RAVDESS analysis

Gianluca Becuzzi, Paolo Junior Mollica, Emanuele Vichi

June 20, 2023

Table of Contents

Introduction

This report aims at the description of the acoustic features of the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS). Two databases are utilized in the following sections: for Modules 1 and 2, the dataset is produced from the audio signal and is referred to as the **statistic's dataset**. Modules 3 and 4 were developed fully on the original **time series dataset**. Both contain the description of the sound expressions of **24 actors** as they vocalize **two different phrases** while interpreting various **emotions**, for a total of **2452 records**.

The statistic's dataset comprises of **434 fields**, the majority of which are taken from the recording's audio signal. Attributes describe the the signal statistics (intensity, mean, skewness, etc.) and the Fourier-related properties (spectral centroid, timbre using cepstral coefficient, etc.) and are labeled by the speaker's sex and other emotional characteristics. A summary of all the available features is displayed in Fig. 1.

Module 1

Understanding & preparation

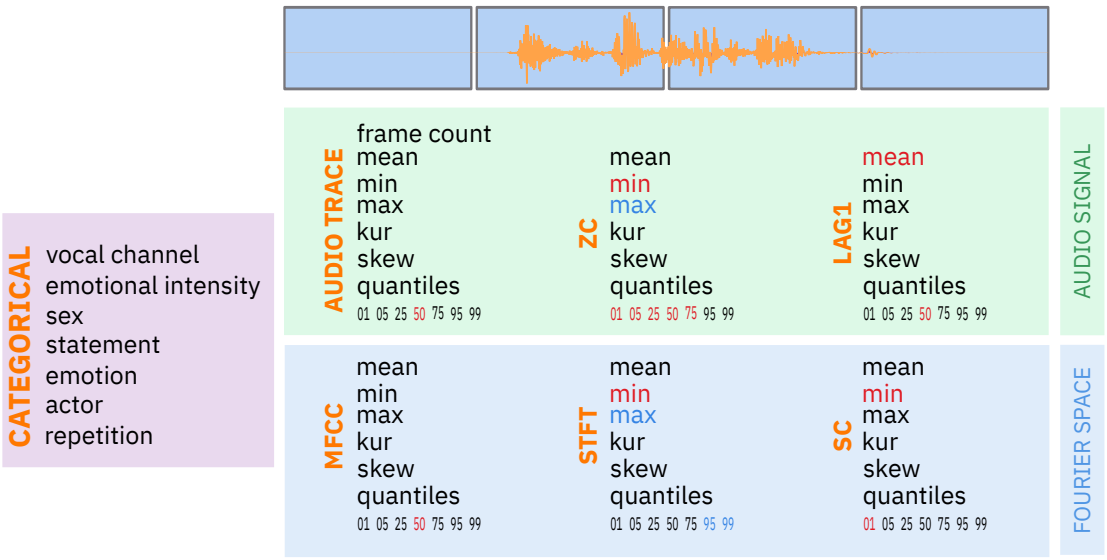


Figure 1: A synopsis of the dataset. The audio signal is separated into four time frames, each of which contains the displayed characteristics, which are organized into seven categories. Features that are entirely or abundantly equal to zero are shown in red, while those that are dominated by ones are shown in blue.

As seen in Fig. 1, many features are not relevant since they exhibit just one value over the whole dataset, therefore columns that are **populated by more than 50%** by a single value are removed.

Also, many features of the **`lag1`** family appear to be exponentially distributed. Among them, the strictly positive and strictly negative ones are **transformed** with

$$x' = \log |x|$$

.

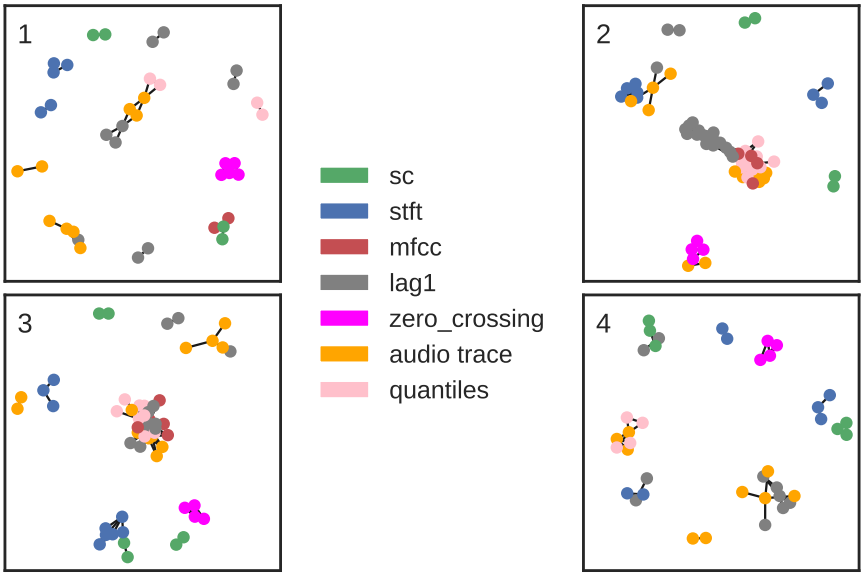


Figure 2: Highly correlated features (Spearman correlation coefficient > 0.95) colored by attribute family for each time window. For the first and the last window, correlated attributes belong predominantly to the same family, while for the two central frames features display a less trivial structure: `lag1`, `quantiles`, `mfcc` and audio features show inter-family relationships.

Highly correlated features (Spearman correlation greater than 0.95) appear to be organized in groups (Fig. 2). Among each group the **representative feature** is selected to be the one with the highest adjusted mutual information with some of the class, i.e. :

$$x_{best} = \arg\max_{x \in group} \left(\max_{C \in classes} AMI(x, y_C) \right)$$

where each attribute x is split in **30 bins** and y_C is the vector of values of the categorical feature C. After this procedure the number or attributes for each record is reduced from **434** to **250**.

Outliers

Outliers are detected **independently for each subgroup** (`sex`, `vocal_channel`) to avoid the deletion of falsely regarded data due to changes in the condition of the speaker.

As it is shown in Fig. 4, the tested outlier-detection methods (Isolation forest, Local outlier factor and Angle-based) tend to consider as erroneous records belonging to particular categories, since the conditional distribution is different and so the local density (for example anger records have much more sparse features). To overcome this issue, the score of each method is normalized and a final scoring is computed by uniform weighting of each:

$$S_i^{aggregated} = \frac{1}{3} S_i^{LOF} + \frac{1}{3} S_i^{ABOD} + \frac{1}{3} S_i^{ISFO}$$

Non-uniform weighting was found to be less balanced, as well as more difficult to judge due to a lack of criterion.

A `contamination` parameter (fraction of outliers over total) can be specified for each detection method used, thus the **parameters of each algorithm were compared by manual analysis** of

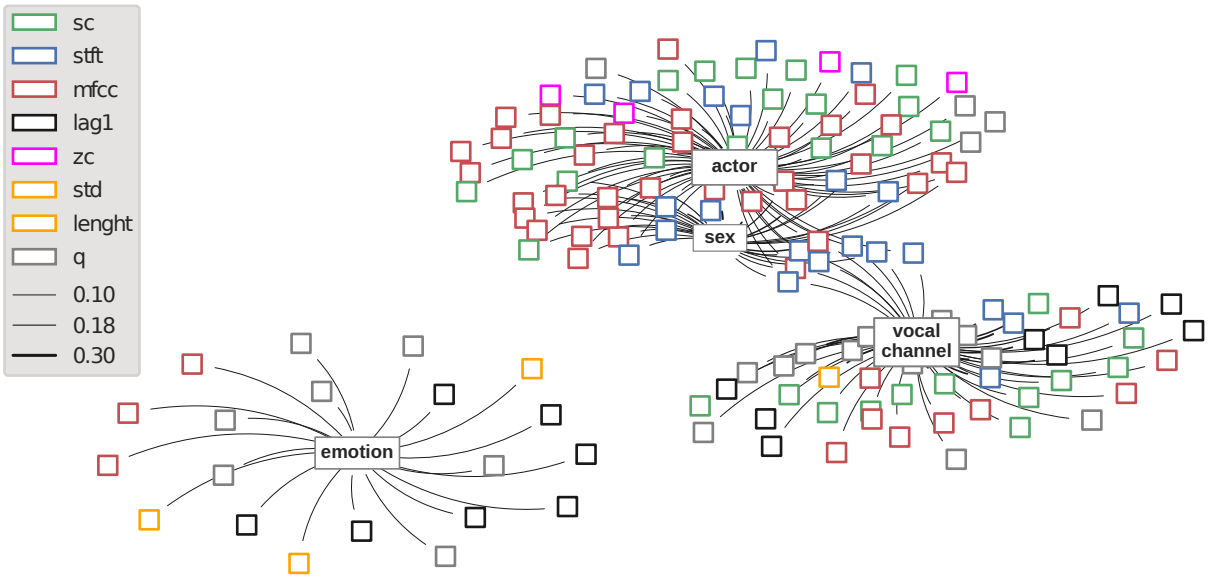


Figure 3: Adjusted mutual information between binned numerical attributes and categorical attributes represented as an embeddend network, where closer means more related. Threshold AMI was set to 0.08. Many `mfcc` and `stft` attributes are related to `sex` and `vocal_channel` while `emotion` is mostly linked to `lag1` and quantile attributes. Furthermore, the spectral centroid is implied in both `actor` and `vocal_channel` discrimination.

the categories of the found outliers (Fig. 4). Default parameters (100 estimators for `Isolation Forest`, 20 neighbors for `Local Outlier Factor` and 5 neighbors for ABOD) were judged to be the best, result of this procedure are represented for Isomap and UMAP embeddings in Fig. 5b.

Imbalanced Learning

The feature on which to perform imbalanced learning was decided on the basis of Fig. 3: in accordance with the values of the mutual information, the **`vocal_channel`** feature was judged to be the most predictable among the categorical variables, and for this reason it was chosen to accomplish this task.

Originally the classes' populations displayed a only slight asymmetry (58% speech, 42% song), so an **imbalance was introduced by deleting** a certain amount of 'song' records in order to obtain 92%-8% proportions.

Before continuing, it should be noted that features selection, for both the undersampling and oversampling tasks, was done based (again) on Fig. 3, and thus excluding columns prefixed with "zc". In addition, since the classifier used for both tasks is KNN, data were scaled with a `MinMaxScaler`.

Undersampling

Subsampling was performed using a **`CondensedNearestNeighbour`**, with which the training set was changed from one of the type [speech: 1071, song: 85] to one of the type [speech: 85, song: 79], as shown in Fig. 8a. Then, hyperparameters of the KNN model were tuned by grid search (Tab.

Figure 4: Belonging categories of the identified outliers for each detection method. To avoid misleading interpretations, each value is normalized over the total number of records of the same category in the whole dataset (fewer records imply fewer outliers). Isolation forest finds as outliers predominantly `angry` and `fearful` records (supposedly owing to increased data sparsity), while ABOD classifies numerous `actor_13` entries as anomalous. Method aggregation reduces category selectivity.

1), and the results of the predictions on the test set are as shown in the confusion matrix, Fig. 8b.

The model reported an accuracy of 0.954619. The results obtained for the other measures are shown in Tab. 2.

Table 1: KNN hyperparameters

n_neighbors	14
weights	uniform
metric	euclidean

Table 2: Classification report for the undersampling case

	precision	recall	F1-score
speech	0.95	0.97	0.96
song	0.96	0.93	0.95
accuracy		95%	

Oversampling

Oversampling was performed using **SMOTE**, with which the training set was changed from one of the type [speech: 1071, song: 85] to one of the type [speech: 1071, song: 1071], as shown in Fig. 9a. Then, hyperparameters of the KNN model were tuned by grid search (Tab. 3), and the results of the predictions on the test set are as shown in the confusion matrix, Fig. 9b.

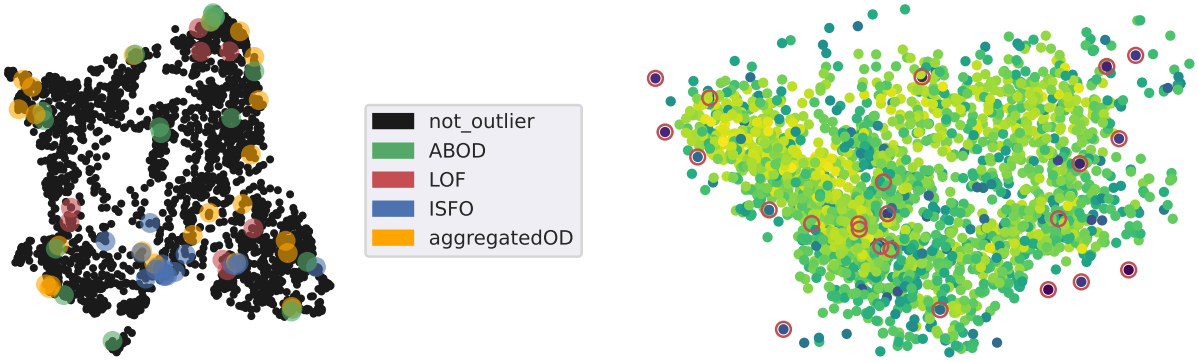
The model reported an accuracy of 0.944894. The results obtained for the other measures are shown in Tab. 4.

Table 3: KNN hyperparameters

n_neighbors	2
weights	uniform
metric	euclidian

Table 4: Classification report for the oversampling case

	precision	recall	F1-score
speech	0.94	0.96	0.95
song	0.95	0.92	0.93
accuracy		94%	



(a) (b)

Figure 5: (a) Outliers represented inside a UMAP embedding (15 neighbours). It can be seen that isolation forest is not balanced since it considers anomalous records belonging to a particular region of the manifold. Aggregating the outliers' scores seems to identify points at the edge of the space, that are roughly mapped at the edges of the embedding. (b) Records represented in a ISOMAP (15 neighbours) and colored by aggregated score (darker means more likely to be outlier). Red circles are the selected 1% of final outliers. Because detection was conducted on local subsets, some of the deleted entries are presented inside the bulk of points.

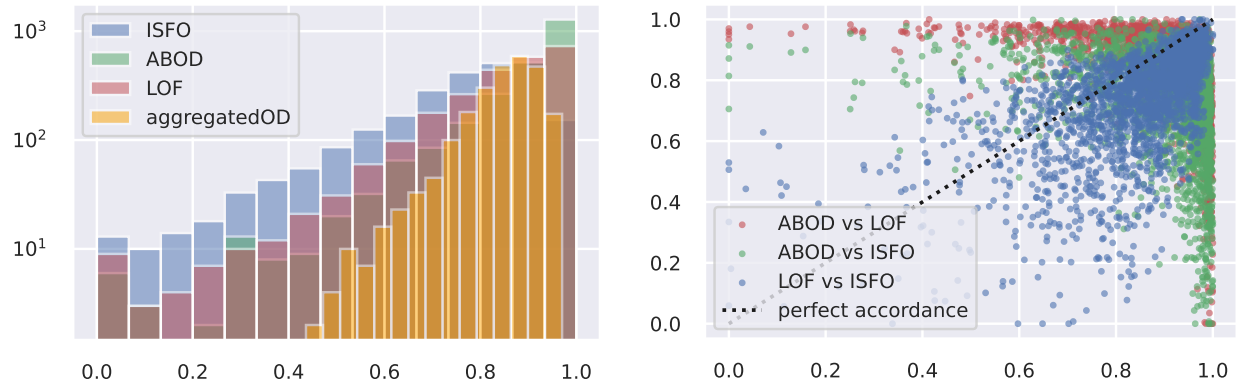
Module 2

Classification

In this section we'll perform, using as a task the one of the previous section, classification task using the following methods: **Logistic Regression, Support Vector Machines, Neural Networks, Ensemble Methods, Gradient Boosting Machines**. We decided to apply this methods to **`vocal_channel`** feature so we applied a one hot encoding on it.
As first step we selected only the columns of the Dataset with continuous attributes, because most of the metrics used in the different methods are built for continuous values, an encoding of the categorical attributes it might be useful for methods on decision tree like **Random Forest**.
Then we selected a **target class** between the two values of **`vocal_channel`**, in our case the target class was **Song**. So we did an **Training-Validation-Test** splitting because we are going to use a validation set in the parameters tuning route.
At this point the Dataset is ready and we'll perform the task applying the different methods. Given the large number of parameters involved in order to optimize the results we'll perform a parameters tuning route using OPTUNA an open source hyperparameter optimization framework to automate hyperparameter that efficiently search in large spaces and prune unpromising trials for faster results. So we firstly set a range/list of values for most significant parameters of each methods, then framework search for the combination of parameters that maximize the accuracy score. As evaluation we didn't report the ROC curve partly because the good results of each classifier brought to similar ROC that get very close to the ideal [0-1] step function and partly because allows us to avoid overloading the text with images.

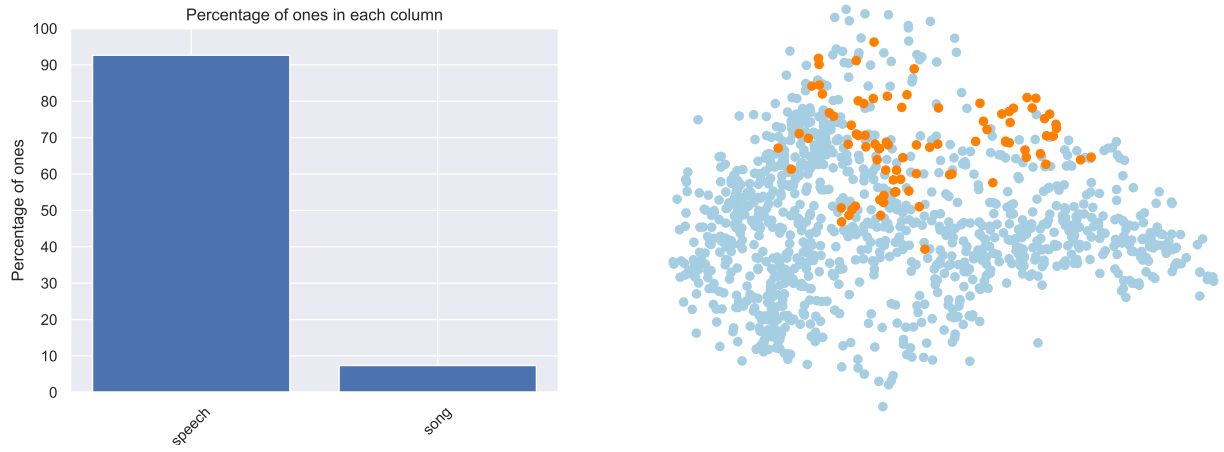
Logistic Regression

The class **Logistic Regression**, which implements regularised logistic regression and can handle both dense and sparse input, will be used in this section.
In Fig. 10 are shown the tested parameters, the optimal ones being highligthed.



(a) (b)

Figure 6: Distribution of the normalized score of each OD method (a) and accordance between them. (b) Aggregating the reduces the number of certain records (score near 1) and completely wrong ones (score near 0), giving a mitigated assessment. This is due to the almost completely uncorrelated estimate of each algorithm: many records have a near-zero score for ABOD but a near-one score for LOF, meaning that the estimates do not agree.



(a) (b)

Figure 7: (a) result of the unbalancing: 92% of the objects in the training set now belongs to the speech class, while the 8% belongs to the song class. (b) Imbalanced training set represented using an ISOMAP embedding.

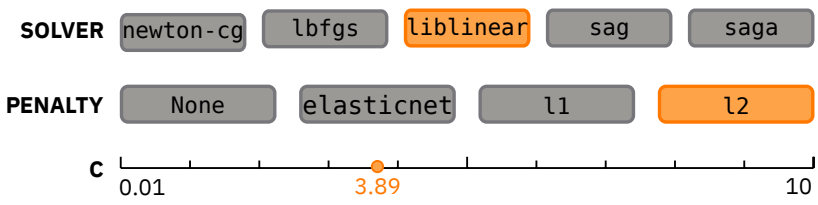
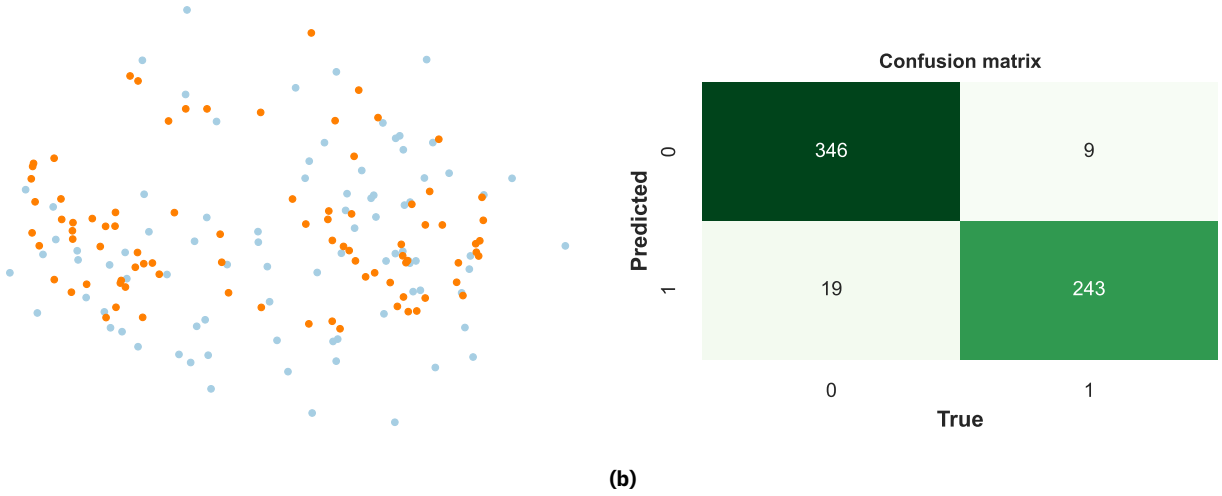


Figure 10: Logistic regression parameters (best ones in orange)

	speech	song
precision	0.91	0.99
recall	0.99	0.92
F1-score	0.95	0.96
accuracy	**95%**	

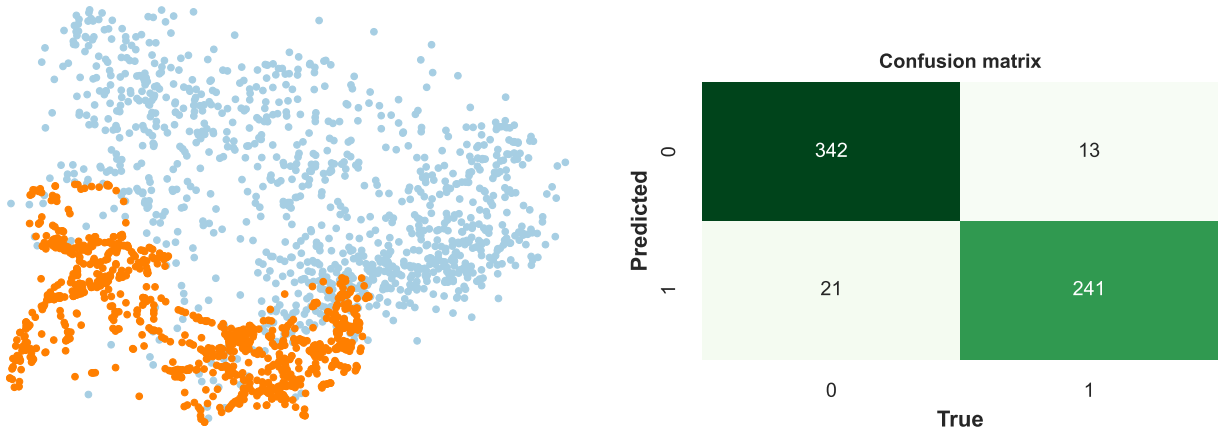
Table 5: Logistic regression classification report

Where **`Solver`** is the algorithm to use in the optimization problem, **`Penalty`** specify the norm of the penalty Ridge('l1')-Lasso('l2') or a combination al both and **`C`** which is the inverse of regularization so smaller value imply stronger regularization. Once we found the best parameters we passed them to the algorithm and so applied the classifier to the training set.



(a) (b)

Figure 8: (a) Undersampled training set represented using an ISOMAP embedding. (b) Confusion matrix containing the results of the classification on the test set, using KNN.



(a) (b)

Figure 9: (a) Oversampled training set represented using an ISOMAP embedding. (b) Confusion matrix containing the results of the classification on the test set, using KNN.

On the test set, we obtained an **accuracy score of 92%**, other evaluation results are reported in Tab:5.

Support Vector Machine: Linear SVC

In this section we will use Linear Support Vector Classification, a linear SVM is a classifier that searches for a hyperplane with the largest margin. Again we perform the parameters tuning trough the optimization routine the results are listed below (Fig. 11).

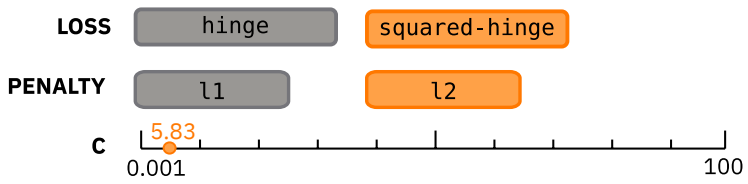


Figure 11: Linear SVM parameters (best ones in orange)

	speech	song
precision	0.92	0.99
recall	0.99	0.94
F1-score	0.96	0.97
accuracy	**96%**	

Table 6: Linear SVM classification report

Finding the support vectors and performing a PCA transformation on the training set, we are able to show the importance of parameter **C**, that once again is proportional to the inverse of regularization, this leads to greater margin for smaller values of C and viceversa smaller margin for larger values of C. An example of this behavior is shown in Fig. 12.

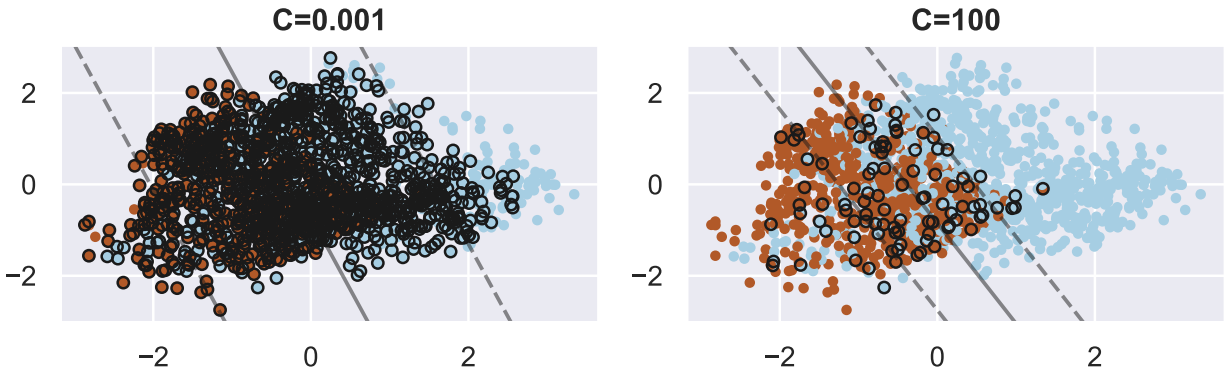
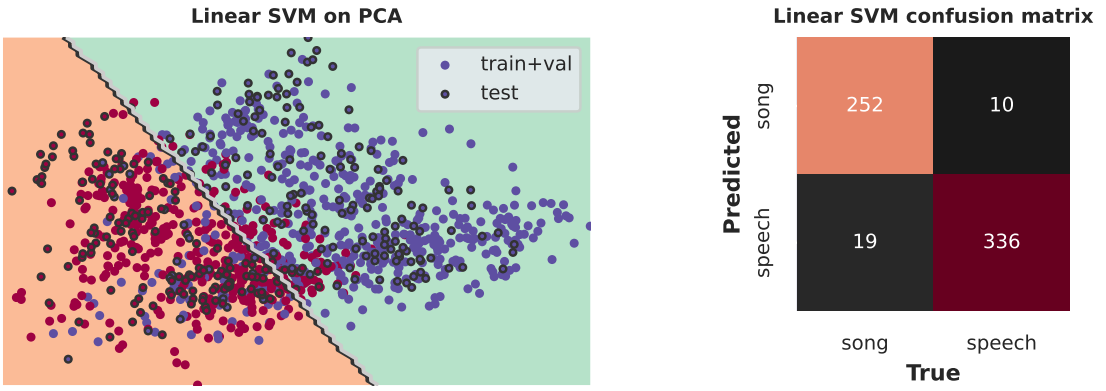


Figure 12: Example of the parameter C role: increasing its value reduces the number of support vectors.



(a) (b)

Figure 13: Results of linear SVM are quite similar to those of Logistic Regression. This can be attributed to the nature of the available data.

Support Vector Machine: Non linear SVC

This class of Support Vector Classification are used when the decision boundary is not linear, they exploit the "Kernel Trick", in particular they use kernel function expressed in terms of the coordinates

in the original space:

$$
\begin{aligned}
K_{POLY}(\mathbf{x}, \mathbf{y}) &= (\mathbf{x} \cdot \mathbf{y} + 1)^d \\
K_{RBF}(\mathbf{x}, \mathbf{y}) &= \exp(-|\mathbf{x} - \mathbf{y}|^2 / (2\sigma^2)) \\
K_{SIGMOID}(\mathbf{x}, \mathbf{y}) &= \tanh(k\, \mathbf{x} \cdot \mathbf{y} + \omega)
\end{aligned}
$$

Each kernel captures different aspects of the input features, so its choice is a central aspect of the model . The optimization routine converged to the following parameters (Fig. 14):

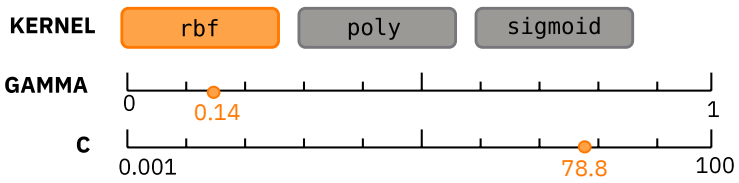
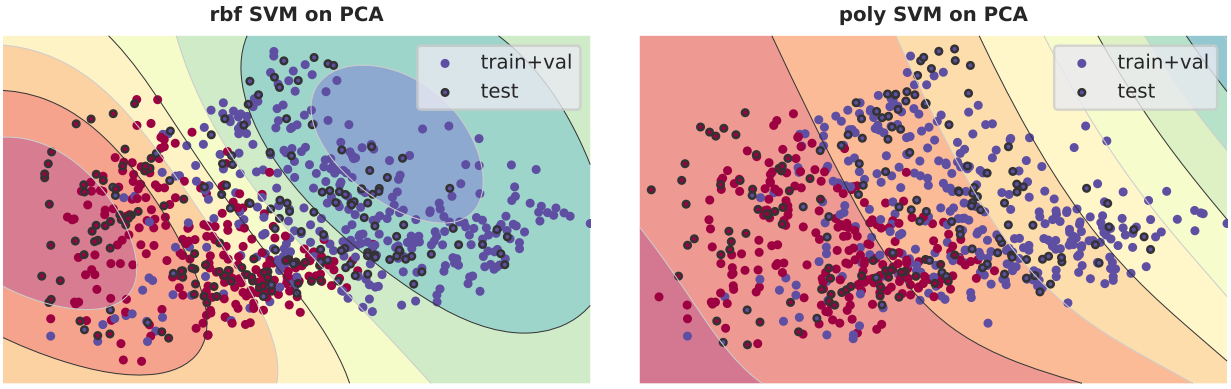


Figure 14: Nonlinear SVM parameters (best ones in orange)

	speech	song
precision	0.92	0.99
recall	0.99	0.94
F1-score	0.96	0.97
accuracy	**96%**	

Table 7: Classification report

In Fig. 15 we can observe how the different kernel act on the decision boundary.



(a) (b)

Figure 15: Bounds for different kernels. Points are a random subsample of the dataset. Both the polynomial and radial-basis kernel slightly grasp the non-linear properties of the boundary.

Neural Networks

Use a neural network to build a classifier offers many customization possibilities. In particular we built a multilayer model setting:

- **epochs** = 50
- **loss** : 'sparse_categorical_crossentropy'
- **metrics** : 'accuracy'

Instead to try to optimize the results we tuned the parameters: **`activation`**, **`hidden layer sizes`**

(implicitly the number of hidden layers too), **optimizer**.
The best combination of parameters is shown in Fig:16.

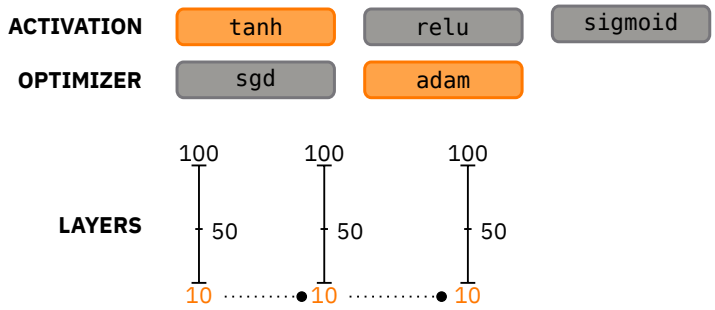


Figure 16: Neural network parameters (best ones in orange)

	speech	song
precision	0.95	0.99
recall	0.98	0.96
F1-score	0.97	0.98
accuracy	**97%**	

Table 8: Neural Network Classification report

Multilayer neural network are universal approximators but could suffer from overfitting if the network is too large or if we run too many epochs (which is why we set $epochs = 50$). So using the validation and training set we printed out the loss function for both during each epochs, the result is shown in Fig:17 where we can see that there is no overtrain.

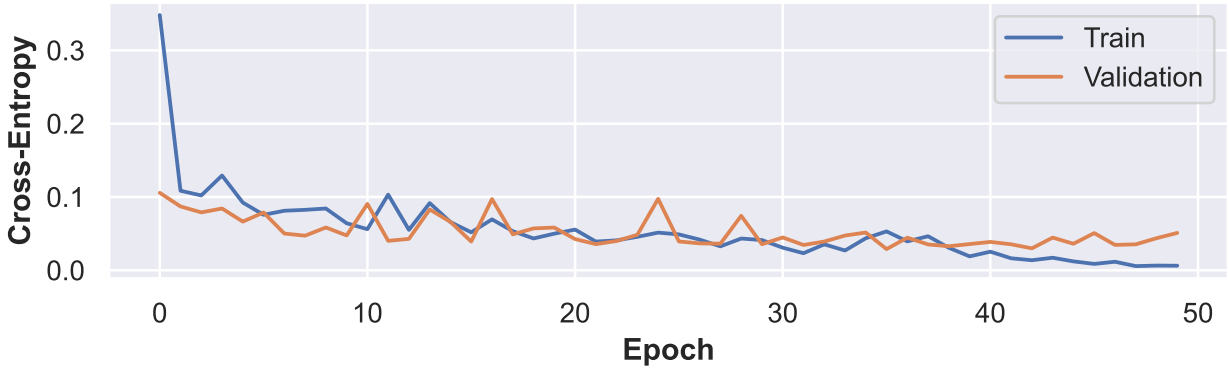


Figure 17: Cross-entropy varying with the training epoch. Training is stopped at the early beginnings of the generalization gap.

Ensemble methods: Random Forest

This class of ensemble methods specifically designed for decision trees, the predicted class of an input sample is a vote by the trees in the forest, weighted by their probability estimates.
The number of the trees in the forest is fixed by the parameter **n_estimators**, then for each tree we can set:
the function to measure the quality of a split through **criterion** parameter, furthermore when looking for the best split we could choose the number of features to consider with **max_features**, the maximum depth of the tree with **max_depth**, the minimum number of samples required to split an internal node using **min_samples_split** parameter and the minimum number of samples required to be at a leaf node with **min_samples_leaf**. In particular optuna setted to False the **Bootstrap**, this means that the whole dataset is used to build each tree while the default setting True is used

to insert randomness and so to avoid overfitting, but for how the model is built we expected that is was already resistant to overfitting.

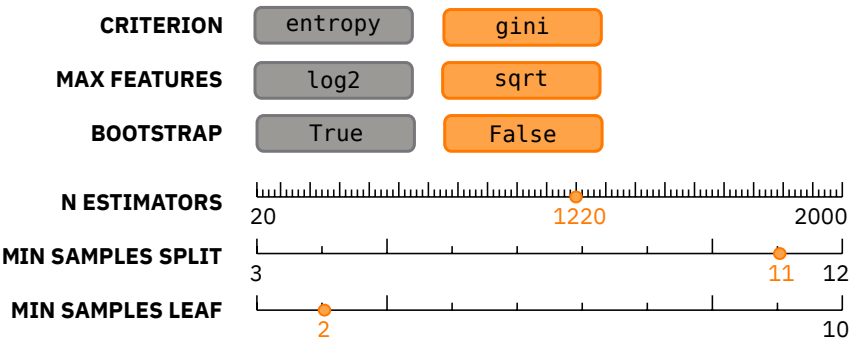


Figure 18: Random Forest parameters (best ones in orange)

	speech	song
precision	0.94	0.99
recall	0.99	0.95
F1-score	0.97	0.97
accuracy	**97%**	

Table 9: Classification report

Gradient Boosting

The Gradient boosting for classification use Log(odds) combined with decision trees.

Like for random forest there are a lot of parameters to tune that modify the structure of the built trees, most of which have the same semantics as before except for **`learning rate`** and **`loss`** where the first scale the contribution of each tree while the second represent the loss function to be optimized. As done for the previous methods, we have run the tuning routine obtaining the results reported below.

Due to the nature of the boundary (for which linear classifiers still score a high accuracy), it is not surprising to observe in Fig. 19 a **relatively high optimal learning rate** of 0.3 (roughly three times the default value), whereas, for the same reasons, an **optimal max depth of 90** was certainly not expected. This may indicate that the boundary, while being linear, is **describable in terms of a condition on multiple variables**, thus being more difficult to grasp using trees.

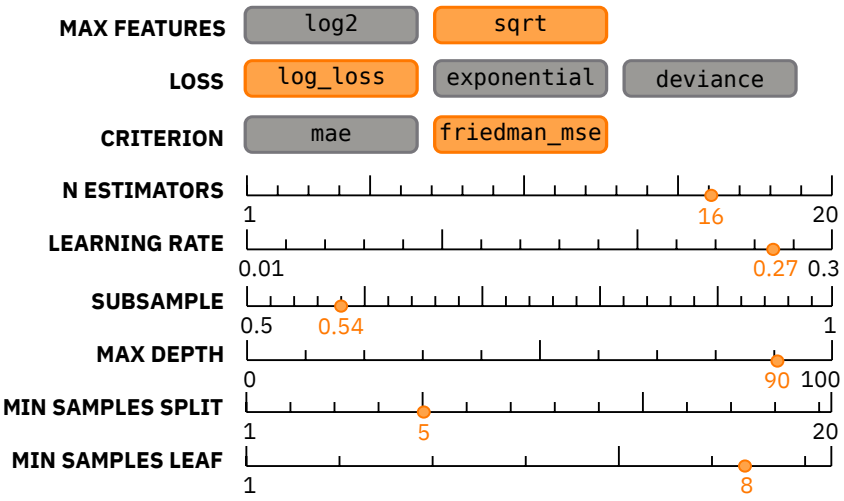


Figure 19: GradientBoosting parameters (best ones in orange)

	speech	song
precision	0.92	0.99
recall	0.99	0.93
F1-score	0.95	0.96
accuracy	**96%**	

Table 10: Classification report

Regression

The goal set for the regression task was to correct the (supposedly) erroneous values of two features, i.e. **`stft_min`** and **`sc_min`**, shown in Fig. 24 and Fig. 27. Both these attribute had a high number of zero values, probably because statistics were generated on audio traces containing zones of silence. This, in particular, meant that the associated quantiles (**`stft_q01`**, **`sc_q01`**, **`stft_q05`**, etc.) also contained many zero valued records, so **quantiles-related features that presented over 20% of null instances were excluded**. Here should be noticed that the quantiles features elimination was not restricted to the "stft" and "sc" related ones, because also some of the others presented this problem. The **non-zero values of each variable were log-scaled** as a preprocessing step.

After this, in order to reduce the computation time, a **feature selection** stage was carried out using the `SelectKBest` class from `sklearn`. A `mutual_info` scoring was preferred to other plainly linear ones (F_statistics, Pearson's correlation) to extend to non-linear relationships.

Figs. 20 and 21 show the chosen 10 best numerical features for both targets. Furthermore, **all categorical attributes were used** for the inference.

Results are compared in terms of the $\mathbf{R}^2$ coefficient and Normalized RMSE (**NRMSE**), defined as the ratio between RMSE and the variable spread, both computed on a random validation set having size 20% of the available non-erroneous data.

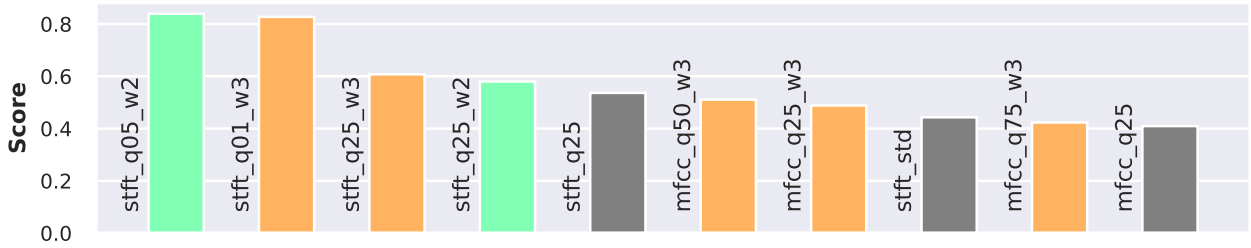


Figure 20: Best 10 features for `stft_min` based on `mutual_info_score`, colored by window. As it was already clear in Fig. 2, the most informative features are the ones contained in the two central windows. Since the target fature is the minimum stft, it is sensible that low quantiles appear at the first 5 positions.

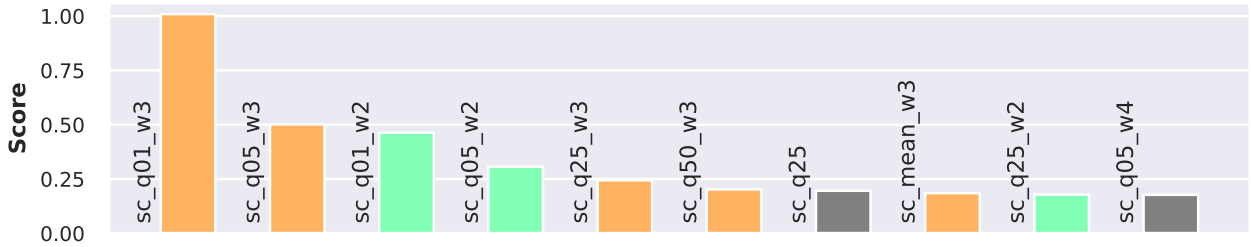


Figure 21: Best 10 features using `mutual_info_regression` as score function for `sc_min`. Again, low quantiles display an important mutual information with the target feature.

Regression on stft_min

To correct the values of **`stft_min`**, two different regression methods have been employed, i.e. `XGBoost` and `Neural Networks`.

For both models, the hyperparameter tuning was performed using `optuna`. In the case of `XGBoost`,

the tuned hyperparameters are shown in Fig. 22. Note that **`gblinear` is excluded from the tested options** for booster because, since it uses linear models, is useful only in case of a high number of features. The **dropout rate** for DART was set to the default value (0.1).

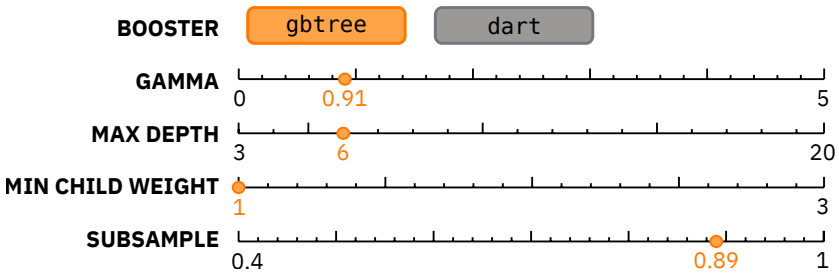


Figure 22: XGBoost parameters for **`stft_min`** regression

R^2	0.92
NRMSE	6%

Table 11: Model evaluation

Instead, for the `Neural Network`, these are the hyperparameters that have been tuned:

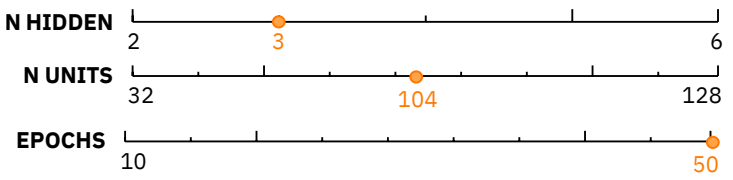


Figure 23: Neural network parameters for **`stft_min`** regression

R^2	0.75
NRMSE	10%

Table 12: Model evaluation

The metrics results are shown in Tab. 12 and Tab. 11. **XGBoost clearly outperforms the `Neural Network`**, allegedly due to a better use of categorical data.

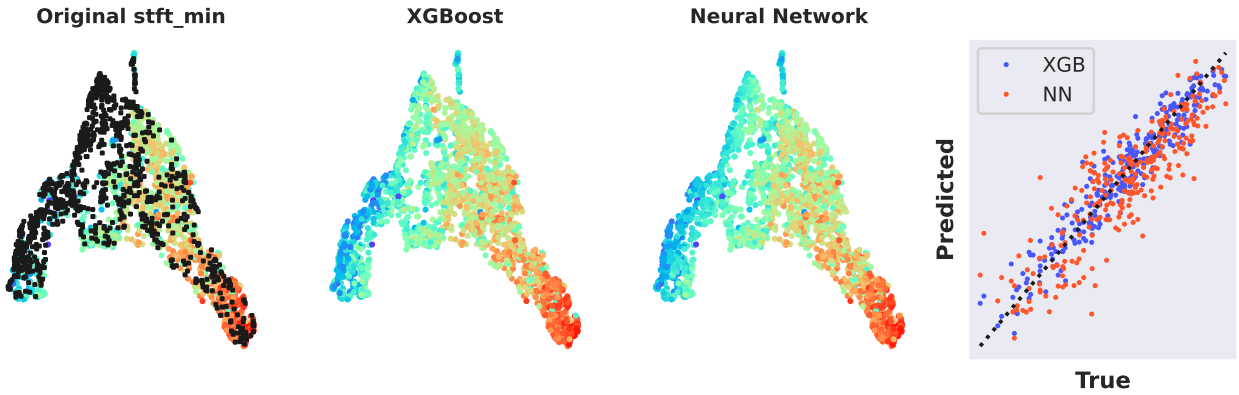


Figure 24: Results of regression for 'missing' values in `stft_min`, represented as the color of points inside an UMAP embedding for the best 10 features. Vaaues are log-scaled, black dots thus represent minus infinity. From the rightmost figure (comparison on validation set), XGBoost seems to only slightly overestimate the target values and not underestimate them, thus reducing the RMSE.

Regression on sc_min

Two models were also used to adjust the distribution of **sc_min**, i.e. Random Forest and, again, XGBoost. Once again, the best values for the hyperparameters were identified using optuna. The hyperparameters that have been tuned for XGBoost are shown in Fig. 25.

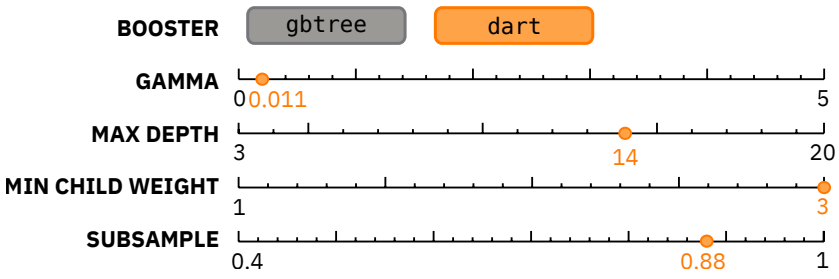


Figure 25: XGBoost parameters for **sc_min** regression

R^2	0.93
NRMSE	3.2%

Table 13: Model evaluation

The choice of DART (having a dropout rate of 0.1) as optimizer and 0.8 as training subsample for each tree indicates that **the model was at risk of overfitting** and these two parameters were chosen to increase the model's robustness. This fact may be due to the value of gamma: the values (in log-scale) of the target variable range between $[\approx 2, \approx 4]$, so the chosen value of **gamma allows splits that reduces the RMSE of about 0.5% the scale of the target variable**.

Meanwhile, the hyperparameters that have been tuned for Random Forest are displayed in Fig. 26.

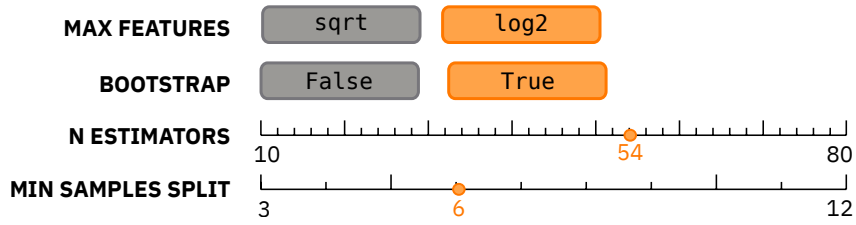


Figure 26: RandomForest parameters for **sc_min** regression

R^2	0.88
NRMSE	4%

Table 14: Model evaluation

Also the random forest approach seems to be exposed to overfitting issues: in fact, samples bootstrapping is chosen over only-features subsample selection. The results of the applied regression methods **show comparable results**, having an R^2 coefficient near 0.9.

Module 3

Time Series: Understanding & Preparation

Due to the high size of the files, traces are **downsampled** by a factor 8. Furthermore, since **silence is present** at the end and at the beginning of each trace, each track is cut at the first and last occurrences of a signal of intensity higher than 5% of the maximum value.

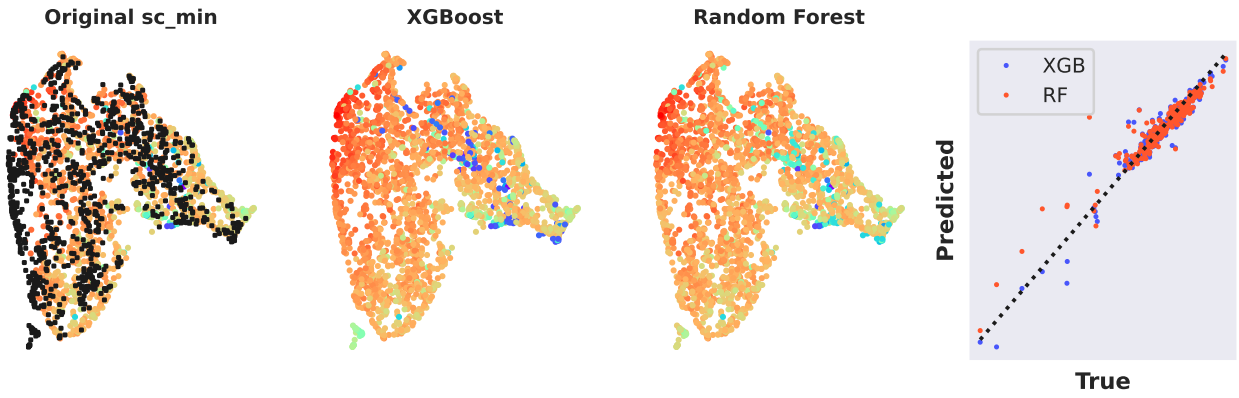


Figure 27: Again UMAP embedding of the 10 best features according to mutual information, `sc_min` values are displayed as the color of points. Both the employed regression methods give worse predictions for lower values of the target variable due to data scarcity.

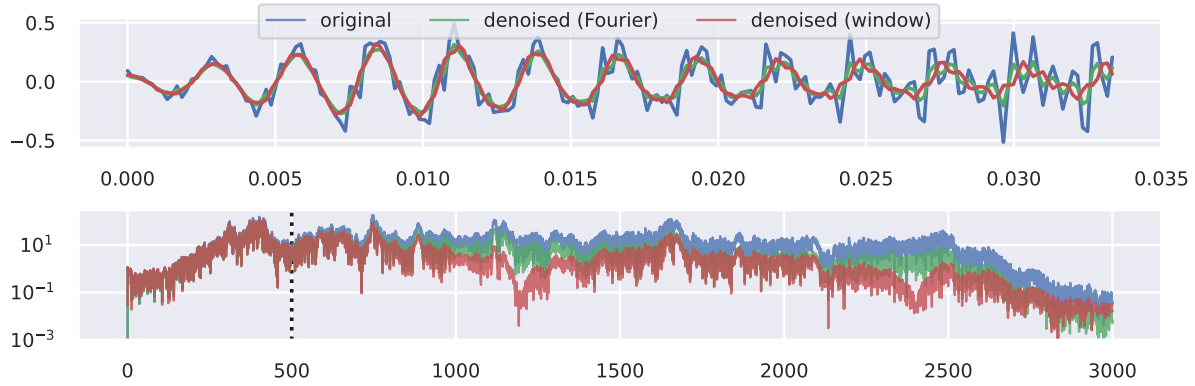


Figure 28: Denoising of the downsampled audio traces: (top) audio signal (time in milliseconds),(bottom) amplitude of the discrete Fourier transform for each denoising method (frequency in Hertz). Denoising using a rolling window introuduces further distortions to the spectrum of the audio signal due to the fact that the convolutional window is hard-edged. The cutting frequency is $500\,\mathrm{Hz}$.

The signal is then **detrended** (by subtraction of the best-fit line), **denoised** by applying a low-pass filter with a cut frequency of $f_0 = 500\,\mathrm{Hz}$ (Fig. 28) and then **normalized** to have overall standard deviation equal to one. The transfer function of the used low-pass filter is given by

$$G(\omega) = \frac{f_0}{\sqrt{f_0^2 + (\omega/2\pi)^2}}$$

implying that the higher is the pitch of a Fourier component, the higher is the amplitude reduction. The denoised trace is then obtained by:

$$y_{denoised}(t) = \mathcal{F}^{-1}[G(\omega) \cdot \mathcal{F}[y(t)]]$$

Due to the Nyquist-Shannon theorem and the applied downsampling, the **maximum representable frequency** amounts to $\frac{1}{8}24\,\mathrm{kHz} = 3\,\mathrm{kHz}$, while the denoising introduces a smooth cut for frequencies over $500\,\mathrm{Hz}$. This choice was justified by manual inspection of the recordings, the one denoised

with a low-pass filter being judged less distorted than the ones undergone rolling-window smoothing.

Finally, the coarse-grain properties of the trace from the point of view of sound intensity are computed from the **short-time intensity trace**, defined as:

$$I^2(t) = \frac{1}{T}\int_{t-T/2}^{t+T/2} \left(y_{denoised}(s) - \langle y_{denoised}(s)\rangle\right)^2 ds$$

i.e. the standard deviation of the signal on a centered moving window of width $T = 10\,\mathrm{m\,sec}$. The obtained array is then scaled in the [0, 1] interval and then standardised to a fixed length of 700 points for each record k by linear resampling:

$$STI_i^{(k)} = I_{m(i)}^{(k)} \qquad m(i) = \left\lfloor i \cdot \frac{\mathrm{len}(I^{(k)})}{700} \right\rfloor \qquad i \in [0, 700)$$

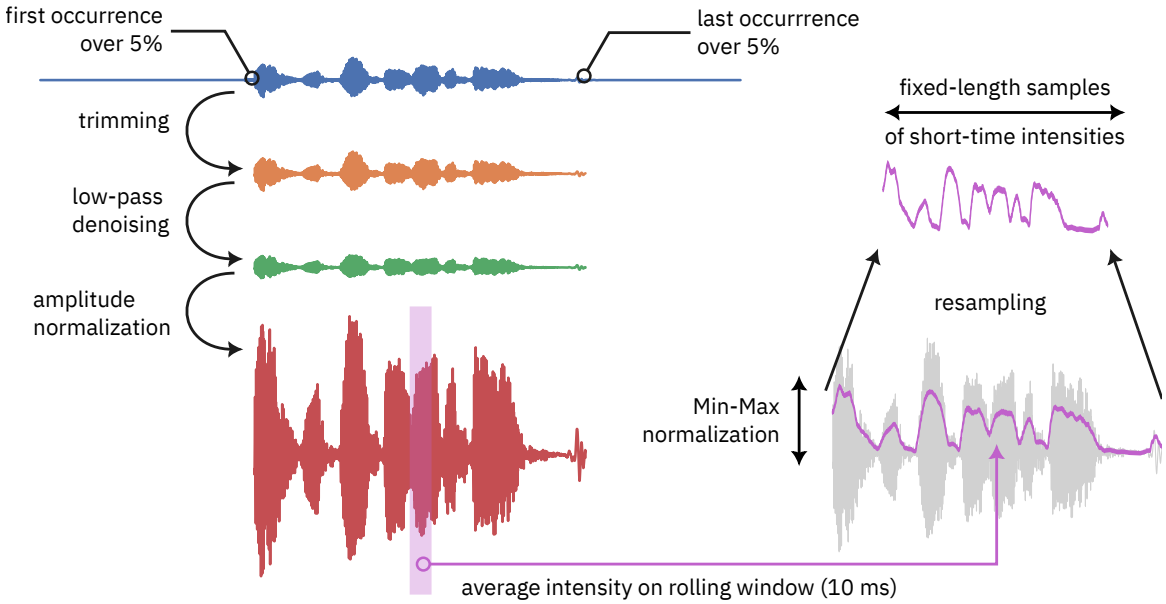


Figure 29: Preprocessing of the audio traces.

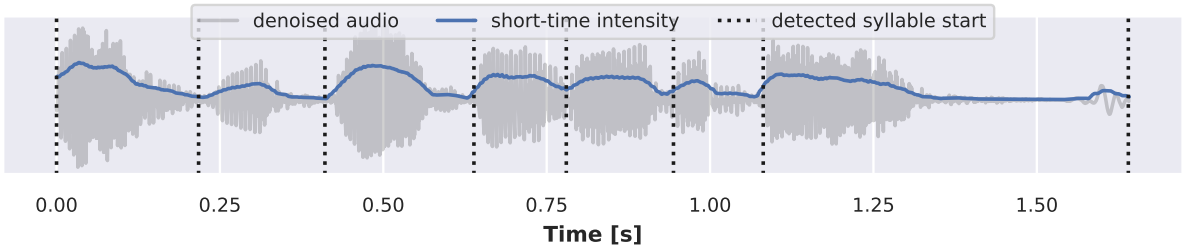
Segmentation

To improve performances in computing time and to get a more homogeneous collection of sub-datasets, each record is then split in syllables by analysis of the derivative of the short-time intensity.

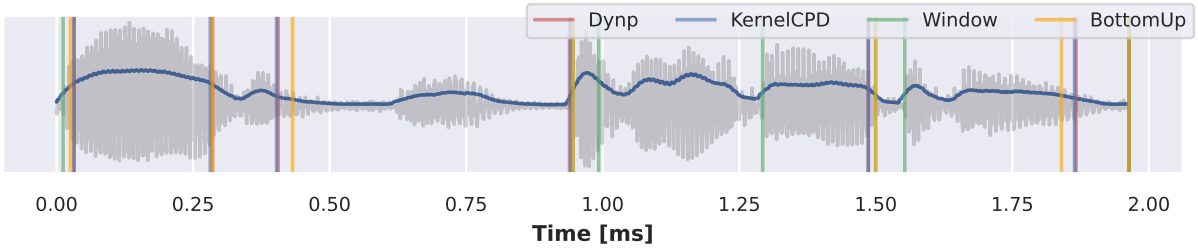
Since both sentences ("kids are talking by the door" and "dogs are sitting by the door") are mainly composed of seven syllables, i.e. time windows of relatively high intensity separated by short silences, the audio trace is split at seven points having the highest intensity derivative (Fig. 30a).

Short-time Fourier transform

Each audio timeseries is splitted in a fixed number of windows and for each window the Fourier transform is computed.



(a) Segmentation of the audio trace in seven syllables using derivative of short-time intensity.



(b) Comparison with alternative changepoint detection algorithms from the `rupture` python package with the same amount of breakpoints. Manual examination of the segments reveals that the tested algorithms appear to cut syllables in the middle.

Figure 30

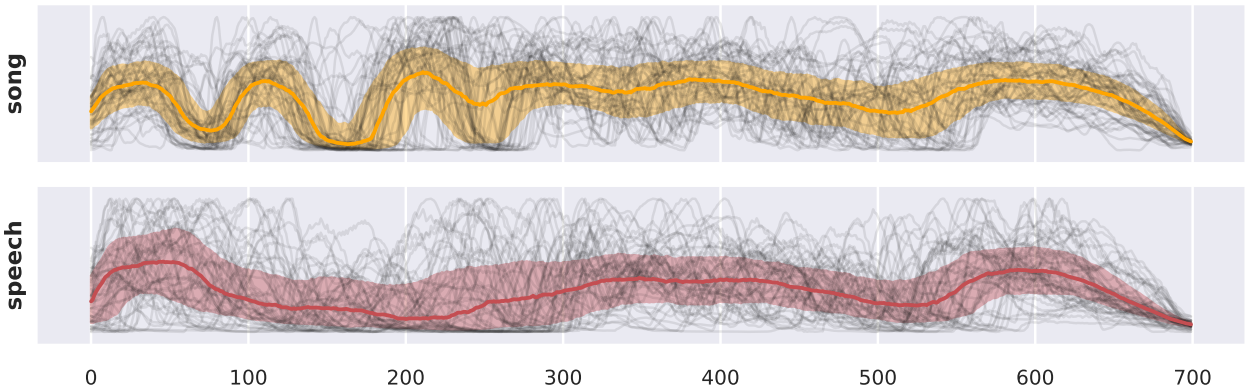
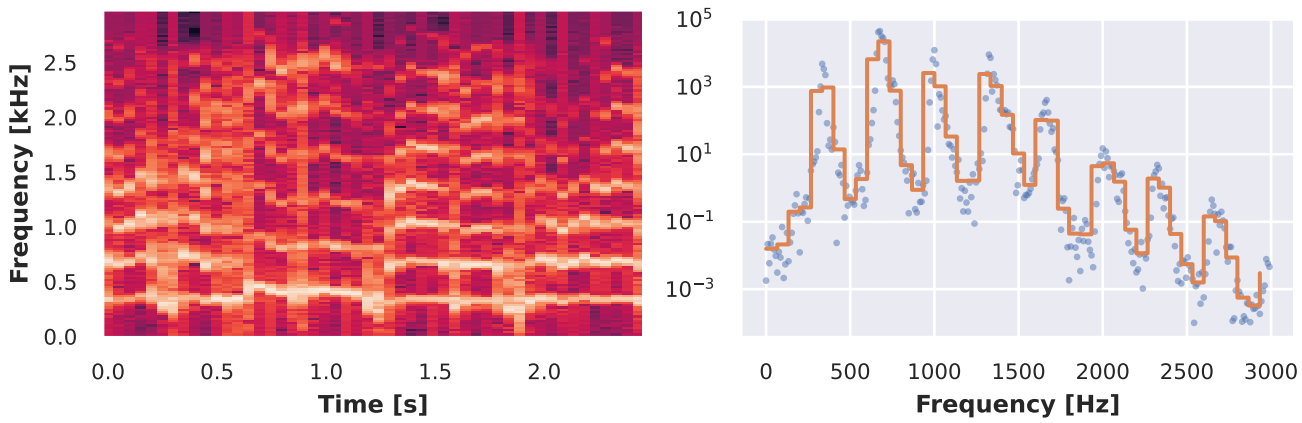


Figure 31: Average short time intensity divided by `vocal_channel`. Black lines are traces, solid line is the median while the band represents the time-dependent interquartile range. For the `song` vocal channel the first syllables (presumably "kids-are-ta" and "dogs-are-sit") are somewhat synchronized, while `speech` shows a wider variety of distribution of the single syllables that are averaged incoherently. For both, the starting sound (kids/dogs) and the last sound (door) are synchronized almost by definition due to the trimming process.



(a) **(b)**

Figure 32: (a) Short-time Fourier transform intensities (i.e. squared coefficients) for 50 time windows, resulting in 145 Fourier coefficients for each frame. (b) To account for varying spectral resolution, the coefficients for each window are divided into bins. The resultant coefficient is the average for each bin.

The number of Fourier coefficients is half the number of time samples contained in each time window, so **increasing the number of time windows decreases the resolution of the spectrum** (while mantaining the same maximum frequency). Since traces have different length, **the spectrum must be binned** to obtain an average energy for a given range of frequencies (Fig. 32).

If the shortest trace has length l, the maximum number of Fourier coefficients for one among the N_T windows is

$$N_F = \frac{1}{2}\frac{l}{N_T}$$

In order to get a balanced description of the time-frequency features of the audio, **the number of time windows and frequency bins are chosen to be equal**:

$$l = \min_{a \in \text{Audios}} len(a) = 5976$$

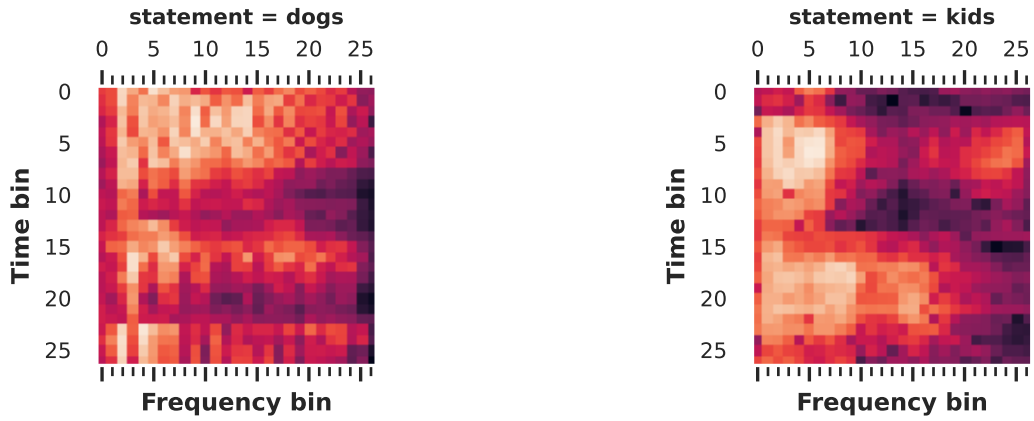
$$N_T = N_F = \lfloor \sqrt{\frac{l}{2}} \rfloor = 54$$

The method outlined above (that will be referred as **'balanced STFT'**) allows for frequency time series of identical length, albeit at the expense of spectral resolution. We also explored an **alternative strategy in which the number of spectral points to have in each frame was fixed**, but the length of the time series was left to vary.

For both methods, the **spectral centroid**(central frequency obtained by weighting frequency bins by intensity) and the **spectral mode** (the frequency of higher intensity) are computed too.

Statement classification and syllables 1 & 3

The first and third syllables (kids, talk- / dogs, sit-) are the ones containing more information about the statement being vocalized. For every record, the balanced STFT procedure exposed above is done for the **sub-recording containing only syllables 1 and 3** (Fig. 33), resulting in square matrices of dimension 27, and will be used for classification in the next sections.



(a) (b)

Figure 33: Balanced STFT for audios with just syllables 1 and 3. The contrast between the vocal "I" in kids and the vocal "A" in ta(lking) may be seen in the right image, which consists mostly of a dark/light patch in bins 5-15 for frequency.

Time Series: Motifs & Discords

Motifs and discords are analysed among both short-time intensities (Fig. 34) and denoised traces (Figs. 35 and 36).

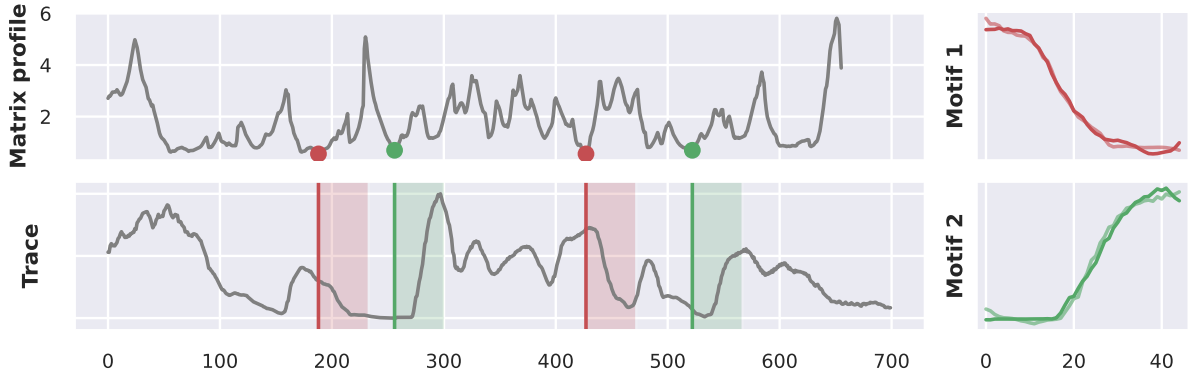


Figure 34: Motifs on short-time intensity traces for a 45 points window. Trivially, the top two motifs correspond to the beginning and the end of syllables for most traces.

Time Series: Clustering

The goal of time series clustering is to partition a set of time series into homogeneous clusters, where time series within the same cluster share similar temporal behaviors or patterns so the choice of distance measure plays a critical role in determining the similarity or dissimilarity between time series.

Clusters are outlined using **euclidean** metric. This choice is justified by three reasons: 1) dinamic time warping, redefining the time scale locally, somewhat **looses information about the rhythm and duration of audio features**, thus is judged less appropriate than the fixed-time comparison implied in the euclidean distance; 2) DTW distance computation is 2 to 3 orders of magnitude **slower** than euclidean metric; 3) many of the problems that DTW overcomes where mitigated in advance

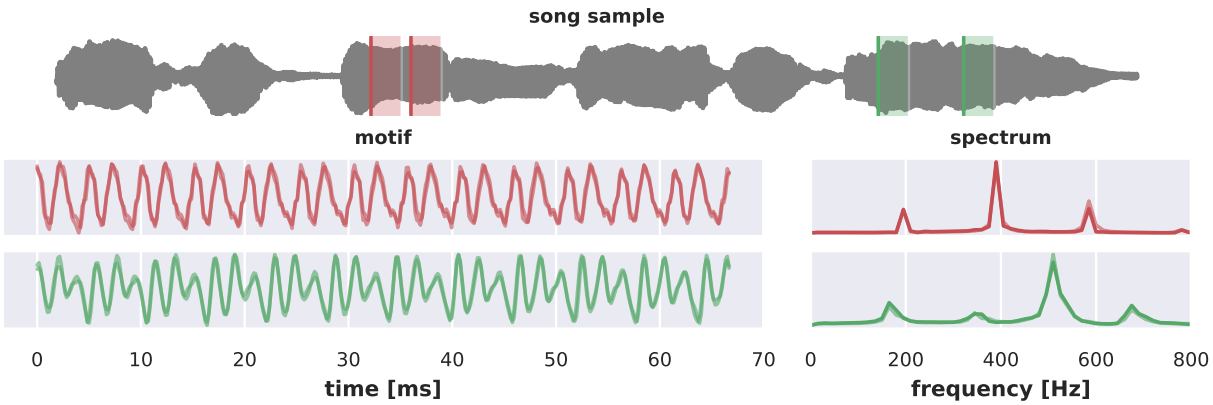


Figure 35: Top two motifs for a song audio trace. The exclusion zone of the matrix profile was set to one window, which is 400 points wide (66 m sec). Traces are almost perfectly overlapping, as well as Fourier coefficients. From the sustain of the sound it can be assessed that each highlighted window correspond to an open vowel, as can be roughly seen from the top trace of the complete audio. In this case the best matches are for the "a" in the first syllable of "talking" and for the "o" contained in "door".

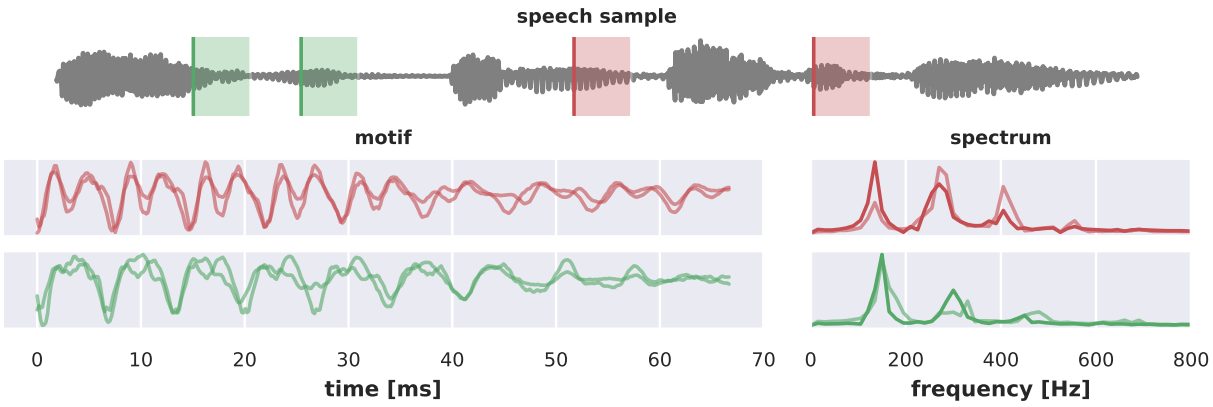


Figure 36: Top two patterns for a speech audio trace, using the same 66 m sec window (400 points). Unlike in Fig. 35, open vowels in speech samples have significantly lower sustain, therefore sound damping plays a significant role in the pattern. Even if the involved vowel is different, the detected motifs correspond to the ending of a syllable (the spectrum has slightly different peaks, but the damping tendency is extremely similar). In song records, increasing the size of the window produces the same outcome.

in the preprocessing stage (time binning, amplitude scaling, etc.) so **results showed negligible differences**.

In the next sections both shape-based clustering and feature-based clustering will be carried out on the **`short-time intensity (STI)`** and **`spectral centroid (SC)`**.

Shape-based clustering

To perform this task `TimeSeriesKMeans`[1] has been used, since it easily allows to compute the **silhouette score**. The latter was intended to be used as a guiding metric to understand the cluster's features. Results of its evaluation, together with adjusted mutual information, are reported in Fig.37.

The **silhouette score did not provide substantial assistance** in determining the optimal number of clusters. Therefore, a manual approach was employed to establish the number of clusters for the

[1] class belonging to the pypi package pyts

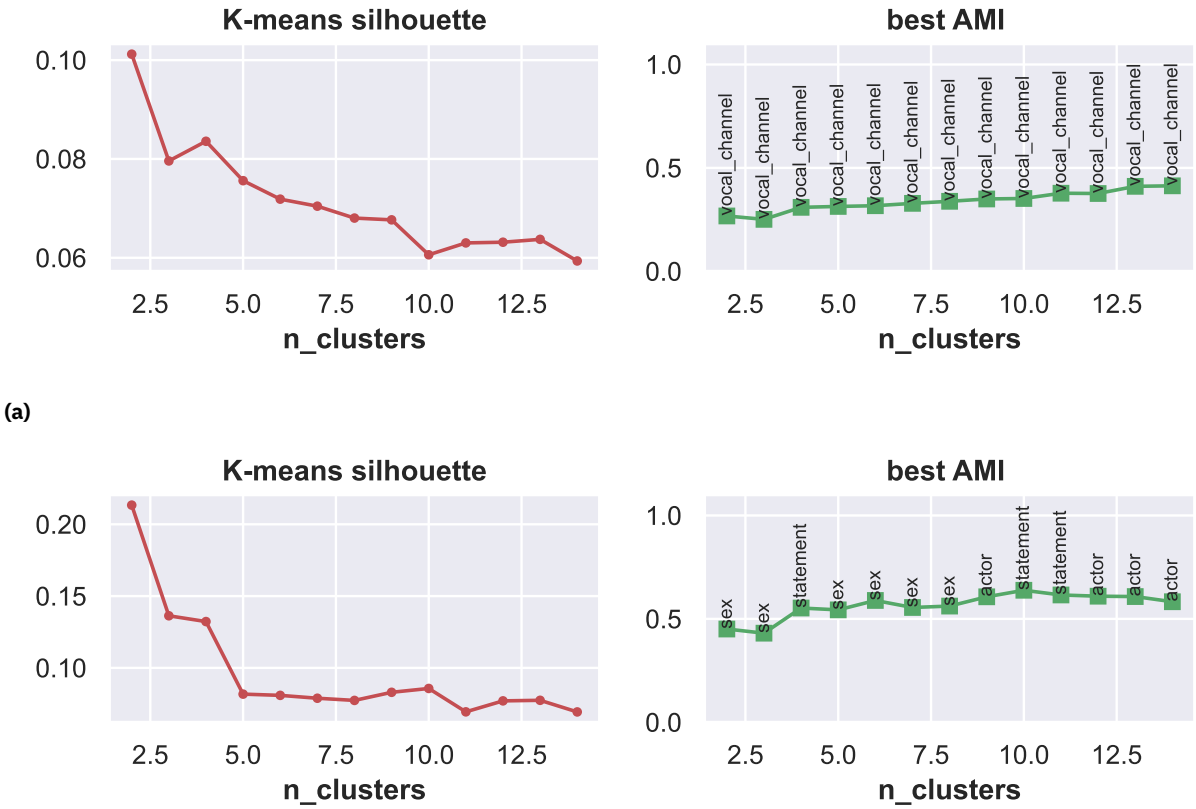
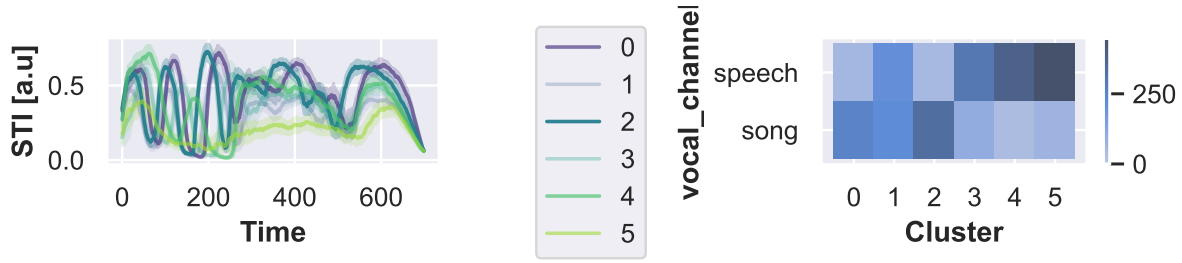


Figure 37: Mean silhouette score and sum of AMIs for different clusters number. AMI is summed over all categories, the best category is annotated. (a) short-time intensity, (b) spectral centroid

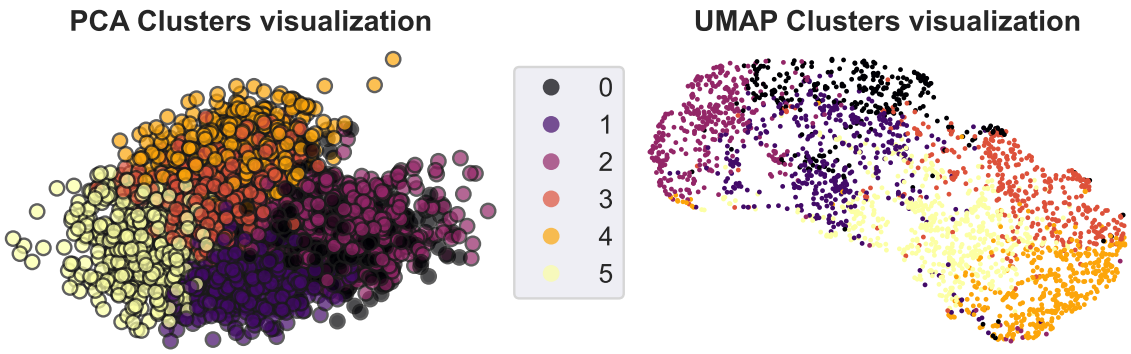
short-time intensity (STI) and spectral centroid (SC) traces: k=6 for STI traces and k=5 for SC traces.

For **exploratory purposes**, cross-occurrences for each categorical features are reported, highlighting the ones carrying more information together with the average cluster trace and a final PCA embedding.

From the results in Fig. 38 we can see how STI traces show a trend towards clusters that discriminate the two values of **`vocal_channel`**, whereas the mean traces of the clusters are similar to the two found in Fig. 31. The tendency of SC traces (Fig. 39) is to form distinct clusters for sex and statement. This phenomenon can be attributed, in part, to the difference in voice frequency range between males and females. Additionally, as discussed for Fig. 33, the clustering patterns are influenced by the characteristics of syllables. These observations are supported by the mean traces of the clusters, where the **mean traces for sex exhibit frequency gaps**, while the **mean traces for statement display peaks at different time bins**. For visualization, we employed Principal Component Analysis (PCA) with two dimensions. The results reveal that the SC traces exhibit distinct and dense clusters, whereas the STI traces exhibit four distinct sections with two clusters that appear to be less densely populated than the others.



(a)



(b)

Figure 38: Clusters using K-means algorithm. (a left) STI traces : the central line is the median trace of the cluster while bands represent the [40-60] quantile range, opacity is the AMI with the field `vocal channel`. (a right) Histogram of the cross-occurrences of the two labelling schemes. (b) Cluster visualization in 2 dimensions.

Features-based clustering

By utilizing the same set of time series data, the K-means algorithm is employed on a set of extracted features. This approach closely resembles the one employed in the clustering task for the initial part of the report, where the provided dataset predominantly comprised numerical features derived from the time series.

In this case, the following set of features has been utilized for extraction from each time series: minimum, maximum, mean, standard deviation, 10th, 25th, 50th, and 80th percentiles, as well as the number of peaks.

Silhouette score and AMI are plotted while varying the number of clusters in Fig.40.

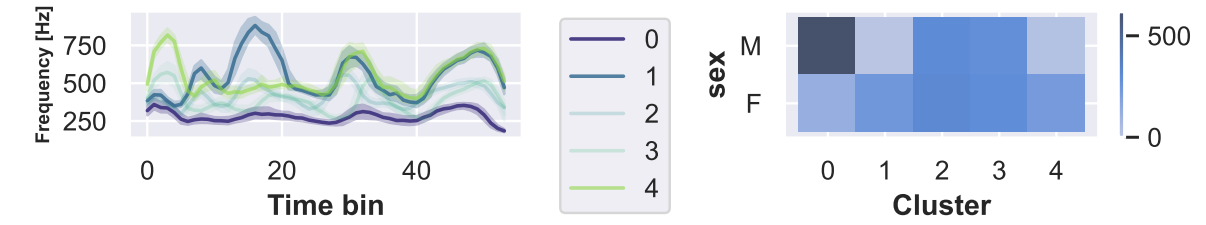
Once again, the analysis of silhouette score did not yield significant clues, so the number of clusters was set equal to 5 for both.

Results of the K-means clustering on both features is not reported since they provide the same remarks of the previous clustering approach, although with worse clustering performance.

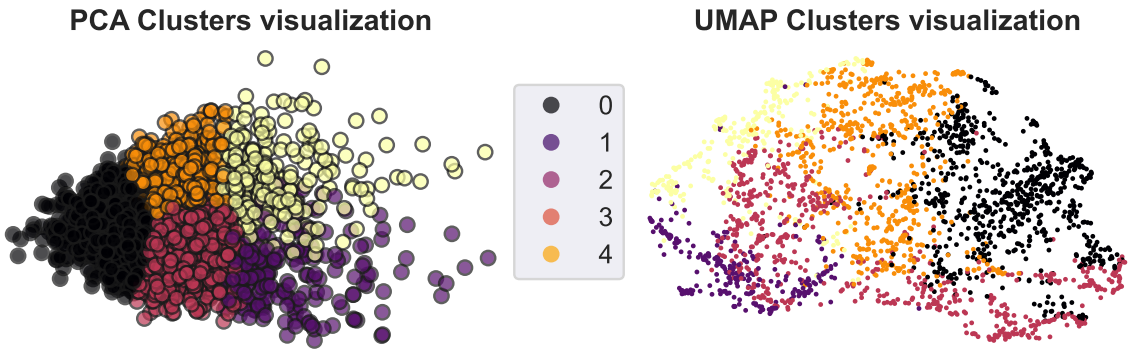
As a confirm, for STI traces the model fails to find informative clusters, since the mean traces of the clusters seems to be very similar to each other. Furthermore, from the clusters visualization of Fig.41(a) it is observable how the two clusters fill the space evenly.

Nevertheless, in the case of features extracted from SC traces, **the algorithm seems to detect clusters based on the different range of frequency and time bin of maxima**. However, it encounters

(a)

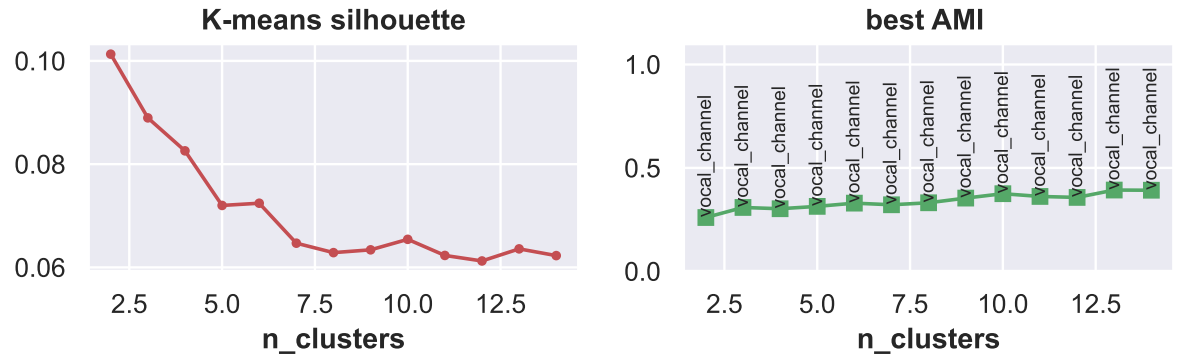


(b)

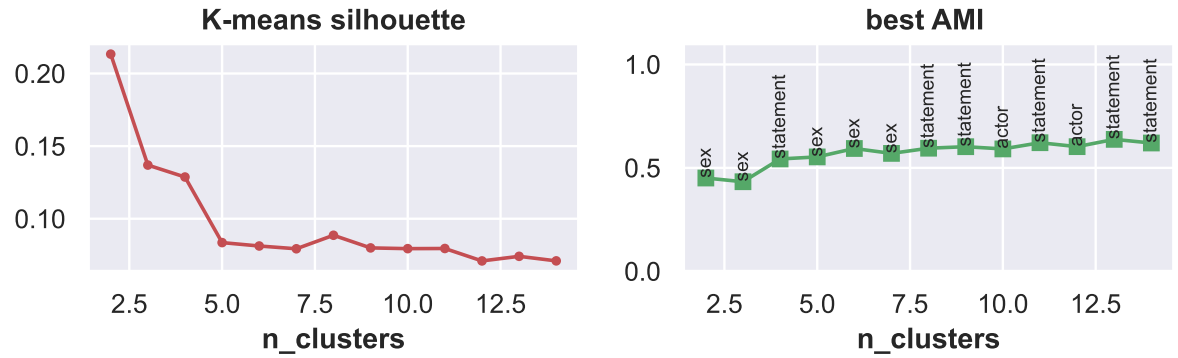


(c)

Figure 39: Spectral centroid clusters using K-means algorithm. (a-b left) Spectral centroid traces for balanced STFT: the central line is the median trace of the cluster while bands represent the interquartile range, opacity is the AMI with the field `statement`. The two clusters that best overlap the `statement` attribute are 1 and 4, displaying a notable difference in frequency around the time bin 15. This is probably due to the difference of spectral power distribution showed also in Fig. 33. (a-b right) Histogram of the cross-occurrences of the two labelling schemes. (c) Cluster visualization in 2 dimensions.



(a)



(b)

Figure 40: Silhouette for KMeans (a) on STI traces and (b) on SC traces for the balanced STFT of syllables 1&3.

greater difficulty in achieving well-separated areas among these clusters.

Time Series: Classification

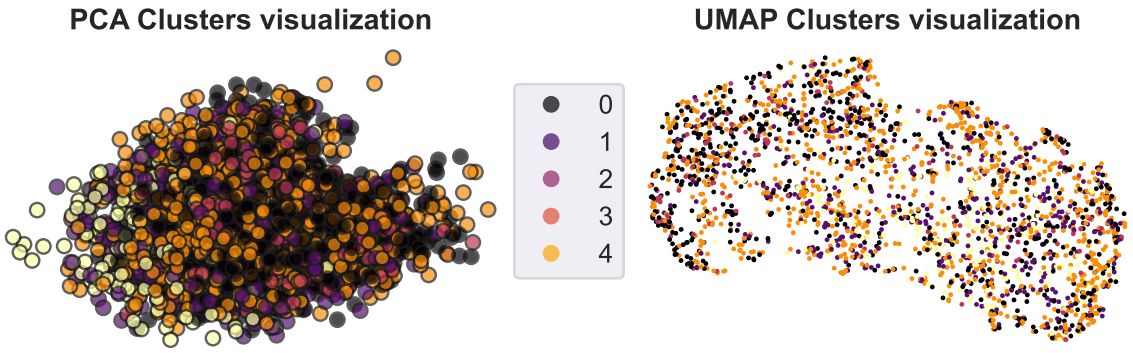
Statement

The applied approach on the syllables 1 and 3 proved to be quite effective for the statement categorization, as was previously apparent in the preceding section (Fig. 42).

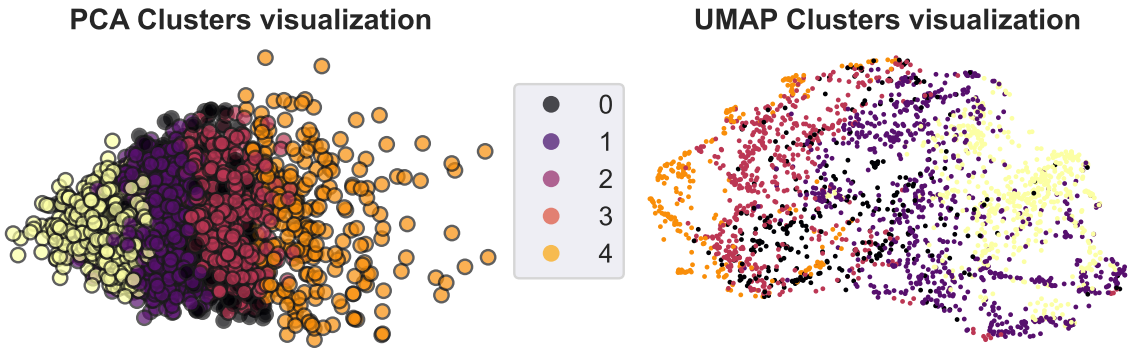
Three models were used for this task: `KNN`, `Shapelets` and a `1D Convolutional Neural Networks`. While the first two were mandatory, the choice to use a `CNN` comes from the idea of seeing the STFT transformation matrices of syllables 1 and 3 as images. Thus, it should be noted how for KNN and CNN a STFT transformation of the data was used, while for shapelets classification the spectral centroid was preferred. These methods will now be analyzed one at a time.

Tuning of hyperparameters for the `KNN` model was performed using **Optuna**. The values obtained for both the Euclidean and DTW cases are shown.

As expected by Fig. 42, the evaluation metrics for the KNN model returned some very good results for both types of distances. In fact, it was obtained an accuracy of 0.96314 in the case of Euclidean distance, and an accuracy of 0.96474 in the case of DTW. The classification reports for both distances are shown in Tab. 15 and Tab. 16.

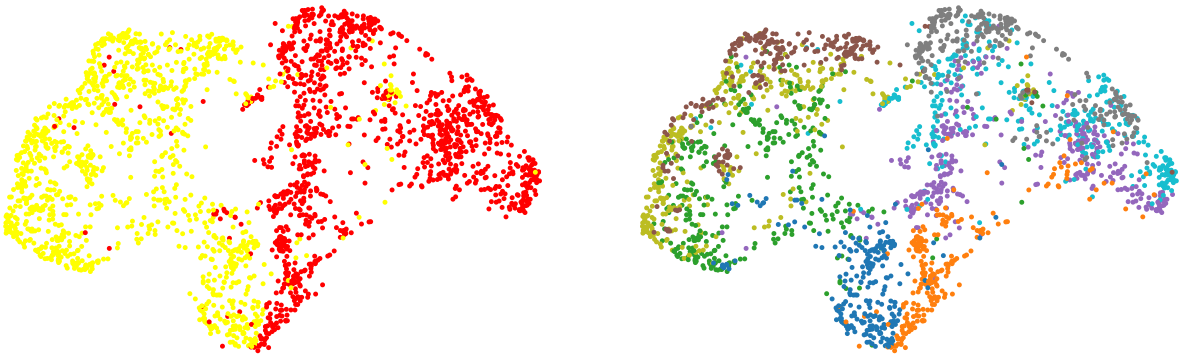


(a)



(b)

Figure 41: Cluster visualization of the extracted features of (a) short-time intensity traces and (b) spectral centroid traces.



(a) (b)

Figure 42: UMAP embedding (50 neighbours) for the balanced STFT of the joint audio for syllables 1 and 3 colored by `statement` (a) and product categories (`statement`, `sex`, `vocal_channel`) (b). The almost perfect separation of the two subgroups in the first figure is indicative of the fact that the chosen transformation was effective. Since UMAP works only on 1D arrays, each STFT matrix was flattened before the embedding was fit.

Table 15: Euclidean distance

	precision	recall	F1-score
dogs	0.97	0.96	0.96
kids	0.96	0.97	0.96

Table 16: DTW distance

	precision	recall	F1-score
dogs	0.97	0.96	0.96
kids	0.96	0.97	0.97

For the Shapelet classification the optimal shapelets have been found using the spectral centroid transformation of the dataset, and then a KNN model was used by passing it as input the distances between the traces and the shapelets.

Shapelets number and size were found using the `grabocka_params_to_shapelet_size_dict` function, passing it as parameters the training set shape (1828, 27), the number of classes (2) and the fraction of the traces length to be used (0.4). This resulted in 4 shapelets of length 10, which were extracted using `ShapeletModel`. The results are reported in Fig. 43.

After transforming the dataset to compute the traces' distances from the shapelets, KNN hyperparameters have been tuned, again, using **Optuna**. Results are reported in Tab. 17.

The model performed worse than expected: indeed, it couldn't find shapelets that best capture the syllables peaks. This may result from the small number of values with which traces were represented using the spectral centroid transformation (27). One possible way to improve the model could be to sample the trace in such a way that more points are obtained at the peaks, or to add weights to the track points as a model parameter in order to help it detect them more easily.

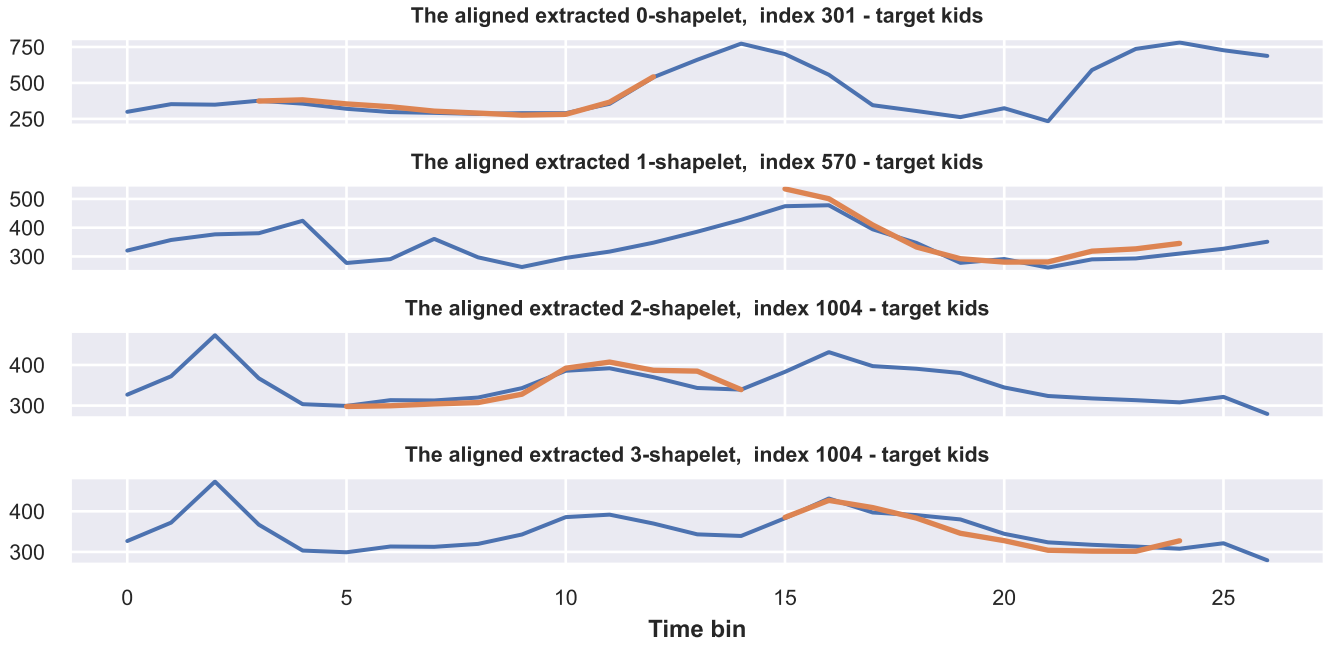


Figure 43: Shapelets found using ShapeletModel

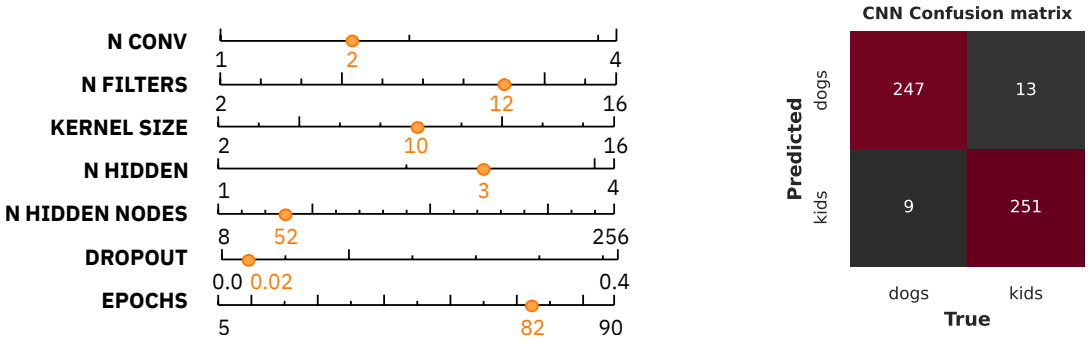
Finally, as mentioned above, the choice to use a 1D CNN comes from the idea of seeing the STFT transformation matrices of syllables 1 and 3 as images, like in Fig. 33a and Fig. 33b. Like for KNN, tuning of hyperparameters was performed using **Optuna**. The model achieved an accuracy of 98%. The classification report is shown in Tab. 18. Considerations on the classifications of the various images are deferred to the Explainability section.

Table 17: Classification report for Shapelets

	precision	recall	F1-score
dogs	0.69	0.79	0.74
kids	0.76	0.64	0.60
accuracy		**72%**	

Table 18: Classification report for the 1D Convolutional Neural Network

	precision	recall	F1-score
dogs	0.99	0.96	0.97
kids	0.96	0.99	0.98
accuracy		**96%**	



(a) **(b)**

Figure 44: (a) Best hyperparameters for the 1D convolutional neural network tested, (b) confusion matrix of the classifier for the target field `statement`.

Module 4

Since balanced STFTs can be interpreted as images and since a classification criterion (i.e. a difference of spectral power for some frequencies) appears to be present from Fig. 33 and Fig. 39, human-made expectations were compared to **LIME**[2] explanations for the balanced STFT of the syllables 1&3.

LIME, in case of images, generates surrogates in the input space by substituting superpixels with a 'fudged' version of them, in order to estimate their impact in the classification probability. The first step was defining the way superpixel are determined (Fig. 45). The default segmentation was inappropriate, so we tested both a simple square segmentation. Since the features of the STFT are often vertical lines, we also tested a random-offset rectangles (ROR) approach.

We exploit the fact that the used data has a standard format by averaging over 300 explanations (Fig. 47). Each explanation generates a number of surrogates in the neighborhood of a sample, then, using a strongly regularised ridge regression, LIME finds the best superpixels that justify the

[2] An open-source python package (with limited warranty) was used: https://github.com/marcotcr/lime/tree/master

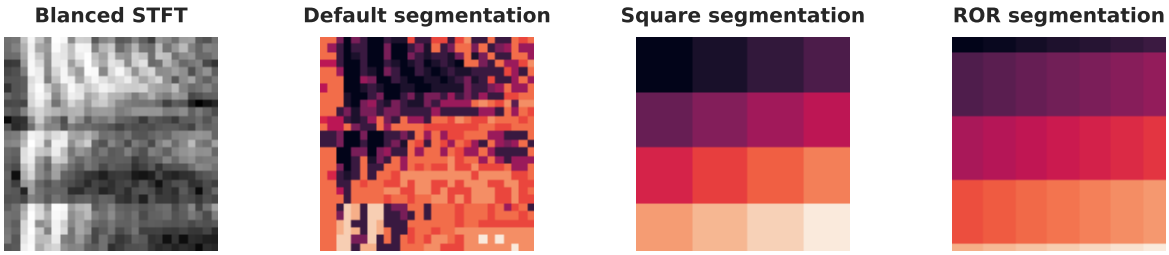


Figure 45: The default segmentation algorithm for images in the used LIME package (i.e. quickshift) is suitable for real-life images and determines superpixels by clustering parts of the image, while the chosen segmentation approach consists simple patches that correspond to (frequency, time) regions.

Figure 46: Some LIME explanations of the statement predictions. The first two figures display the results for a ROR (width 3, height 7) segmentation, while the last for the square (size 9) one. Features in upper left corner seems relevant to `kids`, while the upper right corner is important for `dogs`.

(locally linear) decision boundary. Averaging is plainly a pointless operation due to the local nature of this process, **unless the decision boundary is globally approximately linear in some feature**.

To test this hypothesis, both ridge and lasso linear regression were applied to predict the [0,1] label (1 if prediction was greater than 0.5, 0 otherwise), scoring an accuracy of 94%. The resulting weights (Fig. 48) are the global importance of each pixel and **roughly appear as the weighted sum of the local important features for both categories**.

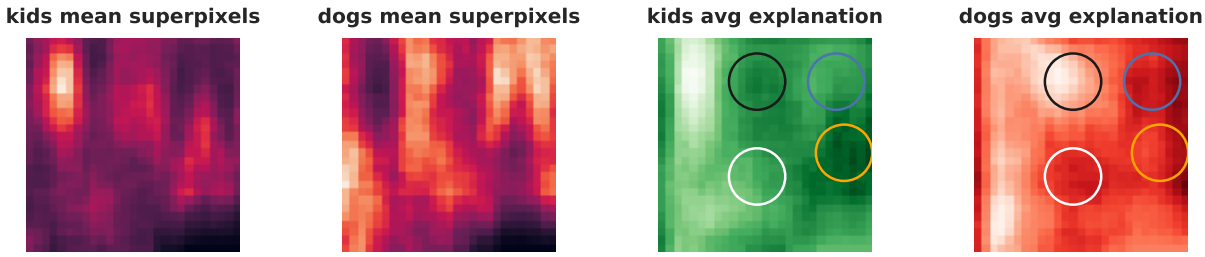


Figure 47: Average important superpixels and explaining features. Note that the figures looks much smoother than the ones in Fig. 45 because of the random offset of the ROR segmentation. (left two figures) The explaining superpixels are averaged for the set of selected 100 images, showing that explaining features have a specific time-frequency distribution. Since the model is a binary classifier with a 97% accuracy, it is sensible that each image appears approximately as the negative of the other, except for regions that are not of interest (lower right corner, the end of the trace). (right two images) Average of the features contained in the explaining superpixels. The three circles highlight time-frequency regions that mostly discriminates between the two categories.

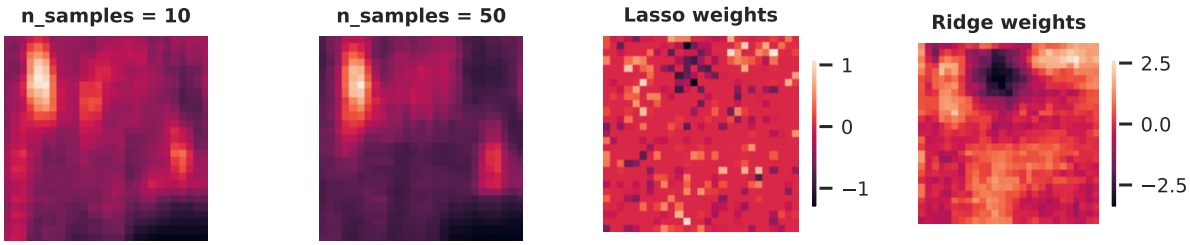


Figure 48: (left two figures) Increasing the number of LIME samples slightly improves the resolution in feature importance. Both images display the average superpixels that justify the class "kids". (right two images) Global weights of each pixel for strongly regularised lasso and ridge regressors, both having an accuracy of 94%. For the case of ridge, in particular, can be noted that the single pixel importance appears as a mixture of the average explaining feature of each class (first two of Fig. 47), in which features explaining the negative class are negatively weighted.