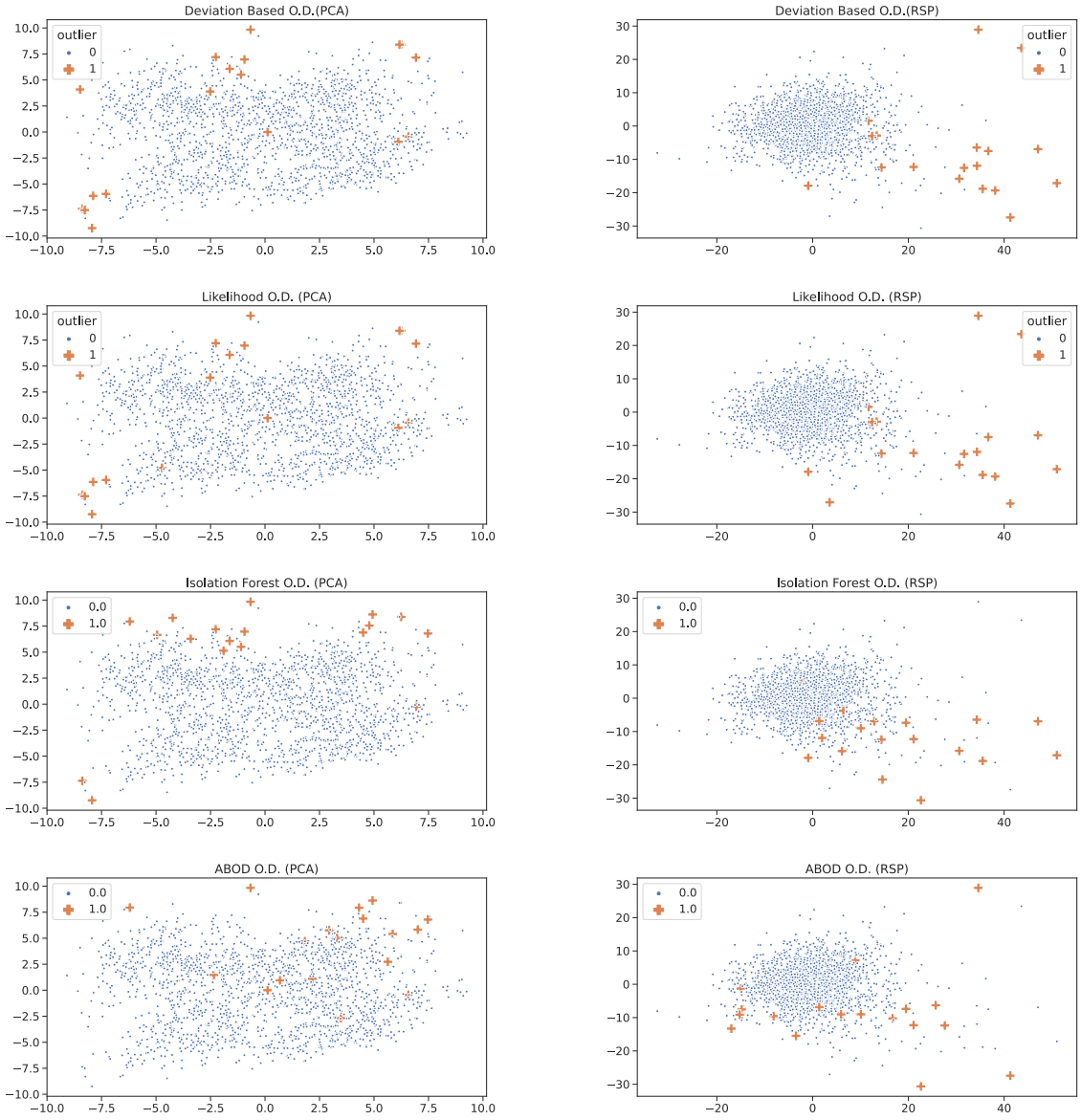

Figure 5: Principal Component Analysis (PCA) and Random Subspace Projection (RSP) representations of the complete Dataset. Top 1% outliers are highlighted in orange

for each numerical attribute based on the categorical attributes seen in section 2.1, excluding *vocal_channel* (because it will be used for classification in section 4) and *emotion* (because it will be used for classification in section 5).

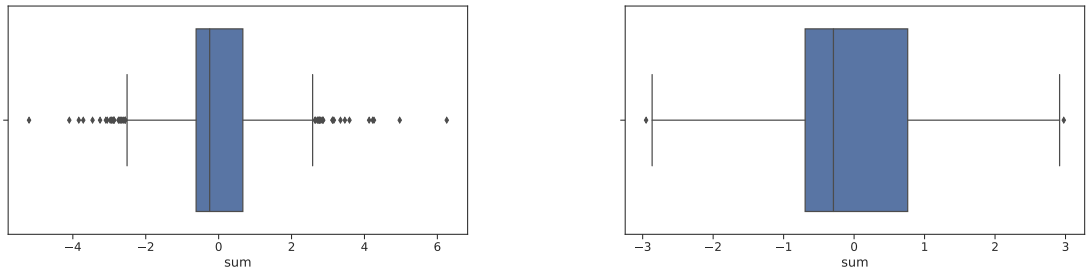

Figure 6: Attribute *sum* before (on the left) and after (on the right) outlier removal and substitution

Figure 6 shows an example of how the two operations illustrated above succeed in effectively removing

outliers from the dataset. Specifically, the removal of outliers detected with *ABOD* resulted in the elimination of 50 records from the *training-set*, and 22 records from the *test-set*, bringing them respectively to 1778 and 602 remaining items.

4 Imbalanced Learning

This section discusses testing in an unbalanced class scenario. Specifically, subsection 4.1 illustrates how the dataset classes were artificially unbalanced, while subsection 4.2 deals with the tested undersampling and oversampling algorithms used to perform classification on a rebalanced dataset.

4.1 Dataset unbalancing and rebalancing

To implement Imbalanced Learning operations, it was decided to make the original dataset artificially imbalanced. To create imbalance, one of the available categorical variables was chosen. According to what was observed in section 2.1, the feature that had the greatest natural imbalance was *vocal_channel*. Therefore, the classification task, along with the artificial unbalancing of the dataset, was implemented on that feature to keep the dataset size as large as possible.

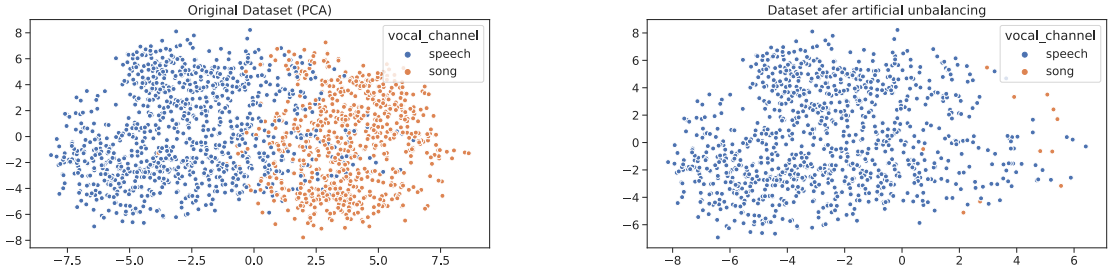

Figure 7: Dataset before and after artificial unbalancing (PCA visualization)

As shown in figure 7, the dataset has been unbalanced significantly: to be precise, after unbalancing, the *speech* class of *vocal_channel* constitutes about 99% of the dataset, while the *song* class constitutes the remaining 1%. In the end, the training dataset passed from a total of 1778 to 1057 records.

After unbalancing the dataset artificially, it was decided to test several techniques for managing imbalanced learning. Specifically, it was decided to test the *ADASYN* oversampling algorithm, and the *CNN* (*Condensed Nearest Neighbor*) undersampling algorithm.

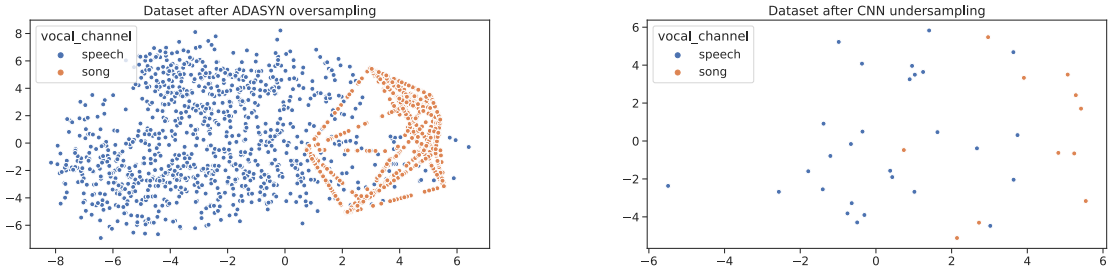

Figure 8: Unbalanced Dataset after ADASYN oversampling and CNN undersampling (PCA visualization)

Figure 8 shows the effect of the two chosen algorithms on the unbalanced dataset. As can be seen, in the case of *ADASYN*, oversampling resulted in a significant increase in the number of records related to the minority class (song), increasing the total number from 1057 to 2095. In the case of undersampling through *CNN* the width of the speech class was drastically reduced, dropping the total records of the dataset from 1057 to 39.

4.2 Classification results

After rebalancing the dataset using the *ADASYN* and *CNN* algorithms, classification tests were performed using the *K Nearest Neighbors* (*KNN*) algorithm. For the algorithm, a tuning of the hyperparameters was performed on the unbalanced dataset, setting the number of neighbors (`n_neighbors`) and the metric (`metric`) to use for distance computation (*Euclidean* or *Cityblock*). Tuning was performed using a randomized search with cross-validation in 20 splits and 3 repetitions.

The best results have been obtained using a number of neighbors equal to 1, and the *Euclidean* metric. With this configuration the tests on the test set were performed, using both the unbalanced and rebalanced dataset as training ones.

Table 3: Classification results for imbalanced learning

Dataset	*Accuracy*	*F1-score*	*Precision*	*Recall*	*ROC-AUC*
Imbalanced	81%	79%	87%	78%	78%
After CNN	89%	88%	91%	87%	87%
After ADASYN	**95%**	**95%**	**95%**	**95%**	**95%**

As table 3 shows, the differences between the scores obtained using the unbalanced dataset and the rebalanced datasets are evident. The difference in Accuracy obtained by training the model with the unbalanced dataset differs by a minimum of 8 and a maximum of 14 percentage points from the Accuracy obtained by training the model with the rebalanced dataset. In particular, using an oversampling (*ADASYN*) allows us to obtain much better results than using an undersampling technique such as *CNN*.

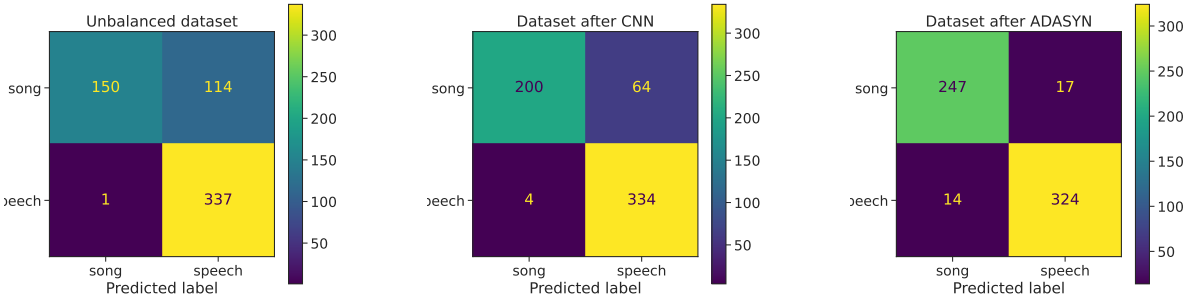

Figure 9: Confusion matrix of the classification with (from the left) the imbalanced dataset, the dataset after CNN and after ADASYN

As figure 9 shows, although undersampling via CNN significantly improves classification scores, the model tends to classify a large number of songs as speech (64). On the contrary, the classification confusion matrix obtained with the ADASYN dataset shows better results: only 17 songs were classified as speech, over six times better than the imbalanced dataset.

5 Advanced Classification

This section illustrates the operations of the classification of emotions through the use of different models. In particular, subsection 5.1 illustrates the behavior of the models as the hyperparameters vary. Subsection 5.3, on the other hand, shows the results obtained by exploiting bagging and boosting techniques with the various models previously tested. Finally, subsection 5.2 shows the results obtained by all models on the test set.

Contrary to what was seen in section 4 (Imbalanced Learning), in these experiments the dataset was kept in its original form, with the sole application of data preparation and outliers removal operations. Therefore, no emotion undersampling or oversampling techniques were used.

5.1 Behavior analysis of the main models

This subsection explains the hyperparameter settings used for each classification model and analyzes the behaviors of the various models as the hyperparameters change on the complete dataset (containing both

speech and song data).

Five main classification models were tested: *Linear Support Vector Machine* (Linear-SVM), *Nonlinear Support Vector Machine* (Nonlinear-SVM), *Random Forest* (Random-Forest), *Multilayer Perceptron* (MLP), and *Logistic Regression* (Logit).

Hyperparameter tuning was performed using *Cross-Validation*. Based on the time taken by each model to perform an iteration, different *Cross-Validation* configurations were used (repeated, non-repeated, with 20 splits, with 10 splits...). A *Randomized-Search* was used for each tuning, to further optimize calculation times.

5.1.1 Linear Support Vector Machine

For Linear-SVC, two main hyper-parameters were tested and optimized:

- The regularization parameter C (`C`), among values 0.001, 0.01, 0.1, 1, 10
- The optimization problem type (`dual`): *dual* or *nondual*

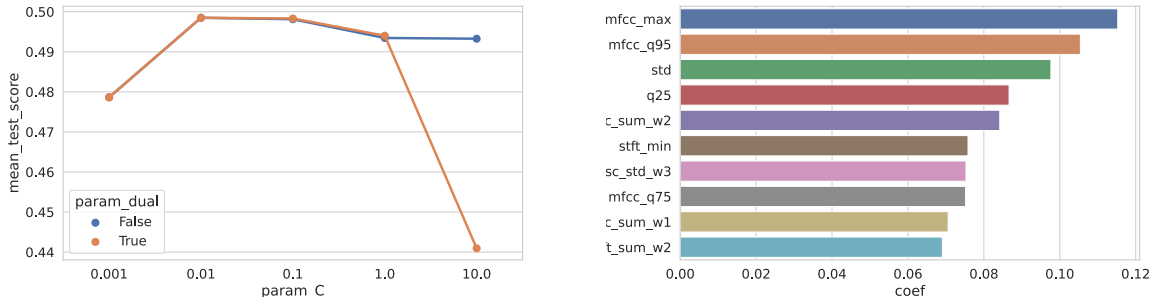

Figure 10: Differences in performances between different values of parameter C in Linear-SVM (on the left), and the 10 most important features (on the right)

As can be seen from figure 10, the performance of Linear-SVM varies significantly as the regularization parameter C changes, leading to the best results with a value of 0.01. There are no particular differences between using the dual or nondual optimization algorithm, but the first of the two showed better performance. The only exception of records with parameter C equal to 10, where using the dual algorithm leads to a drastic drop in performance.

It can also be seen in the figure that the 10 most relevant features for emotion classification include data on Mel-Frequency Cepstral Coefficients (MFCC), which are particularly used in speech recognition. We also note how the standard deviation (std) among the recorded acoustic data is particularly important in distinguishing emotions: different types of emotions might have a wider or narrower range of values.

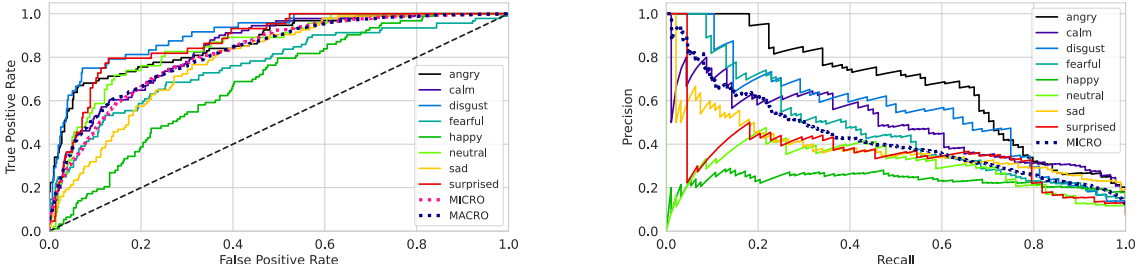

Figure 11: ROC curve (on the left) and Precision-Recall curve (on the right) of Linear-SVC on the complete dataset

As can be seen from figure 11, the classification accuracy is not uniform for all emotions. In particular, *happiness* seems to be the most complex emotion to classify for Linear-SVC, followed by *surprise*, *sadness*, and *neutrality*. We can hypothesize how *happiness* and *surprise* can be easily confused by a machine learning model, as can *neutrality* and *sadness*. In contrast, expressions such as *anger* and *disgust* prove significantly easier to classify. On the other hand, the other emotions stand at average values in both curves.

5.1.2 Nonlinear Support Vector Machine

For the Nonlinear-SVC, the optimized parameters were:

- The Kernel type (`kernel`) to be used in the algorithm : *poly*, *rbf* or *sigmoid*
- The Kernel coefficient (`gamma`) for *rbf*, *poly* and *sigmoid*: *auto* or *scale*
- The Regularization parameter C (`C`): 0.001, 0.01, 0.1, 1 or 10

Specifically, the best Kernel type was found to be *rbf*, while the best type of Kernel coefficient succeeded to be *auto*. At the same time, the value of regularization parameter C that led to the best results was 10.

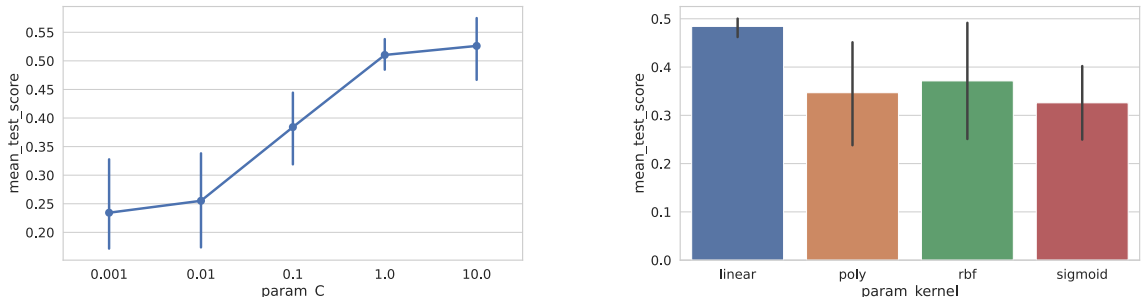

Figure 12: Differences in performances between different values of regularization parameter C (on the left) and different Kernel types (on the right) in Nonlinear-SVC

As can be seen in figure 12, the choice of the right hyper-parameters plays a key role in achieving good classification performance. Contrary to what was seen with Linear-SVC, a higher value of the regularization parameter C allows us to obtain a significant average increase in performance. On average, the *linear* kernel (already seen with Linear-SVC) led to better average results than the other Kernel types, but the *rbf* kernel still managed to outperform Linear-SVC in combination with the regularization parameter C equal to 10.

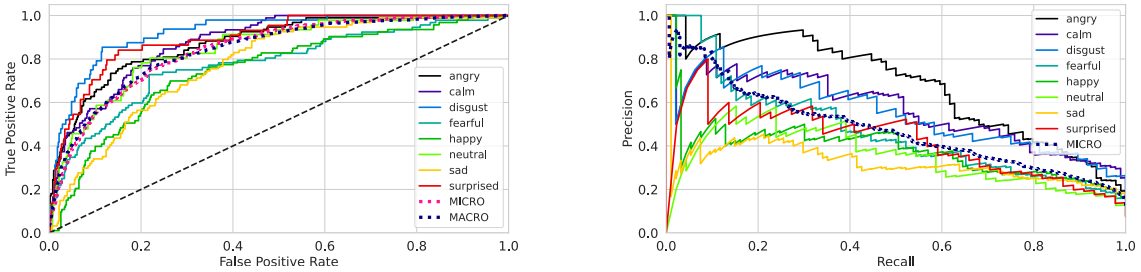

Figure 13: ROC curve (on the left) and Precision-Recall curve (on the right) of Non-Linear SVM for the complete dataset

As can be seen from Figure 13, there are differences in the classification of emotions similar to those observed for Linear-SVC: emotions such as *sadness* and *happiness* are classified worse than other emotions, particularly compared to *disgust* and *anger*.

Since Nonlinear-SVC was shown to be the best-performing classifier in the battery of main models, it was decided to consider the confusion matrix of the emotion classification, to better analyze the differences in the curves in figure 13.

From figure 14 it is possible to see how neutrality is confused with many other emotions, unexpectedly with anger, but also with sadness (as hypothesized with the Linear-SVC). At the same time, disgust is also often confused with a different emotion: calm. It is possible to hypothesize that disgust is a very variable emotion, which can be expressed in many different ways.

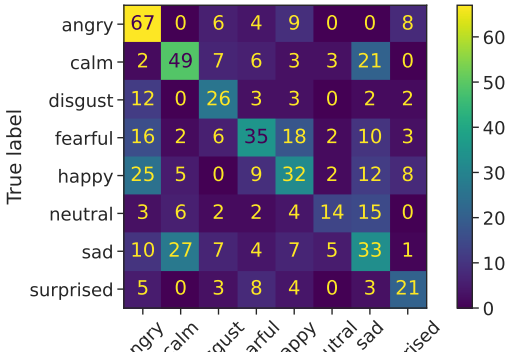

Figure 14: Confusion Matrix for emotion classification of Nonlinear-SVC

5.1.3 Logistic Regression

As already done for Linear-SVC, also for Linear Regression it was tested the regularization parameter C, and the optimization problem type.

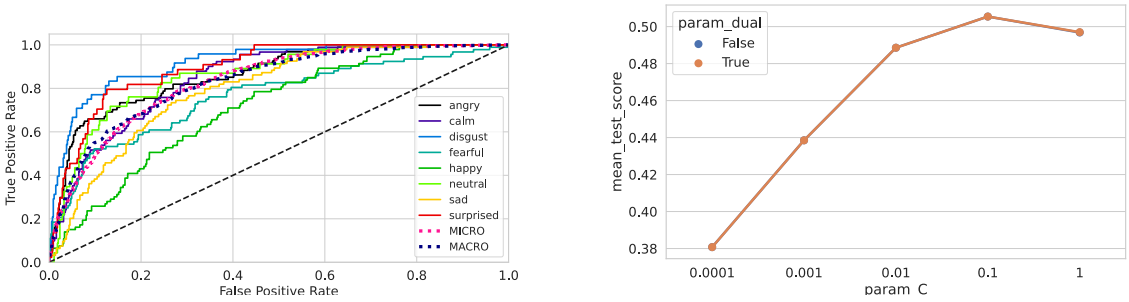

Figure 15: ROC curve (on the left) and differences in performances between different values of parameter C (on the right) of Logistic Regression for the complete dataset

As we can see from figure 15, for C, the best value turned out to be 0.1, and for optimization problem type, the behavior of *dual* and *nondual* is very similar, since they are almost completely overlapping, but True was decided to be the best solution by the randomized search algorithm.

For the Logistic Regression model, it was decided to not show the Precision-Recall curves because the behavior is very similar to Linear-SVC. In the right side of figure 15 we can see the ROC curve and how emotions such as *sadness*, *fear* and *happiness* are the ones that are classified the worst, conversely to *disgust* and *surprise*.

5.1.4 Multilayer Perceptron

For the Multilayer Perceptron, the hyper-parameters that have been optimized are the following:

- size of the hidden layer (`hidden_layer_sizes`), of 32, 2x32, 64, 2x64, or 128 neurons
- learning rate schedule for weight updates (`learning_rate`), of *constant* or *adaptive* type
- activation function for the hidden layer (`activation`), of *logistic*, *tanh*, or *relu* type
- the solver for weight optimization (`solver`): *sgd* or *adam*

Specifically, the most efficient architecture was found to be a single layer of 128 neurons. At the same time, the best activation function turned out to be ReLU, while the best solvers and the best type of learning rate turned out to be Adam and the adaptive type, respectively.

As can be seen from figure 16, as the number of neurons in the hidden layer increases, the performance of the Multilayer Perceptron increases, but not dramatically. Tanh and ReLU activation functions share similar performance for all hidden layer configurations. However, it is interesting to observe how the

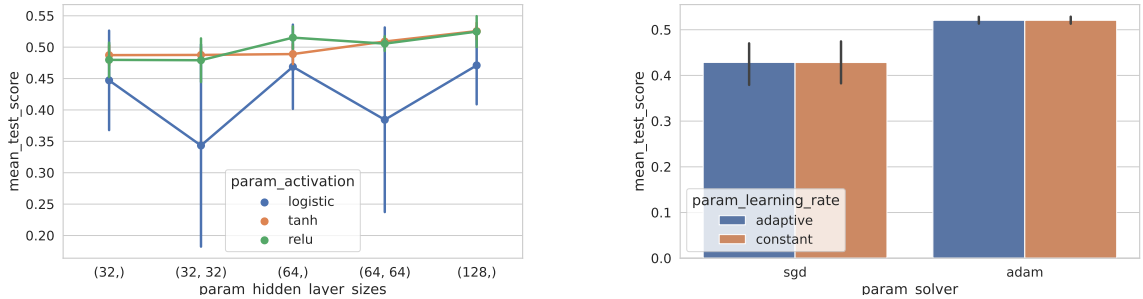

Figure 16: Differences in performances between different values of parameter C and different kernel in Multilayer Perceptron

logistic function leads to a sharp drop in performance in configurations with 2 hidden layers. It is possible to hypothesize how the accuracy of this activation function drops as the number of layers increases. Probably, the problem is also further enhanced during the backpropagation algorithm. Finally, we note how the use of an adaptive or constant learning rate makes no difference in performance in either solver tested.

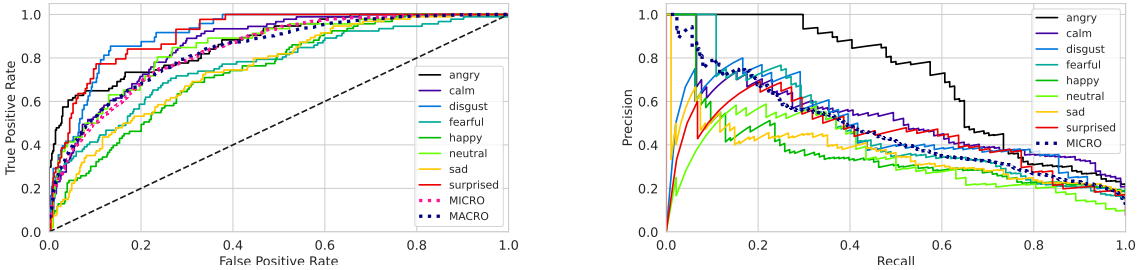

Figure 17: ROC curve (on the left) and Precision-Recall curve (on the right) of Multilayer Perceptron for the complete dataset

In figure 17, it can be seen that there are no notable differences between the various emotions, except for *angry*, which is the most accurate for this model (a characteristic found in many other models). At the same time, emotions such as happiness, fear and sadness are below average in both curves.

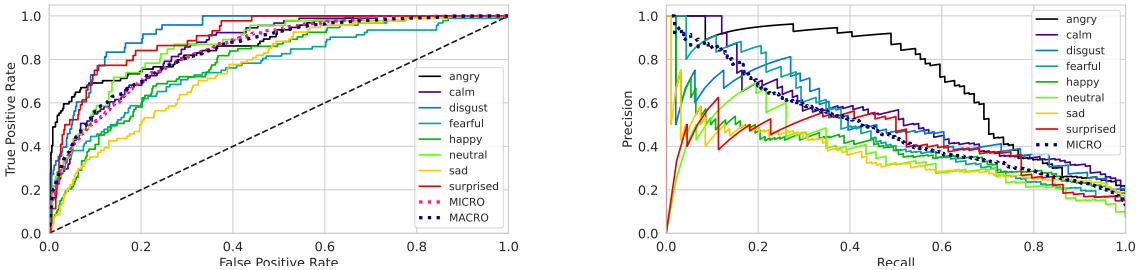

Figure 18: ROC curve (on the left) and Precision-Recall curve (on the right) of Multilayer Perceptron for the complete dataset with Early Stopping

An additional experiment was conducted using the Early Stopping technique to increase the generalization ability of the model, limiting overfitting. Compared with the accuracy obtained with the classical model (45%), using Early Stopping allowed a slight increase to 46% Accuracy (see table 4), establishing itself as the best training configuration.

In any case, the curves in figure 18 show that there is no particular difference in the classification performance of individual emotions compared to the methodology without early stopping.

5.1.5 Random Forest

For Random Forest, several numbers of estimators were tested, such as 10, 100, and 1000 decision trees, and three different impurity measures were compared: Gini, Entropy, and Log-Loss.

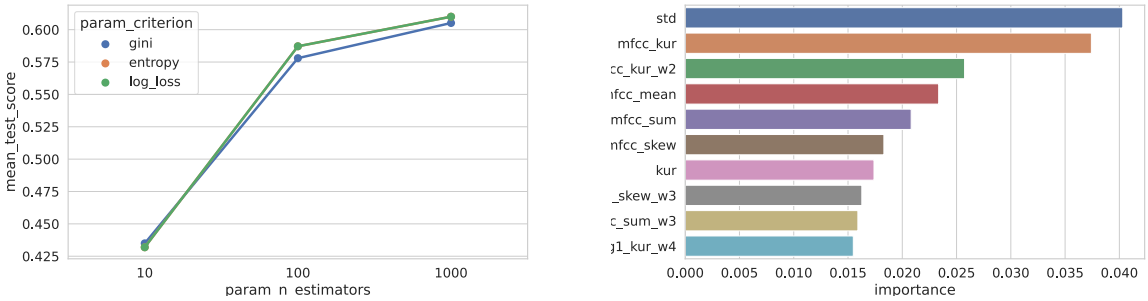

Figure 19: Differences in performances between different numbers of estimators in Random Forest (on the left), and the 10 most important features (on the right)

In Figure 19, it can be seen that the performance of Random Forest improves significantly as the number of estimators increases: with a number of estimators equal to 1000, an Accuracy over 60% is achieved in cross-validation. There is no particular difference in using one criterion for measuring impurity over another, but Entropy turned out to be the best choice.

As already seen for Linear-SVC, also in this case the 10 most relevant features were analyzed in the classification process. Figure 19 shows how the general standard deviation (std) is once again a very important characteristic. Similarly, the data relating to the Mel-Frequency Cepstral Coefficients (MFCC) are also reaffirmed as very important, among the top 5 features. Among the generic features, kurtosis (kur) also appears to have a certain importance, indicating how the distribution of data is equally important for the classification of emotions.

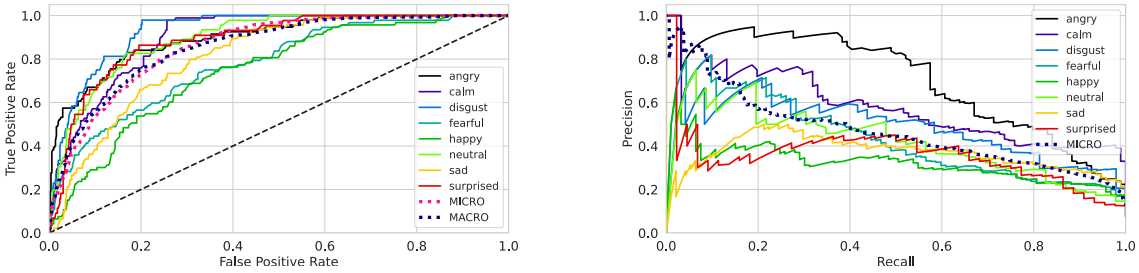

Figure 20: ROC curve (on the left) and Precision-Recall curve (on the right) of Random Forest for the complete dataset

The curves in figure 20 show a behavior similar to that already observed for other models: even the Random Forest shows difficulties in classifying emotions such as *sadness*, *fear* and *happiness*, while it is better in classifying emotions such as *anger*, *calm* and *disgust*. At the same time, *neutrality* and *surprise* settle on average values in both curves. As already done for other models, it is possible to hypothesize that there are pairs of emotions similar to each other in the acoustic characteristics, which hinder each other in the classification process.

5.2 Classification results of the main models

For the classification tasks, tests were performed on three different datasets: the full dataset, a dataset with data related only to speeches, and a dataset with data related only to songs. Accuracy, F1, Precision, Recall, and ROC-AUC scores were calculated for each model.

As can be seen from table 4, in tests conducted on the full dataset, Nonlinear-SVC turned out to be the best model, managing to achieve an Accuracy of 46%. Multilayer Perceptron also achieved very similar scores, with differences of less than 1%.

Table 4: Classification results for the complete dataset

Model	*Accuracy*	*F1-score*	*Precision*	*Recall*	*ROC-AUC*
Linear SVM	40%	37%	39%	39%	-%
Non-Linear SVM	**46%**	**45%**	**47%**	**46%**	**85%**
Random Forest	43%	40%	43%	41%	86%
Multilayer Perceptron	46%	55%	46%	46%	84%
Logistic Regression	42%	40%	42%	40%	83%
Mean	43%	41%	44%	42%	84%

Table 5: Classification results for the just-speech dataset

Model	*Accuracy*	*F1-score*	*Precision*	*Recall*	*ROC-AUC*
Linear SVM	36%	33%	35%	36%	-%
Non-Linear SVM	**38%**	**36%**	**43%**	**38%**	**79%**
Random Forest	32%	29%	33%	33%	77%
Multilayer Perceptron	36%	34%	37%	36%	79%
Logistic Regression	36%	34%	35%	36%	79%
Mean	36%	34%	37%	36%	77%

Table 5 shows the results obtained by testing the same models on the dataset consisting of speech only. Compared to the tests performed on the full dataset, the drop in efficiency of the models is remarkable: Nonlinear-SVC, which remains the best model, obtains 8 percentage points lower in Accuracy. In this regard, we can see that for effective emotion classification, it is necessary to train the models with both speech and song data. Elimination of one of these categories can cause a drastic drop in efficiency.

Table 6: Classification results for the just-song dataset

Model	*Accuracy*	*F1-score*	*Precision*	*Recall*	*ROC-AUC*
Linear SVM	**36%**	36%	36%	**42%**	-%
Non-Linear SVM	35%	36%	39%	40%	**-%**
Random Forest	34%	37%	**42%**	39%	-%
Multilayer Perceptron	35%	**37%**	36%	41%	-%
Logistic Regression	35%	36%	35%	41%	-%
Mean	35%	37%	38%	41%	-%

Table 6 shows the results obtained by training the models solely with song data. In this case, we notice an additional drop in the performance of the models, which ranks below those obtained using speech only. We also note how the best model is not Nonlinear-SVC, but MLP-Classifier, which manages to get better results with this type of dataset. It should be noted that the table of tests performed on the just-song dataset does not include ROC-AUC scores, as the training set did not contain instances related to all emotions.[3]

In general, the results show that using data from only one vocal channel (feature *vocal_channel*) is not sufficient to ensure the same performance as using the full dataset. To achieve optimal performance and a good generalization, it is necessary to train the model to recognize emotions in both speeches and songs. However, analyzing only the differences between the results obtained with the speech-only dataset and the song-only dataset, it is possible to note that there are no substantial differences between the effectiveness of the two types of training: the emotion classification performances are very similar, and it is not possible to say that songs are able to convey emotions better than speech, or vice versa.

[3]The ROC-AUC score function of Scikit-Learn does not make it possible to calculate the result if the classes in the training set differ from those in the test set

5.3 Bagging and Boosting

To increase the performance of the models on the full dataset, it was decided to test two ensemble techniques: bagging and boosting (AdaBoost). Specifically, for the bagging classifier the number of estimators was set to 20, while for AdaBoost it was set to 100. These two values are twice the respective standard values provided by the Scikit-Learn library.

Table 7: Classification results for bagging on the complete dataset

Model	*Accuracy*	*F1-score*	*Precision*	*Recall*	*ROC-AUC*
Linear SVM	40%	38%	41%	39%	-
Non-Linear SVM	**47%**	**45%**	**46%**	**46%**	77%
Random Forest	44%	40%	45%	41%	**86%**
Multilayer Perceptron	46%	44%	46%	44%	85%
Logistic Regression	43%	41%	43%	41%	83%
Mean	44%	42%	44%	42%	83%

Table 7 shows the results obtained by the bagging technique with the various classifiers used. As can be seen, increases over the results observed in table 4 are minimal: the best model, Non-Linear SVM, manages to achieve an increase in Accuracy of only 1 percent. Similar behavior can be seen in the other models.

Table 8: Classification results for boosting on the complete dataset

Model	*Accuracy*	*F1-score*	*Precision*	*Recall*	*ROC-AUC*
Linear SVM	38%	37%	37%	37%	-
Non-Linear SVM	36%	40%	37%	**47%**	84%
Random Forest	**44%**	**40%**	**44%**	42%	**86%**
Logistic Regression	42%	40%	44%	43%	82%
Mean	40%	39%	40%	42%	84%

Table 8 shows that even using AdaBoost with most of the main models tested above did not help increase classification performance. In particular, there is a marked drop in performance in both types of Support Vector Machines (Linear SVM and Non-Linear SVM). The only models that manage to achieve performance in line with previous results turn out to be Random-Forest and Logistic Regression, which still show worse results.

In conclusion, taking advantage of classical bagging and boosting techniques did not lead to noticeable improvements in classification performance, and thus were not shown to be worthwhile to use.

5.3.1 Gradient Boosting Machines

Given the poor performance gains obtained with classical bagging and boosting techniques (AdaBoost), several Gradient Boosting Machines were also tested. Because of the long computation times required by this type of model, it was decided not to perform an hyperparameters tuning phase, keeping the standard values of the Scikit-Learn library. Also, for the XGBoost model, it was necessary to use Scikit-Learn's `LabelEncoder()` to encode the class values.

From the results shown in Table 9 we can make some key observations:

- CatBoost has the best performance among all the models, with the highest accuracy (48%), F1-score (48%), precision (46%), recall (46%), and ROC-AUC score (85%).
- Hist Gradient Boosting has the lowest performance, with the lowest accuracy (22%), F1-score (22%), precision (15%), and recall (18%).
- Gradient Boosting, XGBoost, and LightGBM have similar performance, with accuracy, F1-score, precision, and recall values ranging from 40% to 43%. The ROC-AUC scores for these models are also relatively close, ranging from 76% to 84%.

Table 9: Classification results for gradient boosting machines on the complete dataset

Model	*Accuracy*	*F1-score*	*Precision*	*Recall*	*ROC-AUC*
Gradient Boosting	43%	43%	43%	41%	84%
Hist Gradient Boosting	22%	22%	15%	18%	53%
XGBoost	40%	40%	39%	38%	76%
LightGBM	43%	43%	43%	41%	79%
CatBoost	**48**%	**48**%	**46**%	**46**%	**85**%
Mean	39%	39%	37%	37%	75%

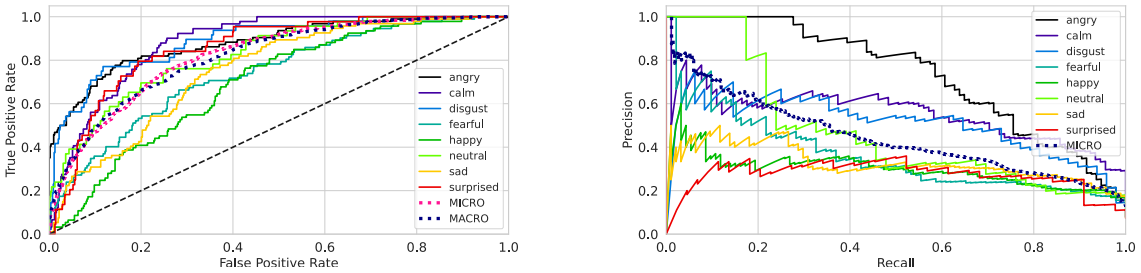

Figure 21: ROC curve for the different emotions of the Gradient Boosting model (on the left), Precision and Recall (on the right)

As can be seen in the figure 21, the emotion that is most accurately detected by Gradient Boosting is *angry*, and the lowest is *fearful*.

As can be seen in the 21 image, the behavior in classifying individual emotions is not particularly different from that seen for the main models. Emotions such as anger, calmness, and disgust continue to be identified better than the other

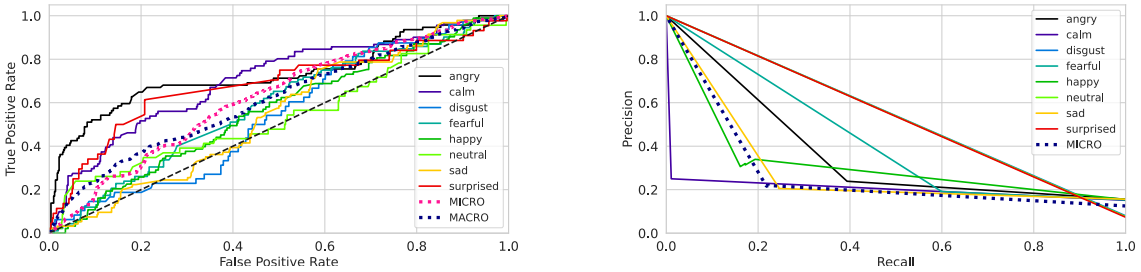

Figure 22: Precision and Recall for the different emotions of the Histogram-based Gradient Boosting model (on the right), ROC curve (on the left)

Histogram-Based Gradient Boosting is undoubtedly the least performing model. This is also visible from the graphs in Figure 22, where several anomalies can be seen, particularly in the Precision-Recall curve. Emotions such as *disgust*, *neutral* and *surprise* reported an accuracy of 0%.

XGBoost (figure figure 23) and LightGBM (figure 24) show behavior in classifying individual emotions comparable to that observed for classic Gradient Boosting. With an Accuracy of 43%, LightGBM is the second-best model together with classic Gradient Boosting.

CatBoost (figure 25) turns out to be the best model since it gets 60% accuracy for *angry* and *calm* emotions, and even for emotions where the accuracy is lower the results are still good. For example, the worst emotion is *sad*, where you still get 40% accuracy, which is a good result when compared to other Gradient Boosting models. Overall, CatBoost turned out to be the best classification model obtained so far.

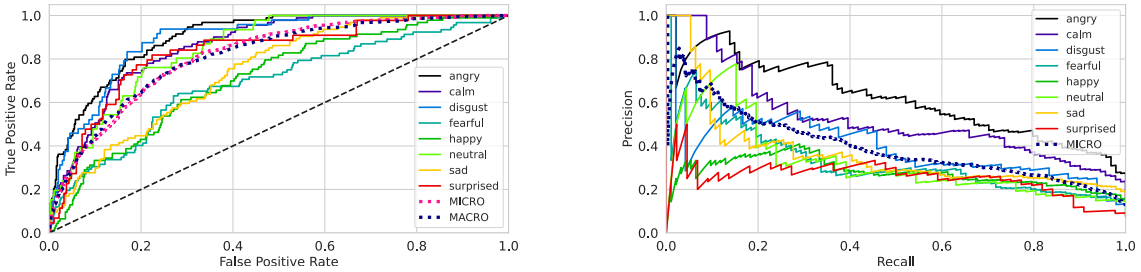

Figure 23: Precision and Recall for the different emotions of the XGBoost model (on the right), ROC curve (on the left)

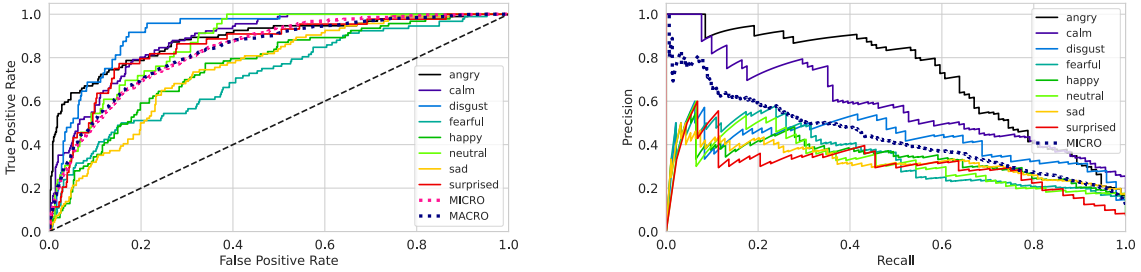

Figure 24: Precision and Recall for the different emotions of the LightGBM model (on the right), ROC curve (on the left)

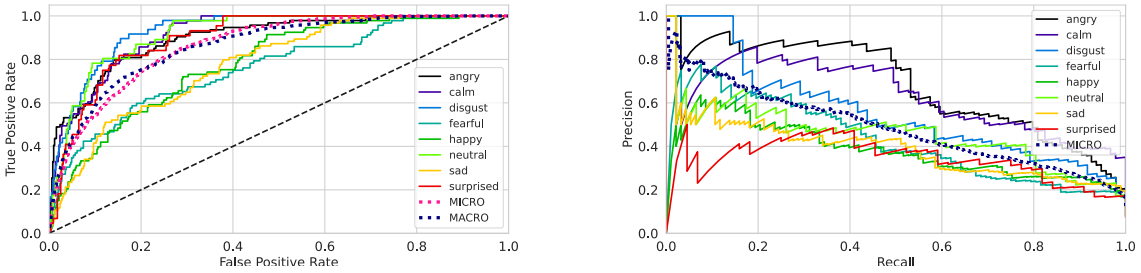

Figure 25: Precision and Recall for the different emotions of the CatBoost model (on the right), ROC curve (on the left)

6 Advanced Regression

In this section, two advanced regression models have been analyzed: Multilayer Perceptron Regressor and Random Forest Regressor. Specifically, the models were tested for the regression of 3 of the attributes that were found to be most important for the classification process (see section 5.1): *std*, *mfcc_kur*, *mfcc_kur_w2*. These two specific models were chosen due to their ability to work with multiple regression tasks.

For both models, a tuning phase of the hyperparameters was performed, with the same methods used for the classification. Specifically, for Multilayer Perceptron the following parameters were tested:

- Hidden layer size (`hidden_layer_sizes`), with a single hidden layer with number of neurons of 16, 32, 64, 128
- Activation function (`activation`), with the *Logistic*, *Hyperbolic Tangent*, and *ReLU* function
- Learning rate (`learning_rate`), with *constant* or *adaptive* type
- Alpha parameter (`alpha`), with values of 0.0001, 0.01, 1

Instead, for Random Forest Regressor, the only parameter tested was the number of estimators, with values of 10, 100, and 1000. For both models, all the others parameters were left as default in the Scikit-Learn library.

It turned out that the best parameters for Multilayer Perceptron were 128 neurons, the ReLu activation function, a constant learning rate, and an alpha parameter equal to 1. At the same time, for the Random Forest Regressor, the best result was achieved using a number of estimators equal to 1000.

Table 10: Regression results for both the models tested

Attribute	*Multilayer Perceptron*			*Random Forest*		
	R^2	*MAE*	*MSE*	R^2	*MAE*	*MSE*
std	0.81	0.27	0.17	0.73	0.38	0.25
mfcc_kur	0.75	0.33	0.25	0.70	0.37	0.29
mfcc_kur_w2	0.84	0.30	0.16	0.77	0.37	0.23
Mean	**0.80**	**0.30**	**0.19**	0.73	0.37	0.26

As can be seen from table 10, Multilayer Perceptron Regressor achieved the best average results for all three evaluation metrics. Even observing the scores relating to the single attributes on which the regression was performed, Multilayer Perceptron proved to be the best-performing model. These results are in line with what was observed in the classification phase: also in that case Multilayer Perceptron had obtained better results than Random Forest.

7 Time Series

This section illustrates the main data mining operations observed in the previous sections but performed on the time series of the audio tracks that compose the dataset under analysis. In particular, subsection 7.1 shows how the data were prepared for subsequent operations. Whereas, in sections 7.2, 7.3 and 7.4 the Clustering, Classification, and Motifs/Anomaly discovery operations are illustrated respectively.

7.1 Data understanding and preparation

The time series dataset follows some of the characteristics already illustrated in section 2: there is a training set made up of 1828 time series, and a test set made up of 624 time series. The distribution based on the categorical variables is the one illustrated in section 2.1.

The time series of the audio tracks have a variable length, but not exceeding 305.906 points. This length is complex to use for subsequent operations, given long calculation times. In this regard, it was decided to carry out data preparation operations to reduce the length of the time series, trying to sacrifice as little information as possible. In particular, the following operations were performed:

1. Cleaning: time series have been cleaned from NaN values. In the dataset, all columns with at least one NaN value have been removed, reducing the length of each Time Series to 140.941 points
2. Decimation: time series have been reduced to a tenth of their original length, removing 90% of the points that made them up
3. Noise removal: time series were processed to remove noise, using a rolling average with a 141-point window (1% of the maximum length of a decimated time series). This operation transformed the first 141 points (columns of the dataset) of the time series into NaN values. These columns have been eliminated as in step 1, bringing the time series length from 14.094 points to 13.955
4. Approximation: the time series has been further reduced by 90%, to 1395 points, through the Piecewise Aggregate Approximation (PAA)

Figure 26 shows how after the cleaning, decimation, and noise-removal operations, the original time series is much more compact and clean, with much less noise.

Following the initial data preparation operations, it was decided to perform a Piecewise Aggregate Approximation (PAA) to further reduce the size of the time series, but maintain a high degree of information. Figure 27 shows how the PAA succeeds in excellently approximating the behavior of the time series

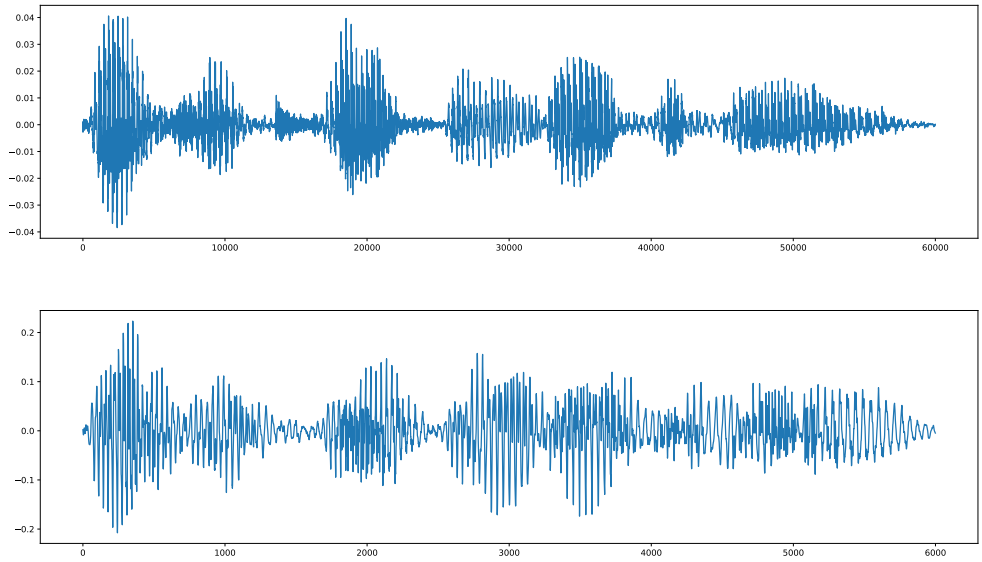

Figure 26: Portion of a Time Series before and after cleaning, decimation and noise removal

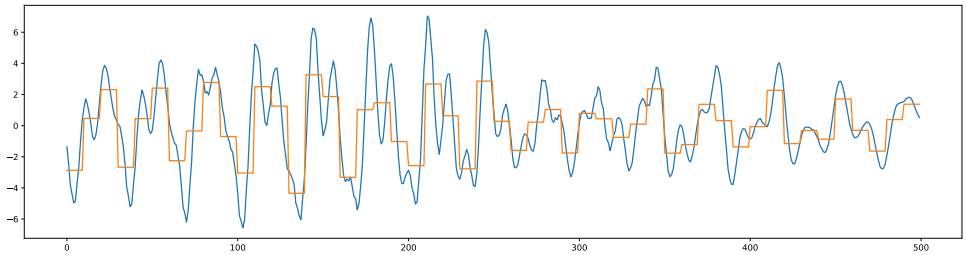

Figure 27: Portion of a Time Series with its PAA approximation

while extremely reducing its length. PAA was chosen because, compared with other approximations, it is very fast, and it allows the use of non-Euclidean distances, and weighted Euclidean distances.

7.2 Clustering

Clustering operations were done on two types of dataset, both reduced and approximated with PAA: the big one with 1395 columns, and the small one with 139. The reason behind the choice of using also a smaller approximate dataset lies in the fact that some algorithms combined with certain distances did not allow for fast enough computational time with a dataset of 1395 columns. In this regard, the smaller dataset was used in those tasks with longer computational time.

To find groups of time series in an unsupervised way, it was decided to try different techniques: shape-based (section 7.2.1), feature-based (section 7.2.2), and compression-based (section 7.2.3) clustering. With algorithms that needed a number of clusters in advance (e.g. KMeans and its variations), it was set to 8 as it is the number of emotions. For visualization, it was decided to show clusters with PCA and RSP, the two dimensionality reduction techniques which have been also used in section 3.

7.2.1 Shape-based Clustering

Shape-based clustering compares shapes. For this technique, it was decided to use KMeans and its variation TimeSeriesKMeans. KMeans was tested twice, both with the big and the small dataset. The first comparison that we can make is about centers. Every cluster has a center, a representative time series for the specific group. In figure 28 centers obtained with both datasets are displayed.

Obviously, cluster centers of the big dataset contain more information, but the ones of the small are clearer: it is visible how two of them (in brown and grey) have peaks at the beginning of the series, four

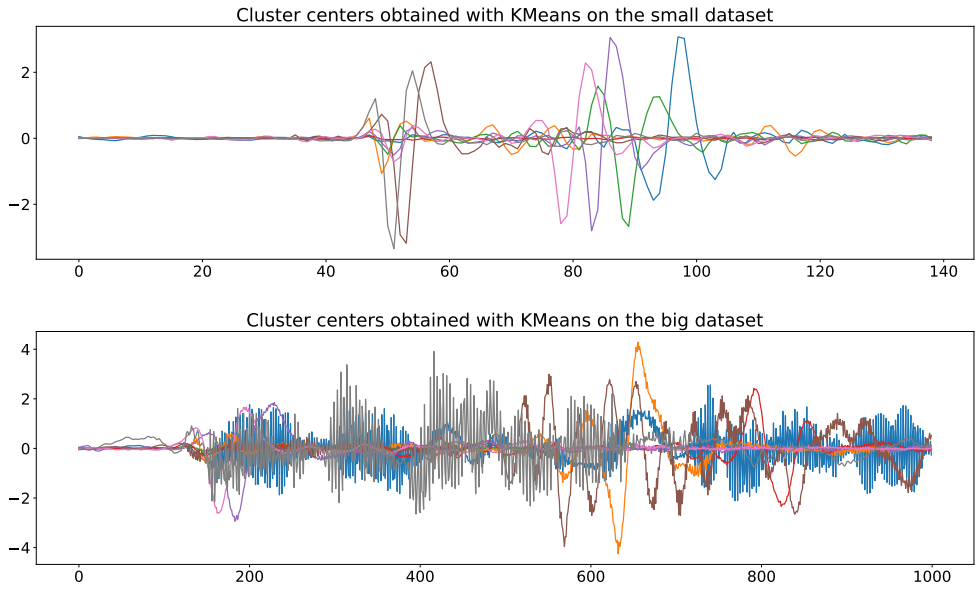

Figure 28: Cluster centers obtained with KMeans on small and big datasets

of them (in pink, purple, green and blue) have peaks in the second part of the time series, and the last two have almost no peaks. The distance of each time series from its center is called *inertia* and it is fundamental to judge how good the clustering is.

Results of the clustering obtained with KMeans are shown in figure 29, both with PCA and RSP. When using the small dataset, with both visualization there is a cluster that contains the majority of data. In PCA we have one cluster of 1487 elements, one of 152, and the others of about 30 elements. With RSP it is a little more balanced: there is one cluster of 1074 elements, four of about 100-150, and three of less than 100. The figure shows the number of elements inside each cluster, and it can be seen a very dense area in the middle with more or less sparsity around with respect to the visualization used. Inertia turned out to be high with both datasets (57593.63 with the small dataset and 821035.33 with the big one).

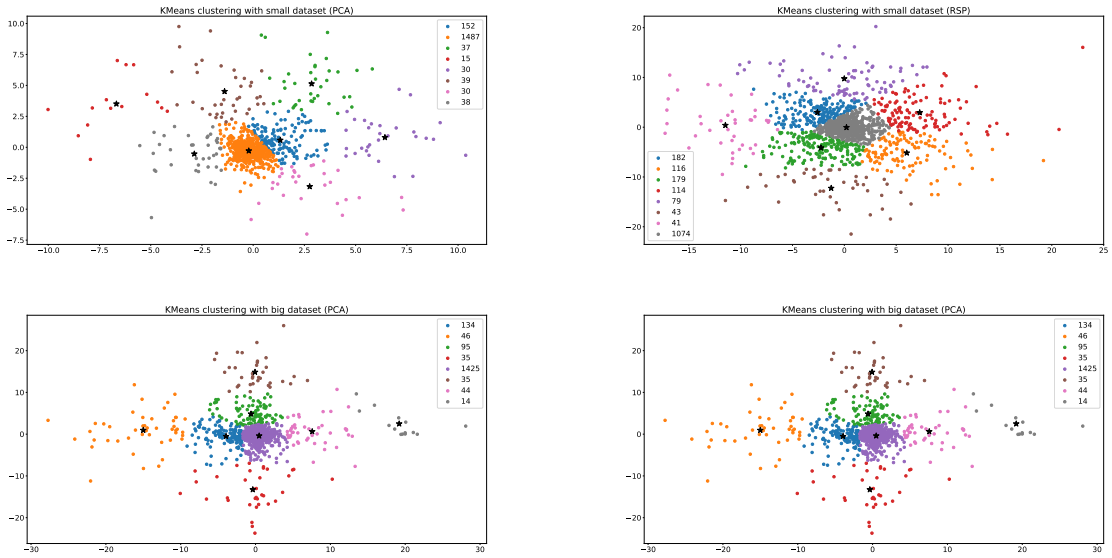

Figure 29: Principal Component Analysis (PCA) and Random Subspace Projection (RSP) representations of clusters obtained with KMeans on both the big and the small dataset. Centers are shown as black stars.

After that, TimeSeriesKMeans was tested with two different metrics: Euclidean distance and Dynamic

Time Warping. Since DTW is a slow algorithm, it was decided to use the small dataset with it and the big dataset with Euclidean distance. Results are shown in figure 30. It is visible how using DTW, clusters are very unbalanced with both visualizations: one of the groups contains the majority of data, and others are small compared to it. Using Euclidean distance this happens only with PCA visualization, while RSP clusters are almost balanced. Something to notice is inertia: with DTW it is very low (9.25) and with Euclidean distance, it is very high (449.14). This difference is important even if expected: DTW is applied to align the Time Series.

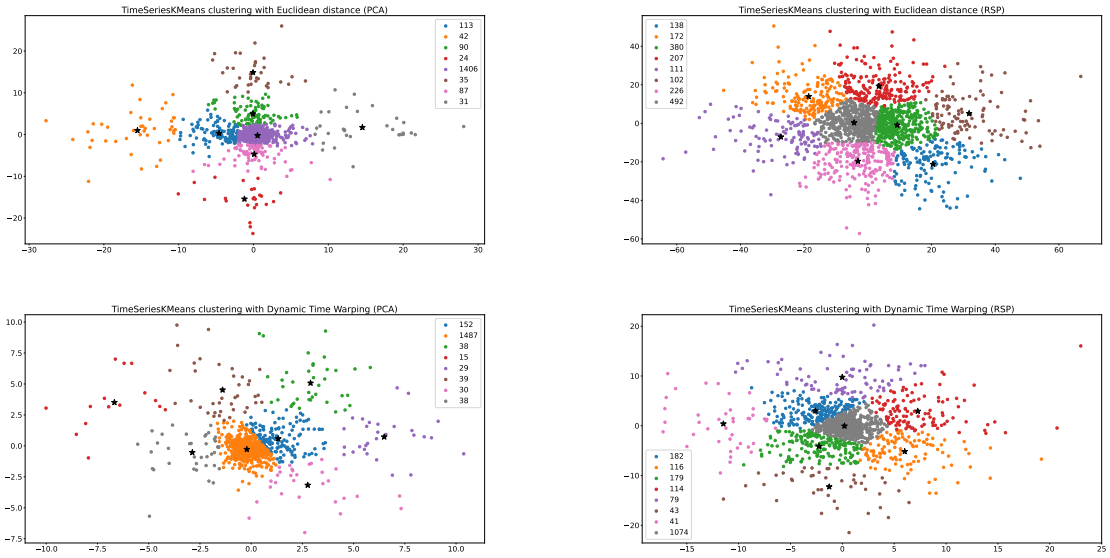

Figure 30: Principal Component Analysis (PCA) and Random Subspace Projection (RSP) representations of clusters obtained with TimeSeriesKMeans with Euclidean distance and Dinamic Time Warping. Centers are shown as black stars.

Finally, knowing that KMeans and its variations don't find noise points but assign every point to a cluster, we can consider the best clustering as the one with the smallest inertia, so the one in which time series in the same cluster are more similar to their center (and to each other). In this case, the best shape-based clustering results were obtained with TimeSeriesKMeans using DTW as a metric. Instead, the most balanced division was obtained with TimeSeriesKMeans using Euclidean distance as a metric and RSP visualization.

7.2.2 Feature-based Clustering

Feature-based clustering requires extracting some features from the time series. General features were extracted in this case, e.g. mean, median, standard deviation, percentiles, inter-quartile range, etc. The clustering has then been obtained without working on the temporal information of the time series but only on these features. It used KMeans with 8 clusters, on the big dataset. Inertia turned out to be 80.01, which is high but very much better compared to the one obtained with the shape-based clustering with KMeans.

Clustering results are shown in figure 31. With PCA visualization points lie on two lines, one more dense than the other, while with RSP visualization there is a dense region of points with a sparse tail. Clusters are almost equally divided in both of them.

7.2.3 Compression-based Clustering

Compression-based clustering involves a compression algorithm. It was decided to use CDM on the small dataset due to the computational time, and then DBSCAN and OPTICS, both with Euclidean distance as metrics, were performed on compressed data. This type of clustering obtained the worst results: it only assigns the point to one cluster with zero noise, or it considers all points as noise. Regarding DBSCAN, different values of epsilon were tried but the results didn't change. Representation results obtained with DBSCAN are shown in figure 32: with RSP there is a dense area in the middle, surrounded by some

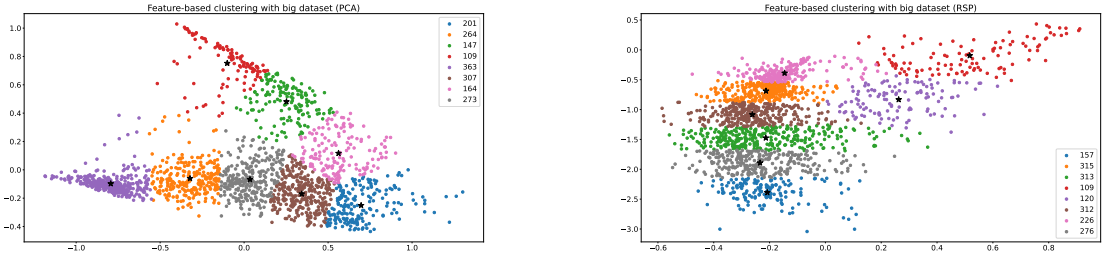

Figure 31: Principal Component Analysis (PCA) and Random Subspace Projection (RSP) representations of clusters obtained with KMeans using general features. Centers are shown as black stars

sparse data. With PCA there is a dense area on the left side with a sparse tail on the right. As can be seen, data failed to be clusterized.

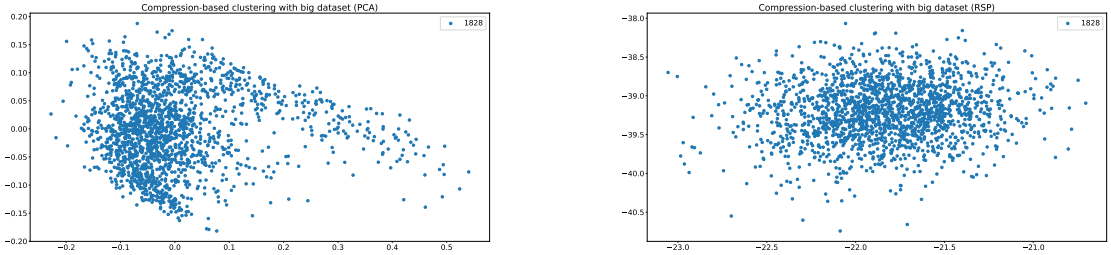

Figure 32: Principal Component Analysis (PCA) and Random Subspace Projection (RSP) representations of clusters obtained with DBSCAN using CDM

7.3 Classification

This section shows the results of the classification performed on the Time Series. Tests were carried out for different datasets, following the same line as in the feature-based and advanced classification: complete dataset, song-only dataset, and speech-only dataset.

Specifically, in section 7.3.1 the KNN algorithm was tested with two types of distance: Euclidean and Dynamic Time Warp (DTW). In addition, a shape-based method and a Convolutional Neural Network were also tested in section 7.3.2 and 7.3.3. Finally, in section 7.3.4, the results obtained with a feature-based classifier are shown.

The largest approximate dataset of 1395 columns was used for all tests. However, as also seen in clustering, a smaller 139-column dataset had to be used in some cases because of the long computation times required by certain algorithms.

7.3.1 KNN

This subsection shows the results of experiments conducted with the KNN Time Series model using two metrics: Dynamic Time Warp (DTW) and Euclidean Distance. For both metrics, it was decided to use the small dataset (139 columns), due to the very high computation time. Even using the Sakoe-Chiba band constraint for DTW, the computational time was still too high.

Dataset	*Metric*	*Accuracy*	*F1-score*	*Precision*	*Recall*	*ROC-AUC*
Complete	Euclidean	17%	11%	**23%**	22%	51%
Just Speech	Euclidean	15%	10%	18%	**25%**	**52%**
Just Song	Euclidean	15%	13%	17%	17%	**52%**
Complete	Dynamic Time Warp	**18%**	**17%**	16%	16%	-
Just Speech	Dynamic Time Warp	**18%**	**17%**	16%	16%	-
Just Song	Dynamic Time Warp	**18%**	14%	16%	15%	-
	Mean	17%	14%	18%	18%	52%

Table 11: Classification results for KNN

Table 11 shows that using DTW gives better results even when using datasets containing only songs or only speech. Conversely, using the Euclidean distance results in underperformance in all cases.

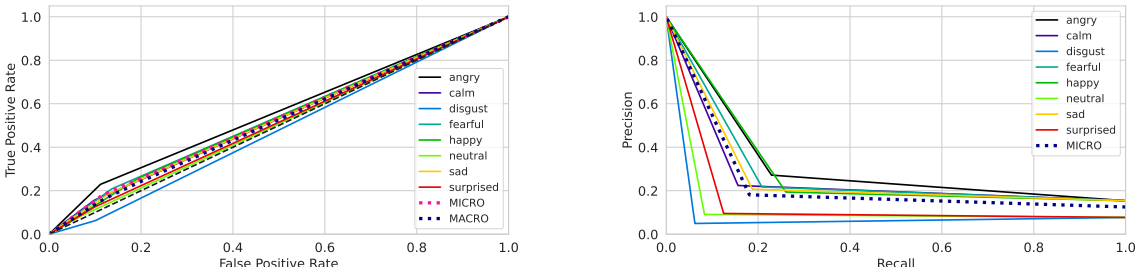

Figure 33: ROC curve (on the left) and Precision-Recall curve (on the right) of KNN with DTW for the complete dataset

In figure 33 it can be seen that in the ROC curve, there are no particular differences between the various emotions. It can be noticed that the emotion *disgust* is the worst to be classified, as it is under the diagonal line. On the contrary, emotion *angry* is the best one. In general, the classification of emotions by time series with KNN obtains poor results.

7.3.2 Shape-based

For the Shapelet Classifier, it was decided to use the learning-based methodology, also to optimize the computation times.

Dataset	*Accuracy*	*F1-score*	*Precision*	*Recall*	*ROC-AUC*
Complete	15%	08%	12%	20%	48%
Just Speech	14%	06%	13%	13%	50%
Just Song	**16%**	**10%**	**17%**	**21%**	**55%**
Mean	15%	08%	14%	18%	51%

Table 12: Classification results for Shapelet Classifier

As can be seen in the table 12, for the Shapelet Classifier the highest performance was obtained using the complete dataset and with the dataset containing only songs, which achieved similar results. The drop in performance observed using the dataset with only speeches is considerable.

In figure 34 it is visible that many emotion classification performances lie below the diagonal. In particular, *calm*, *happy*, *sad*, and *neutral* do not reach values above the diagonal line, while *angry* and *disgust* achieve the best results, as happens with other classifiers. In general, in addition to very low performances, it can be noted that there are no marked differences between the classification of individual emotions, which have very similar classification scores.

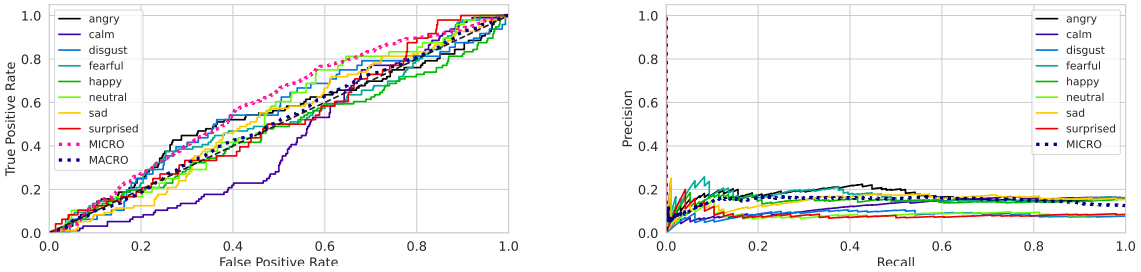

Figure 34: ROC curve (on the left) and Precision-Recall curve (on the right) of Shapelet Classifier for the complete dataset

7.3.3 Convolutional Neural Network

A simple convolutional neural network (CNN) was tested through the Keras framework. The model architecture can be described as follows:

1. **Input Layer:** This layer accepts an input of shape (*n_timesteps*, 1), where *n_timesteps* represents the number of time steps in the input sequence.
2. **Convolutional Layer 1:** This layer applies 16 filters to the input sequence with a kernel size of 8 and uses the Rectified Linear Unit (ReLU) activation function. The purpose of this layer is to extract local features from the input data.
3. **Batch Normalization:** This layer normalizes the activations of the previous layer, improving the stability and convergence of the network during training.
4. **Activation (ReLU):** This layer applies the ReLU activation function element-wise to introduce non-linearity into the model.
5. **Dropout:** This layer applies dropout regularization with a rate of 0.3, randomly setting 30
6. **Convolutional Layer 2:** This layer applies 32 filters to the previous layer's output with a kernel size of 5 and uses the ReLU activation function. It further extracts higher-level features from the input data.
7. **Batch Normalization:** This layer normalizes the activations of the previous layer.
8. **Activation (ReLU):** This layer applies the ReLU activation function element-wise.
9. **Dropout:** This layer applies dropout regularization with a rate of 0.3.
10. **Convolutional Layer 3:** This layer applies 64 filters to the previous layer's output with a kernel size of 3 and uses the ReLU activation function. It captures more complex patterns in the data.
11. **Batch Normalization:** This layer normalizes the activations of the previous layer.
12. **Activation (ReLU):** This layer applies the ReLU activation function element-wise.
13. **Dropout:** This layer applies dropout regularization with a rate of 0.3.
14. **Global Average Pooling:** This layer performs average pooling over the time dimension, resulting in a fixed-length feature vector. It helps reduce the dimensionality of the data.
15. **Dense Layer:** This fully connected layer has *n_outputs* neurons and uses the sigmoid activation function. It maps the extracted features to the output classes, producing the final predictions.
16. **Compilation:** The model is compiled with the sparse categorical cross-entropy loss function, the Adam optimizer, and the accuracy metric. The sparse categorical cross-entropy loss is suitable for multi-class classification tasks.

Dataset	*Accuracy*	*F1-score*	*Precision*	*Recall*	*ROC-AUC*
Complete	**26%**	**25%**	**26%**	**30%**	**67%**
Just Speech	20%	21%	22%	22%	62%
Just Song	18%	13%	15%	21%	52%
Mean	21%	20%	21%	24%	60%

Table 13: Classification results for CNN

The results obtained by CNN on the time series were clearly superior to all other models. As already seen, the complete dataset is the one that obtains the best results, while the dataset containing only speech still obtains good results (when compared to the other models). In line with all other tests, the dataset containing only songs results in the model significantly underperforming the other datasets.

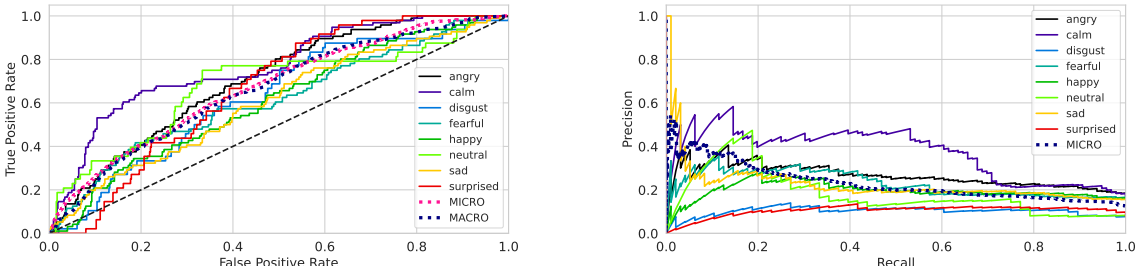

Figure 35: ROC curve (on the left) and Precision-Recall curve (on the right) of CNN for the complete dataset

As can be seen from figure 35, contrary to what was observed with other classifiers, emotions like *calm* and *neutral* achieved the best results. Others like *fearful* and *happy* can be considered the worst, but their scores are better with respect to previous models.

7.3.4 Feature-based

This model extracted the following features for each time series and then subjected them to the Decision Tree Classifier:

1. Mean (*avg*)
2. Standard Deviation (*std*)
3. Variance (*var*)
4. Median (*med*)
5. 10th Percentile (*10p*)
6. 25th Percentile (*25p*)
7. 50th Percentile or Median (*50p*)
8. 75th Percentile (*75p*)
9. 90th Percentile (*90p*)
10. Interquartile Range (*iqr*)
11. Coefficient of Variation (*cov*)
12. Skewness (*skw*)
13. Kurtosis (*kur*)

Dataset	*Accuracy*	*F1-score*	*Precision*	*Recall*	*ROC-AUC*
Complete	**21%**	21%	21%	22%	**60%**
Just Speech	17%	16%	16%	18%	56%
Just Song	20%	**23%**	**23%**	**25%**	49%
Mean	19%	20%	20%	22%	55%

Table 14: Classification results for Decision Tree (feature-based)

Table 14 shows that, again, the results obtained with the full dataset are the best. However, the dataset consisting only of songs gets very similar results, as opposed to the dataset containing only spoken sentences, which gets the worst results. In general, the results are still better than KNN and Shapelet Classifier.

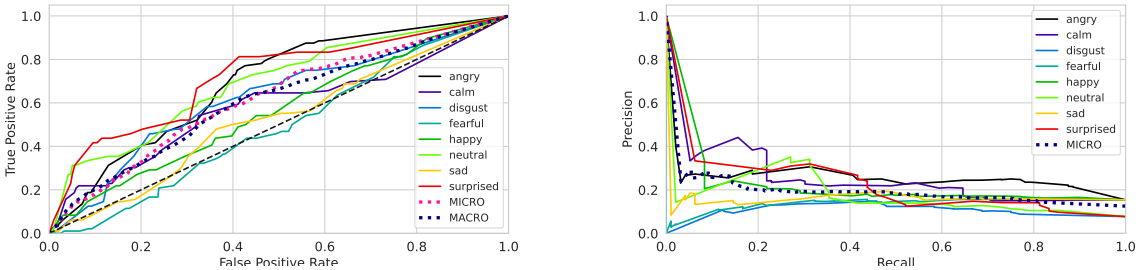

Figure 36: ROC curve (on the left) and Precision-Recall curve (on the right) of Decision Tree (Feature-based) for the complete dataset

As can be seen from figure 36, emotions like *sad* and *fearful* tend to go below the diagonal. Others like *surprised*, *neutral* and *happy* can be considered the best although they do not differ significantly from the others.

In general, classification by Time Series proved to be considerably less performant and more ambiguous than seen with classification by features. The ambiguity is also evident in the differences in performance between datasets: while in classification by features the full dataset always performed the best, in some tests performed by time series the just-song dataset performed better. The difference between the performance of the just-speech and just-song datasets also proved to be not always consistent: the just-song dataset proved to be better than the just-speech dataset several times, contrary to what was observed in the classification by features. However, the poor performance obtained cannot make us consider these observations as preponderant over those made by classification by features.

7.4 Motifs and anomaly discovery

In this subsection, the motifs and anomalies relating to the same sentence pronounced with *calm* emotion, both spoken and sung, were analyzed and compared.

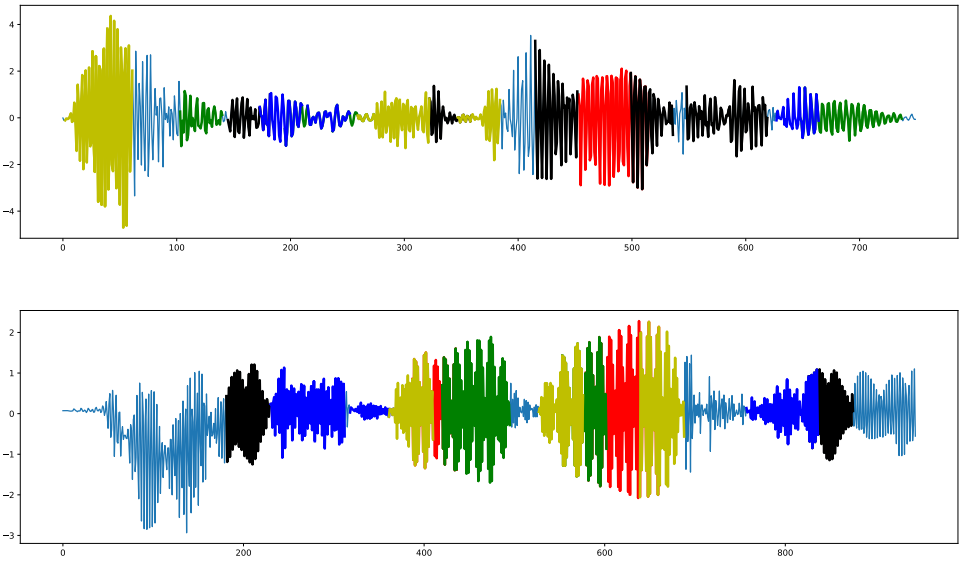

Figure 37: Motifs detected in the same calm sentence spoken (top) and sung (bottom)

Figure 37 shows how the two time series under analysis is remarkably different. In particular, the time series for the spoken sentence has a larger amplitude than that for the sung sentence, which instead fluctuates in a range from -3 to 2.

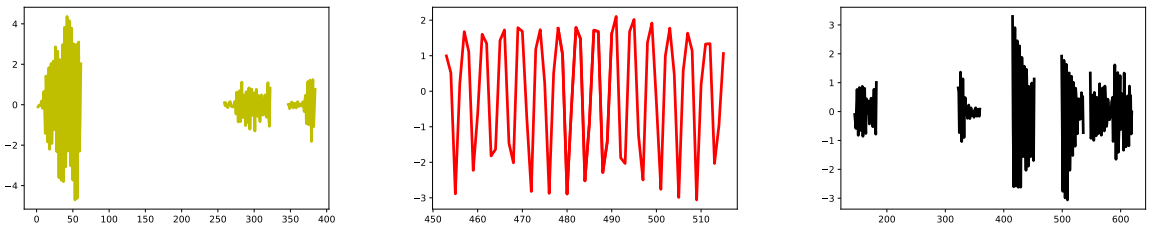

Figure 38: First three motifs discovered in the spoken sentence

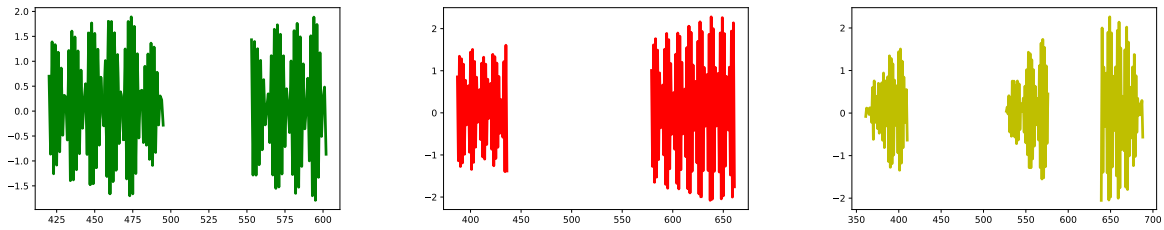

Figure 39: First three motifs discovered in the spoken sentence

In each case, both sentences have several motifs, as shown in figure 38 and 39. Limiting ourselves to the first three, we see that the motifs found typically have a sinusoidal-like shape and that in some cases (spoken sentence) they are detected one immediately following the other.

In most cases, the motif occurs within the time series 2 to 3 times. In the case of the spoken phrase, however, some motifs (such as the third) are detected five times within the time series. In general, motifs seem to resemble each other between the spoken sentence and the sung sentence, showing no particular differences.

Figure 40 shows that anomalies were also detected in both sentences. In the spoken sentence, anomalies seem to be located more in the stretches of the time series where amplitude narrowings are present. In

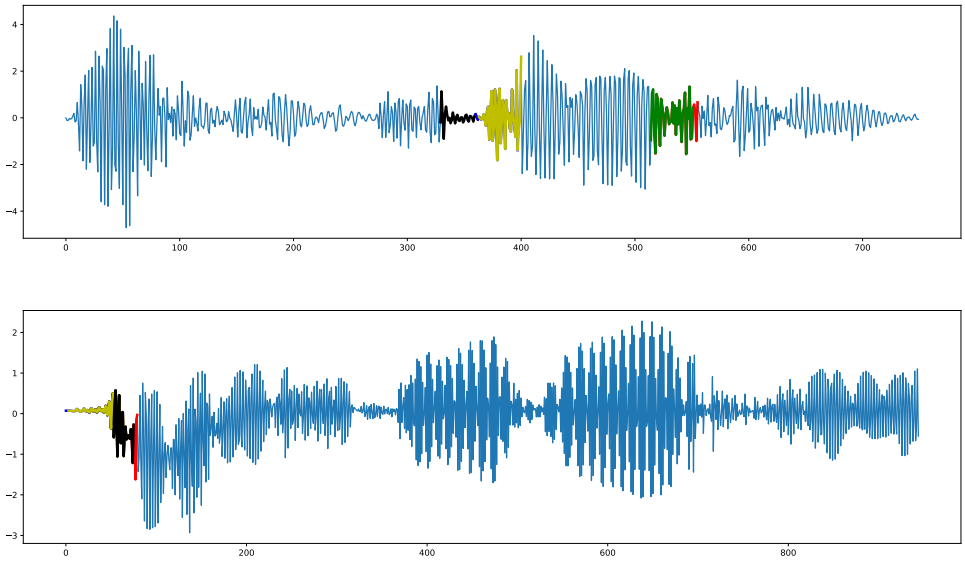

Figure 40: Anomalies detected in the same calm sentence spoken (top) and sung (bottom)

contrast, in the sung phrase, the only anomalies detected are located at the beginning of the time series and could be attributable to the fact that the phrase has not yet started at that stage of the series.

8 Explainable AI

This section explores the classification of emotions developed in section 5, both globally (subsection 8.1) and locally (subsection 8.2). Specifically, in the local explanations, the differences between the classification of a song and a speech are explored, in line with the overall goal of the project.

8.1 Global explanation

In this first stage, it was decided to analyze the emotion classification process in global terms. To do this, the best-performing model among those tested in section 5 was considered: CatBoost (with an Accuracy of 48%). The predictions obtained from this model were used to train a simpler model: Decision-Tree.

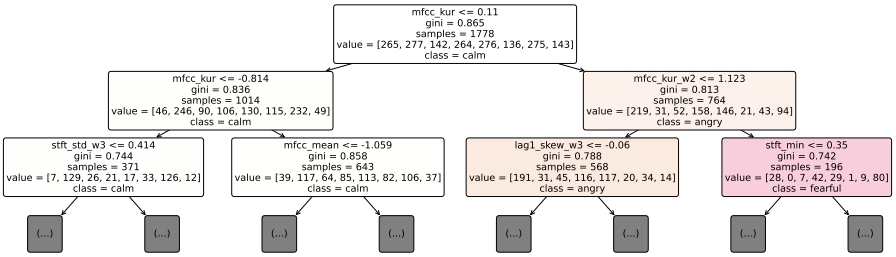

Figure 41: Global explanation Tree for emotions classification with CatBoost

Figure 41 shows the first two levels of the decision tree obtained with CatBoost predictions trained for emotion classification. As can be seen, features related to *Mel-Frequency Cepstral Coefficients* (MFCC) (particularly used in speech recognition), play a very important role in the early stages of classification. In particular, they allow immediate classification into two opposite emotional groups: calm (left in the decision tree) and anger/scared (right in the decision tree). MFCC kurtosis (feature *mfcc_kur*) turns out to be the first criterion considered for this subdivision.

MFCC-type features also turned out to be among the most important when the behavior of Linear-SVC and Random-Forest was analyzed in subsection 5.1. This gives us further confirmation of the reliability of this explanation.

8.2 Local explanations

In this subsection, it was decided to exploit three explainable AI algorithms to enlighten the differences between the emotions classification of spoken and sung sentences. Specifically, two sentences associated with the same emotion (*calm*), the same words, and the same actor, but with different vocal channels, were selected.

Specifically, subsection 8.2.1 illustrates the results obtained using the SHAP algorithm, subsection 8.2.2 illustrates the results obtained using *Local Interpretable Model-agnostic Explanations* (LIME) algorithm, while subsection 8.2.3 illustrates the results obtained by *LOcal Rule-based Explanations* (LORE).

8.2.1 SHAP

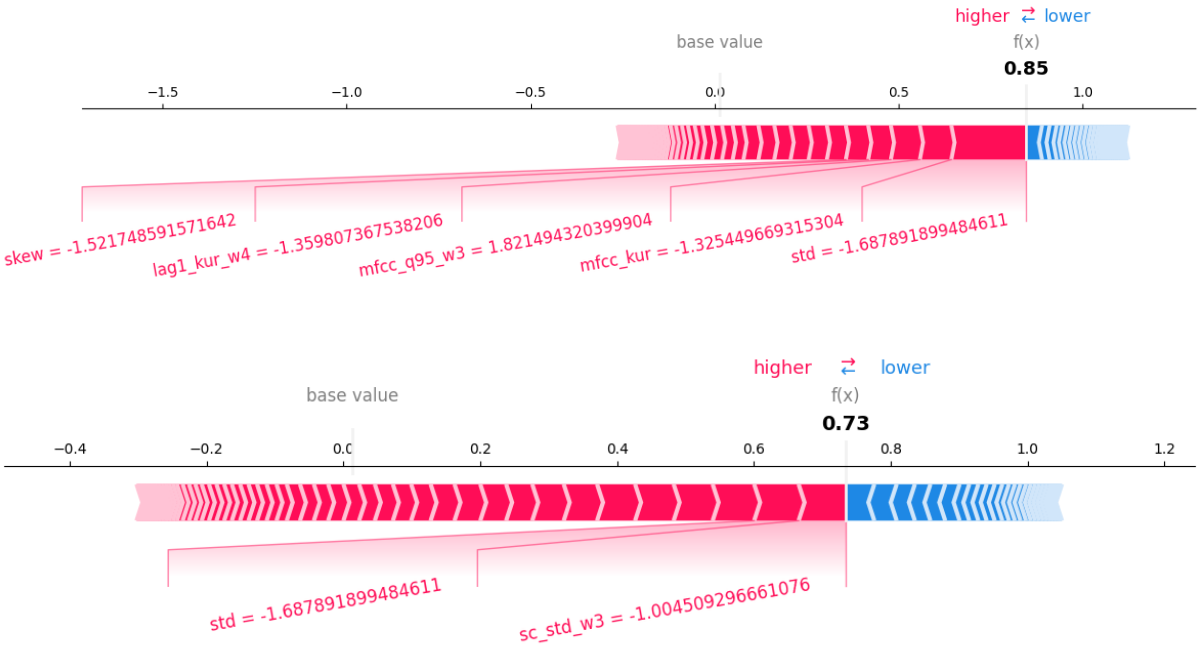

Figure 42: Explanations by SHAP algorithm for the classification of the same instance with different vocal channel: speech (top) and song (bottom)

Figure 42 shows some differences between the classification process performed for the same calm phrase, whether spoken or sung. First, we see that the probability associated with the spoken phrase (0.85) is slightly higher than that associated with the sung phrase (0.73). These values, although associated with only two phrases, may indicate how the classification of emotions in spoken phrases is simpler than in sung phrases.

Regarding the influence of features, we can see that the standard deviation (feature *std*) proves to be very important in the classification of calm in both sentences. In addition, the standard deviation associated with *Spectral Centroids* (feature *sc_std_w3*) also proves to be important in the sung sentence. In the spoken sentence, however, features associated with *Mel-Frequency Cepstral Coefficients* (MFCC) are again confirmed as important, albeit to a lesser extent than the standard deviation.

8.2.2 LIME

Figure 43 shows the results obtained by the LIME algorithm: as can be seen, the influence in the classification process of features already detected by the SHAP algorithm is confirmed (see figure 42). The standard deviation (feature *std*) turns out to be the most influential for both vocal channels.

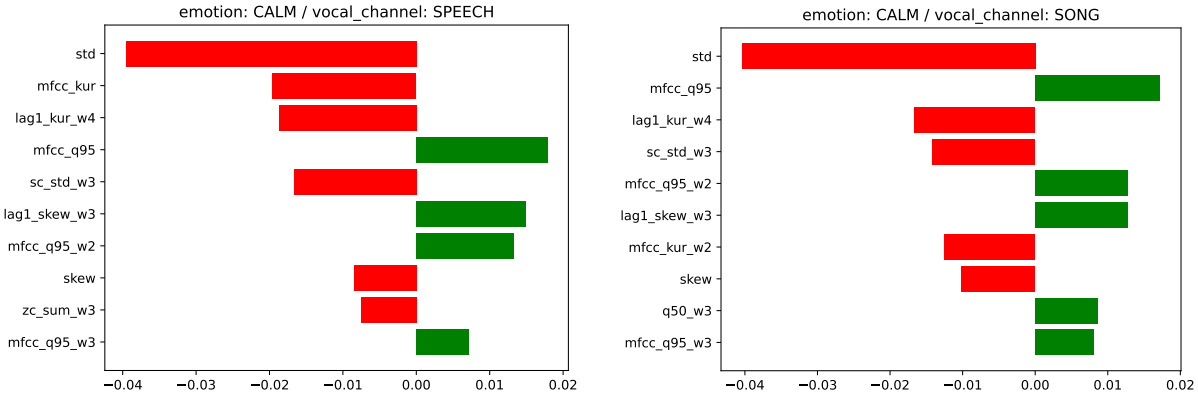

Figure 43: Explanations by LIME algorithm for the classification of the same instance with different vocal channel: speech (top) and song (bottom)

In general, the most influential features turn out to be the same for both spoken and sung phrases. The only differences can be detected in the presence of a feature related to zero crossing rate (*zc_sum_w3*) in the spoken sentence classification, while the presence of a quantile feature (*q50_w3*) for the sung sentence.

8.2.3 LORE

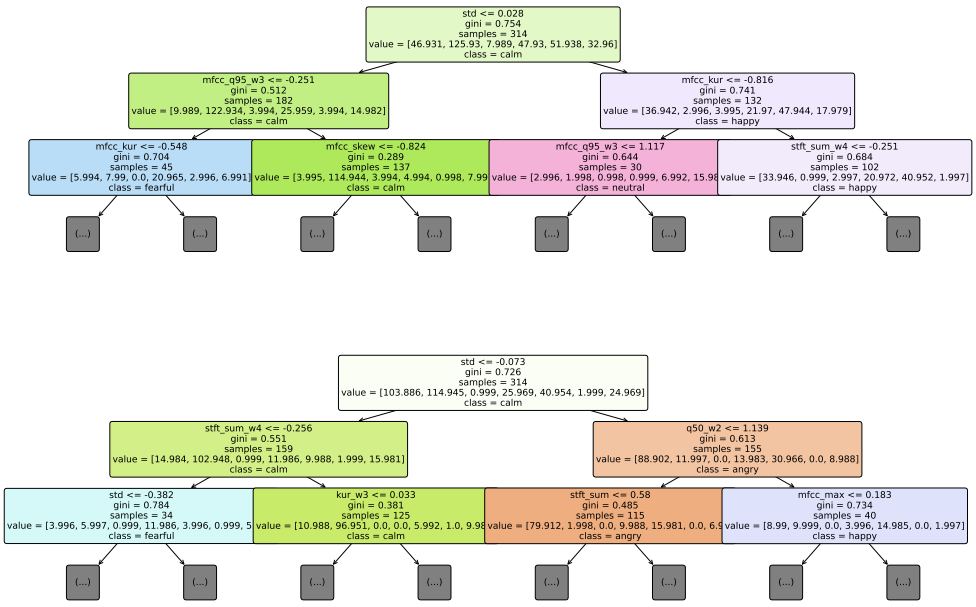

Figure 44: Explanations by LORE algorithm for the classification of the same instance with different vocal channel: speech (top) and song (bottom)

Figure 44 shows the decision trees containing the classification path followed for the same calm phrase, whether sung or played. As already seen in the analysis of SHAP results (figure 42) the standard deviation (feature std) asserts itself as the most important feature in the first step of classification. Both trees show how this feature acts as a watershed between two distinct emotional groups: the left subtree, dominated by *calm*, and the right subtree, dominated by "stronger" emotions such as *anger* and *happiness* (with some exceptions).

As already seen for SHAP, in the left subtree of the spoken sentence MFCC-type features are prevalent, establishing themselves as very influential in the recognition of this specific emotion. In contrast, they

do not seem to have the same importance in the classification of the sung sentence: instead, we find a feature related to *STFT chromagram* (*stft_sum_w4*)

9 Conclusions

In this project, several data mining operations were performed using a Dataset useful for emotion recognition and classification. In particular, it was decided to explore the differences in classification results between using a dataset including two types of vocal channels (spoken sentences and sung sentences), and two datasets including only spoken sentences or only sung sentences.

The objective was to analyze two main points:

- Is it possible to obtain the same classification results as a dataset with multiple vocal channels, but using a dataset with a single vocal channel?
- Do sung phrases allow a model to be trained more efficiently than spoken phrases, and thus convey emotions better, or vice versa?

The results showed that it is not possible to obtain the scores from training a complete dataset using only a dataset containing sung sentences or spoken sentences. All the tests performed show that the drop in performance of models trained with only one vocal channel is significantly high.

At the same time, no particular differences were shown between the models trained with a dataset composed only of spoken phrases or a dataset composed only of sung phrases. It is possible to detect only slight improvements when using only spoken phrases: consequently, we can state that training using only sung phrases is the least effective of all.

In this regard, we can hypothesize how spoken phrases can convey emotions better than sung phrases. Thus, spoken sentences allow better identification and recognition of emotionality, even if only slightly, and within the limits of a Machine Learning model.

Nevertheless, the only way to enable complete and effective emotion recognition is to use data belonging to multiple vocal channels.