University of Pisa
Department of Computer Science
M.Sc. in Data Science and Business Informatics

Data Mining II project report

The Ryerson Audio-Visual Database of Emotional Speech and Song Features (RAVDESS - Features)

Authors

Pierfrancesco Benincasa

Francesco Giacomo Curcio

Niccolò Seghieri

Tutors

Prof. Riccardo Guidotti

Dr. Francesco Spinnato

June 6, 2023

Contents

List of Figures

List of Tables

Introduction

The RAVDESS - Features dataset contains audio of 24 professional actors (12 female, 12 male), vocalizing two statements in a neutral North American accent. Speech includes calm, happy, sad, angry, fearful, surprise, and disgust expressions, and song contains calm, happy, sad, angry, and fearful emotions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression.

1 Data understanding & preparation

In a first step of this analysis, dataset refers to the training set made up by 1828 rows and 434 columns. In order not to compromise the impartiality of the results, in this first part, only the training set is taken into account. There are no missing information, so there is no need to implement any correction based on substituting or removing values from any record. Only an exclusion was made at a certain point (and it was temporary for dimensionality reduction's purpose): columns which have object or string data types were eliminated in order to implement numerical methods easily (e.g. variance threshold).

1.1 Data semantics

This dataset was created from RAVDESS, extracting basic statistics from the original audio data and after transforming it using: differencing, zero-crossing rate, Mel-Frequency Cepstral Coefficients, spectral centroid, and the stft chromagram. Features were extracted from the 2452 wav files. Features are extracted also by dividing each time series into 4 non overlapping windows. Features are divided into:

- **modality** *(object)*: represents in our case only vocal audio;
- **vocal channel** *(object)*: represents binary speech or song;
- **emotion** *(object)* explains of what kind of tone is the data (speech includes calm, happy, sad, angry, fearful, surprise, and disgust expressions, and song contains calm, happy, sad, angry, and fearful emotions);
- **emotional intensity** *(object)* is the same for both of vocal channel and also in this case the description is above;
- **statement** *(object)* showed the two in the dataset;
- **repetition** *(object)* for each statement stands for statement times reproduced;
- **actor** *(float)* indicates the number referred to the specific actor (each one if enumerated with a specific);

- **sex** *(object)* refers to male/female binary;
- **filename** *(str)* name of the corresponding audio file;
- **frame count** *(float)* the number of frames from the sample.

From each audio waveform the following features are extracted: sum, mean, std, min, max, q01, q05, q25, q50, q75, q95, q99 (0.01, 0.05 quantiles and so on), kur, skew (kurtosis and skewness, respectively). Each audio file was then transformed using:

- **lag1**: difference between each observation and the precedent;
- **zc**: zero crossing rate;
- **mfcc**: Mel-Frequency Cepstral Coefficients;
- **sc**: spectral centroid;
- **stft**: stft chromagram.

For each of these transform the same features are extracted as in previous lines (sum, mean, std, etc..).

There's also a naming schema in which features are extracted first at a global level (i.e. for the entire signal), then dividing into 4 equally sized windows. Windows are indicated with the string w1 or w2 or w3 or w4.

2 Simple classification

Before the application of specific kind of methods to conduce our analysis, for supervised classification is used Decision Tree classifier, implemented in this section in order to create a sort of baseline for the comparison with more complex approaches, that will be taken in account later.

2.1 Decision Tree Classifier

Only training set is taken as a reference and used from this phase 'til will be specified in a second moment. Globally, considering the large amount of columns in the dataset, expected performances are in line with expectations. To give an idea of which is the situation at this starting point, next figure will assess results usfeul to the reader to understand the amount of improvements of further steps.

No techniques for dimensionality reduction or transformation of the data of any kind were used before fitting the model. Parameters chosen, based on the analysis of a cross-validation grid search's output.

	Precision	Recall	F1 - score	Support
angry	0,420	0,526	0,467	95
calm	0,440	0,640	0,522	75
disgust	0,357	0,341	0,349	44
fearful	0,333	0,344	0,338	64
happy	0,343	0,407	0,372	86
neutral	0,391	0,176	0,243	51
sad	0,270	0,205	0,233	83
surprised	0,400	0,196	0,263	51
accuracy			0,375	549
macro avg	0,369	0,354	0,360	549
weighted avg	0,368	0,375	0,348	549

Table 1: An overview of results obtained with decision-tree classifier. All emotions are distributed in a validation set made up by 30% of training set values.

3 Anomaly detection

In this part, our goal is to discover the top 1% of records that have the highest outlier score as they may distort the results of some clustering and classification algorithms. To do this, we used different types of algorithms from different families: ABOD, isolation forest, LODA, KNN and LOF. We used 5 of them in order to have different approaches and mitigate the strengths and weaknesses of each one: in particular, first 3 were chosen because they better manage the problem of high dimensionality of the data while the second 2 may have some problems due to the fact they use distances and densities. Dataset used in this section is the original one with all 434 attributes.

3.1 Main results in anomaly detection

Below, there are the main parameters selected for each method with relative results:

- ABOD: we considered 15 neighbours for each point and 180 outliers were found;
- Isolation Forest: the number of base estimators is 100 and 93 outliers found;
- LODA: the number of bins is 10 and random cuts is 100 and 183 outliers found;
- KNN: we considered 5 neighbours for each point and 162 outliers found;
- LOF: we considered 5 neighbours for each point and 169 outliers found.

Given the nature of this unsupervised task, and the lack of measures of goodness, parameters selected for each method are often the default ones or differ a little from them. However, below is reported visualizations obtained through PCA to see in which region fall most of them in order to assess the quality of the outputs. We can see that there is a large cloud of inliers (coloured in blue) and that the outliers (in red) are located in the northeast region of the graphs. In addition, from the LOF graph we can observe how many outliers in the graph are confused with the inliers respect to the

other methods: this could be due to the fact that the algorithm is based on the concept of density and in an environment with many dimensions this can be meaningless.

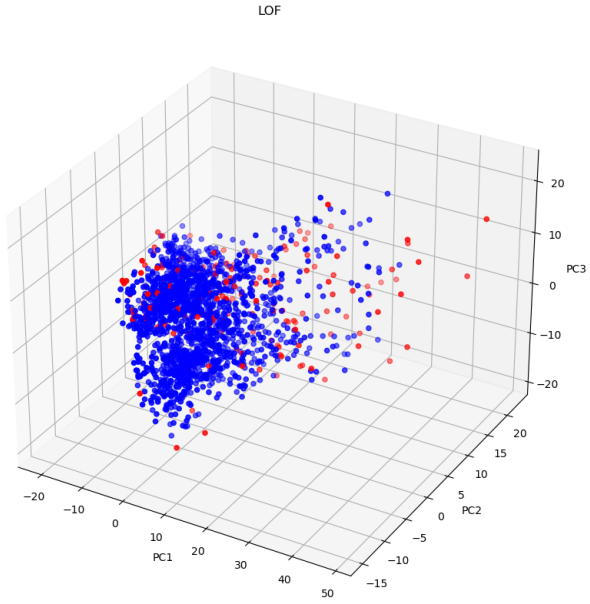
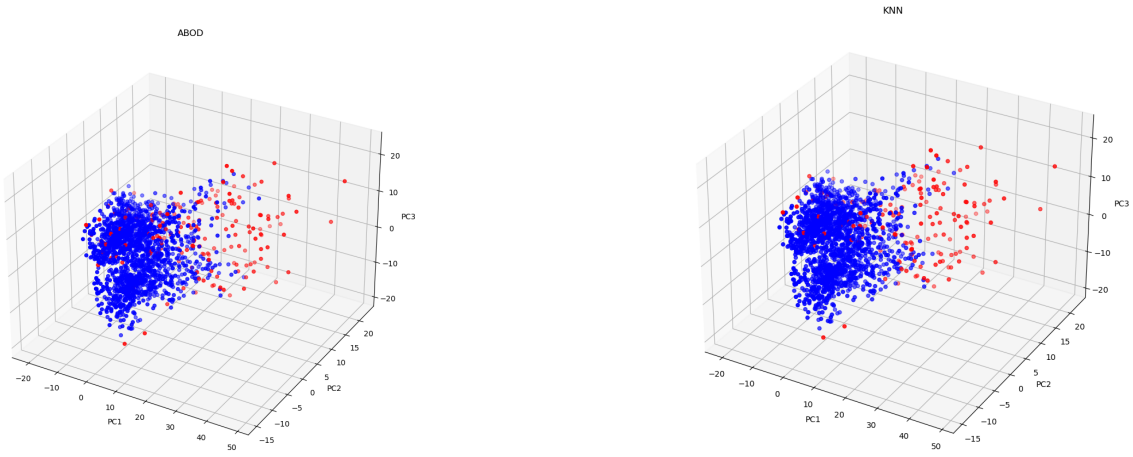


Figure 1: LOF - Principal component analysis. In red, outliers.



(a) (b)

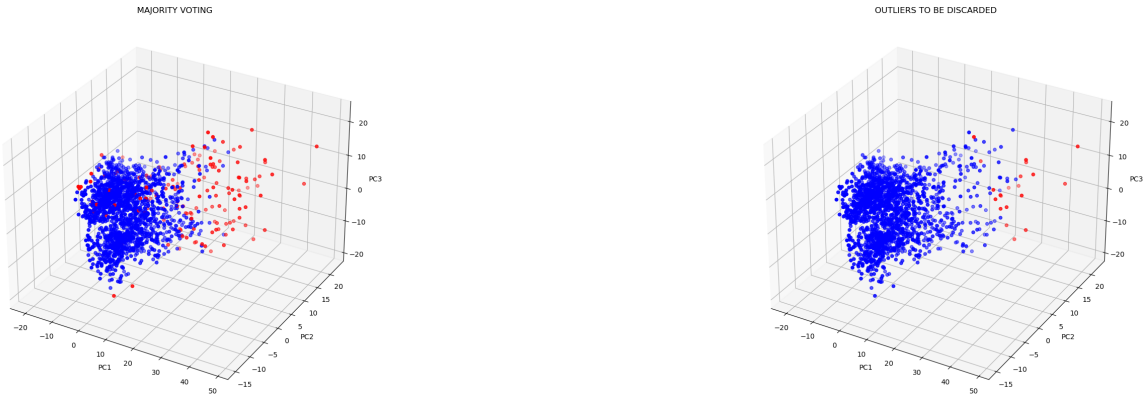
Figure 2: 3D anomaly detection algorithms results. (a) ABOD, (b) KNN

(a) (b)

Figure 3: 3D anomaly detection algorithms results - bis. (a) Isolation forest, (b) LODA

3.2 Results assessment

In order to select the top 1% of records that have the highest outlier score, i.e. around 20 observations, we opted for a simple majority voting strategy: if at least 3 out of 5 methods classify the point as an outlier then it is probably anomalous. We preferred this approach over an intersection of results because this would have been much more stringent and would therefore have given a decisive impact even for the weakest algorithms. We therefore obtained 153 points classified as outliers using this method. Then, in order to select 20 of them, we decided to sort the outliers using the isolation forest anomaly score as it does not have any weaknesses relating to high dimensionality and is therefore considered quite reliable.



(a) (b)

Figure 4: Majority voting results: (a) all points, (b) top 1% outliers

4 Imbalanced learning

In this section, our goal is to solve a classification problem with unbalanced classes. However, our dataset is fairly balanced for all categorical variables, so we decided to artificially unbalance the vocal_channel class from Speech: 1071, Song: 737 to Speech: 1071, Song: 74 in order to create a 93.5% - 6.5% imbalanced problem.

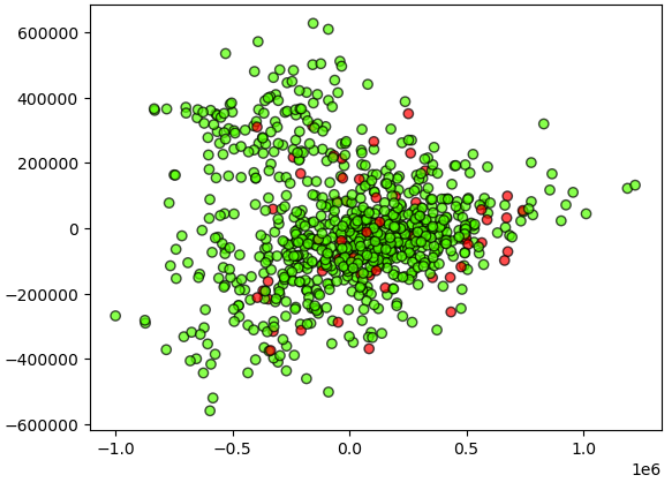


Figure 5: PCA on unbalanced data. In green, dominant class (speech), in red minority class (song).

Dataset used in this section doesn't contain the 20 outliers eliminated earlier and, moreover, it has only 207 attributes out of the 434 as we set a variance threshold of 10% by eliminating those that did not reach it. The classifier chosen was the Decision Tree and before implementing balancing algorithms we ran the algorithm finding as best parameters (via grid-search) max_depth: 8, min_samples_leaf: 5, min_samples_split: 11. Applying the model we obtained an accuracy of 97%, thus better than a dummy classifier that only predicts the dominant class having an accuracy of 93.5%. Despite this, as can be seen from table 2 in the classification report, song is more difficult to classify, having a recall of 58% and an overall F1 - score of 69 wrt a speech of 98%.

4.1 Undersampling

Results obtained with Undersampling techniques were not very good, resulting in a deterioration of performance compared to the model obtained with the unbalanced classes. In particular we report:

- CNN returned Speech: 94, Song: 55 partially rebalancing dataset. However, the accuracy of the model is 93% (similar to the dummy classifier), the precision for Song

drops to 50% while f1 - score drops by only 4 percentage points due to the increase in recall to 95%.

- Random Undersampling returned Speech: 55, Song: 55 rebalancing the dataset. In this case the accuracy is even lower, dropping to 85%, and there is a noticeable deterioration in identifying Song.

4.2 Oversampling

Conversely, oversampling techniques improved performance compared to the model obtained with unbalanced classes.

- Random Oversampling returned Speech: 803, Song: 803 rebalancing the dataset. This technique gives an accuracy equal to the model obtained on the unbalanced data but improves the f-1 score due to a recall increased from 58% to 68%.

- Smote returned Speech: 803, Song: 803 rebalancing the dataset. The accuracy is just below that of the model built on the unbalanced dataset but there is an increase in the f1 - score to 72% due to improved precision.

- Adasyn returned Speech: 1071, Song: 1066 practically rebalancing the dataset. Applying this algorithm gives the best results on the model obtained: accuracy rises to 99% and above all Song has a recall of 100% without having problems with precision, which reach 90% with only two errors.

4.3 Results assessment

Models obtained on rebalanced data using undersampling techniques give us rather poor results and suggest that the best configuration would be to remain with the model obtained on unbalanced dataset. The reason could be that train on so few instances may not be sufficient to capture all the patterns in order to build a good model. On the other hand, oversampling algorithms seem to solve such problems by obtaining good results despite the fact that at a graphical level it may appear that the structure created invade the decision boundary of the dominant class (for Adasyn and Smote). However, in our case the creation of synthetic data brings benefits by guaranteeing good performance.

5 Dimensionality reduction

Dimensionality reduction simply refers to the process of reducing the number of attributes in a dataset while keeping as much of the variation of the original one. It

	Precision	Recall	F1 - score	Accuracy
Unbalanced data	0,850	0,580	0,690	0,970
CNN Undersampling	0,500	0,950	0,650	0,930
Random Undersampling	0,280	0,790	0,410	0,850
SMOTE Oversampling	0,700	0,740	0,720	0,960
Adasyn Oversampling	0,900	1,00	0,950	0,990
Random Oversampling	0,810	0,680	0,740	0,970

Table 2: Imbalanced learning performances wrt to "song" in vocal_channel and overall accuracy.

leads to several advantages, such as: less training time and less computational resources whilst maintaining or increasing the overall performance of machine learning algorithms, avoiding the problem of overfitting, removing noise in the data. There are mainly two types of dimensionality reduction techniques: **feature selection** and **feature projection**. Feature Selection methods only keep the most important features in a dataset and remove the redundant ones. There is no transformation applied. On the other hand, Feature Projection methods transform the data in the high-dimensional space to a space of fewer dimensions.

5.1 Feature-Selection methods

5.1.1 Variance Threshold

The Variance Threshold technique removes all features with a training-set variance lower than some threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all samples. The threshold chosen for this project was 0.2, resulting in 190 variables instead of 426 (that is the number of columns eliminating data object or string from the original one with 434).

5.1.2 Univariate Feature Selection

Univariate feature selection works by selecting the best features based on univariate statistical tests. Each feature is compared to the target variable, to see whether there is any statistically significant relationship between them. It is also called analysis of variance (ANOVA). The number of features must be selected a-priori. The parameter k set for making a test is up to 10 variables in this case.

5.1.3 Recursive Feature Elimination

Recursive feature elimination attempts to eliminate dependencies that exist in the model. Features are ranked by recursively eliminating a small number of features per loop. Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), RFE selects features by recursively considering smaller and smaller

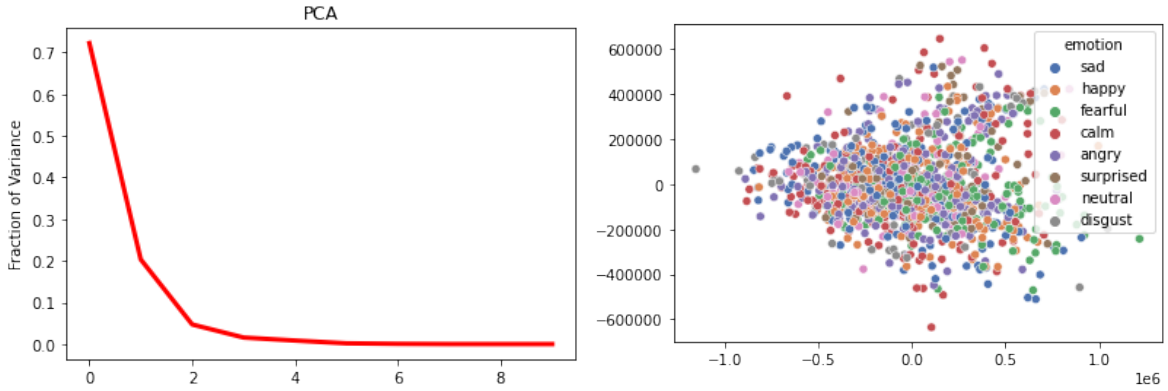
sets of features. By considering a Decision Tree classifier, as model, this technique returns 69 variables.

5.2 Feature-projection methods

Just to give an idea on how data could be sparse in space, in this subsection a 2D representation of methods is give. Numerical and weighted results, will be included in the final summarizing table.

5.2.1 PCA - Principal Components Analysis

Scope of PCA is to find a small number of linear combinations of the observed variables which explain most of the variation in the data. Given 426 variables, is possible to catch more than 96% of variance with only 2 of them, quasi-99% with 3.



(a) (b)

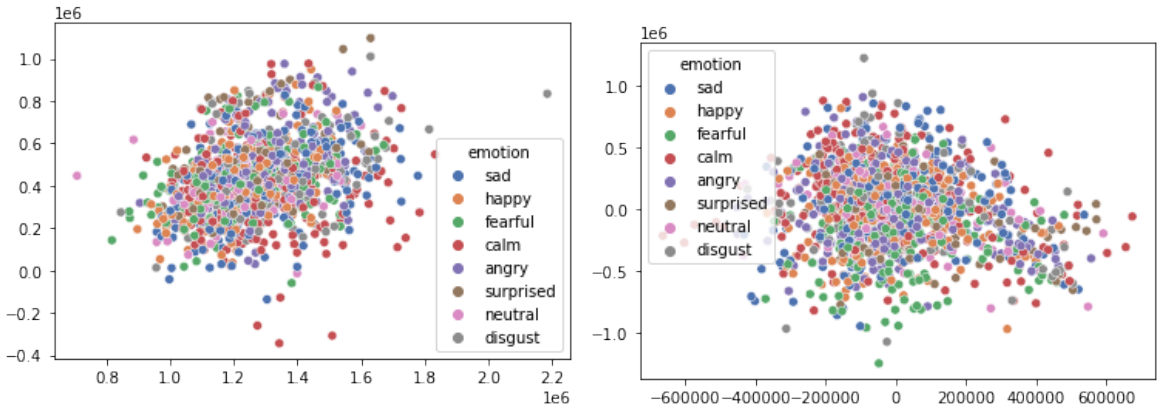
Figure 6: Variables-variance relationship (a) and PCA-2D representation (b).

5.2.2 Random Subspace Projection

High-dimensional data is projected with this technique into low-dimensional space using a random matrix whose columns have unit length. No attempt to optimize criterion is made, structure of data (e.g. distances) is preserved and it represents a computationally cheap method.

5.2.3 Multi-dimensional Scaling

Given a pairwise dissimilarity matrix (no need to be a metric), the goal of MDS is to learn a mapping of data into a lower dimensionality such that the relative distances are preserved. If two points are close in the feature space, it should be close in the latent factor space. K used to define in this case the structure of method is 2.



(a) Random Subspace Projection (b) Multi-dimensional Scaling

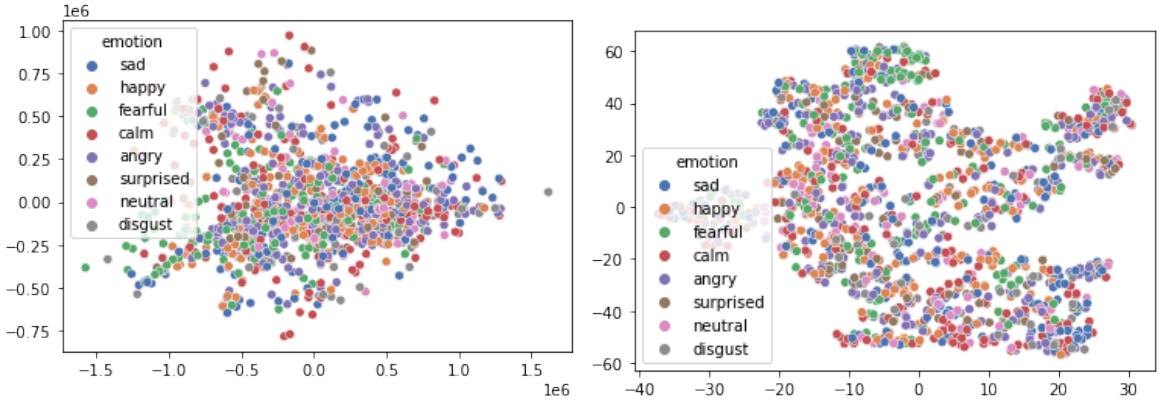
Figure 7: RsP and MdS visualization in 2D.

5.2.4 IsoMap

Preserves the intrinsic geometry of the data, uses the geodesic manifold distances between all pairs, it is a MDS method and handles non-linear manifold.

5.2.5 T-SNE

This method measure pairwise similarities between high-dimensional and low dimensional objects.T-SNE tries to preserve local structure: in fact local dimensional neighborhood should be the same as original neighborhood. Same is done for distance and neighbor preservation, and unlike PCA is almost only used for visualization.



(a) Isomap (b) tSNE

Figure 8: Isomap and tSNE visualization in 2D.

	Precision	Recall	F1 - score	Accuracy
Feature Selection Methods				
Univariate feature selection	0,357	0,342	0,343	0,302
Select from model	0,338	0,342	0,328	0,346
Recursive feature elimination	0,317	0,317	0,309	0,317
Feature Projection Methods				
PCA	0,180	0,170	0,170	0,170
Random subspace projection	0,170	0,160	0,160	0,160
Multi-dimensional scaling	0,120	0,120	0,120	0,120
IsoMap	0,150	0,150	0,140	0,150
t-SNE	0,130	0,150	0,140	0,150
Decision-tree Classifier	0,368	0,375	0,348	0,375

Table 3: An overview of results obtained with methods for dimensionality reduction compared with decision-tree classifier.

5.3 Results of feature selection and projection methods

As said before, a short recap about real performances is in table 3, considering an analysis made on emotions on the same validation set (starting from the same training set) and considering as a reference the weighted average for all of applied methods.

6 Advanced Classification

In this section various methods of advanced classification are implemented. For each one, hyper-parameter tuning was performed, and outputs analysed. Dataset used is the original one with all attributes (columns), including data objects and string, without outliers (deleted in previous steps)

6.1 Support Vector Machine

Support vector machine is a classifier that attempts to learn a model by constructing a hyperplane, given a feature space, in such a way to separate the different class labels. It follows that the goal will be to find those margins that maximise the distance to specific points in the training called support vectors in order to ensure sufficient generalisation. Another characteristic aspect with this approach is the presence of non-linearly separable data, which results in the implementation of different kernels. In order to keep all these peculiarities in mind, we took the parameter C into strong consideration, which tells us how much we want to avoid misclassifying each training example; thus a high value of C would give us smaller margins and fewer errors in the training, on the other hand a high value of C would create larger margins and at the same time more errors in the training phase. Furthermore, different types of kernel were used in this parts: linear, RBF, polynomial, sigmoidal. The data was standardized with MinMax.

	Precision	Recall	F1 - score	Accuracy
Linear SVM	0,520	0,500	0,470	0,500
RBF	0,520	0,510	0,490	0,510
Polynomial	0,450	0,450	0,420	0,450
Sigmoidal	0,390	0,390	0,380	0,380

Table 4: SVM methods results referred to weighted avg and overall accuracy.

6.1.1 Linear approach

In order to select the best model, we carried out cross validation for C equal to [0.001, 0.01, 0.1, 1, 10, 100], obtaining the best result for C: 0.1 with an accuracy of approximately 49%. The performance was then confirmed by observing an accuracy of 50% on the test.

6.1.2 NON-Linear approaches

Also for the non-linear kernels, we used a similar approach to the previous one for the different values of the parameter C, namely:

- *RBF*: the best value of C was 10, giving an average accuracy value in validation of around 52% and in the test set of 51%.

- *POLYNOMIAL*: here we looked for the best configuration between the value to be assigned to C and the degree to be used. This was searched using GridSearch which returned the best pair C: 10 and Degree: 5 and observed an accuracy of 45% on the test set.

- *SIGMOIDAL*: the best value of C was 100, giving an average accuracy value on validation of 38% and on the test a similar result, thus observing the worst performance.

6.1.3 SVM assessment

Worst results were observed with Sigmoidal and Polynomial Kernels. Surprisingly, the performance of the linear model is almost the same of the best non-linear model obtained with RBF: this can be seen from the tuning carried out, observing that for too high/small C values we would have caused on the one hand too much rigidity, not generalising, and on the other hand too much flexibility by not fully capturing the patterns among the data to be followed.

6.2 Neural Networks

In this section, we will use artificial Neural Networks (ANNs) in order to exploit their power to solve the inherent problem of emotion recognition. The libraries adopted for

the evaluation and construction of the model are Sklearn, Tensorflow and Keras and the data were standardised with StandardScaler.

6.2.1 Neural Networks with Keras

In this part, a simple sequential model was built with dense layers in which we have 434 nodes as input followed by 128 hidden layers with relu activation function and another 64 with the same activation function. Finally, as output, 8 neurons (equal to the number of classes) with softmax activation function. In addition, compiler has the following parameters: loss = sparse categorical crossentropy, optimiser = adam, metrics = accuracy. We initially performed a fit without any regularisation and no early stopping condition by setting epochs: 200, validation split: 0.25 and a callback referring to the best value obtained on the accuracy of the validation set. This led us to an accuracy close to 48% and a loss of 1.67 on the test. In a second step, with same parameters, we inserted an early stopping condition with patience of 40 on the accuracy value of the validation set in order to handle the overfitting problem better; as a result, the epochs were less than 53. The results obtained on the test are very similar to the previous ones in terms of accuracy but with a loss of 2.33. We therefore tried a regularisation term L2: 0.01 with the early stopping condition, for the same motivation about overfitting, in which we observed 83 epochs and an accuracy result on the test of just over 48% and a loss of 3.39. We also tested different parameters for the regularisation factor: 0.001, which gave us a performance, again in terms of accuracy, of 49%, and 0.1, with an accuracy of 46%.

6.2.2 Neural Networks with Sklearn

In this part, we did some parameter tuning to check which architecture and values would be best for the activation functions and the learning rate. In particular, we implemented a RandomizedSearchCV for: solver: [adam, sgd], momentum: [0, 0.1, 0.2, 0.3, 0.5, 0.7,0.9], hidden_layer_sizes: [(512, 264, 128,64,32), (128, 64), (100, 50, 50)], activation:[relu,tanh], learning_rate:[constant, invscaling, adaptive]. As output we obtained: 'solver': 'adam', 'momentum': 0, 'learning_rate': 'constant', 'hidden_layer_sizes': (128, 64), 'activation': 'tanh'. Applying the obtained model to the test, the result was similar to the previous ones with a slightly higher accuracy of 50%.

6.3 Ensemble methods

6.3.1 Pre-processing

Before explaining how pre-processing is done, is important to say that variables are standardized with **StandardScaler** method from **sklearn.preprocessing** library. Data pre-processing is fundamental independently from ensamble methods used (they took in charge, for this specific case, decision tree classifier), allowing to make predictions on dataset with different data types. In this case we have two steps: encoding of

categorical variables integer values and standardization. Scope of normalization is to evitate loss of informative content and reducing variation between numerical attributes in dataset.

6.3.2 Bagging

Bagging technique is the acronym for "bootstrap aggregating". In this method, subsamples from a dataset are created and they are called "bootstrap sampling". Random subsets of a dataset are created using replacement, meaning that the same data point may be present in several subsets. These subsets are now treated as independent datasets, on which several Machine Learning models (Decision Tree in this case) will be fitted. During testing time, predictions from all such models are trained on different subsets of the same data taken in account for. There is an aggregation mechanism used to compute the final prediction (like Majority vote).

6.3.3 Random forest

The random forest algorithm is an extension of the bagging method as it uses both bagging and feature randomness to create an uncorrelated forest of decision trees. Feature randomness, also known as feature bagging or "the random subspace method", generates a random subset of features, which ensures low correlation among decision trees. This is a key difference between decision trees and random forests. While decision trees consider all the possible feature splits, random forests only select a subset of those features.

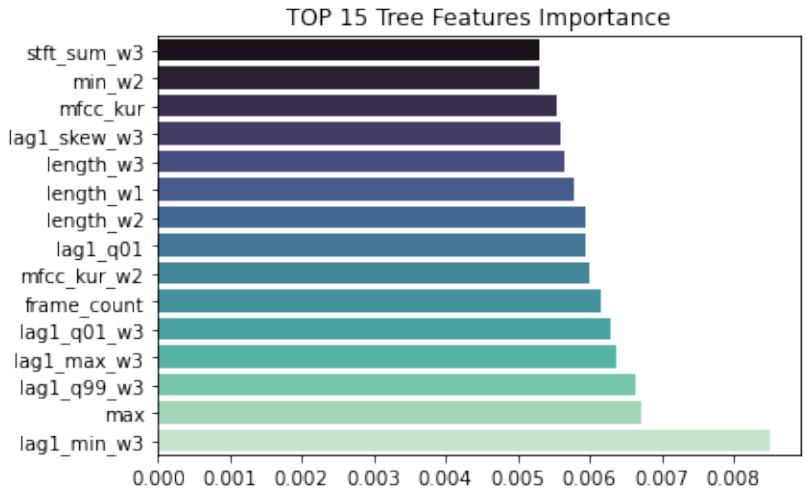
6.3.4 Adaboost

The boosting ensemble mechanism works in a way markedly different from the bagging mechanism. Here, instead of parallel processing of data, sequential processing of the dataset occurs. The first classifier is fed with the entire dataset, and the predictions are analyzed. Instances where Classifier-1 fails to produce correct predictions (that are samples near the decision boundary of the feature space) are fed to the second classifier thanks to the weights assigned to each record update at each iteration. This is done so that Classifier-2 can specifically focus on the problematic areas of feature space and learn an appropriate decision boundary. Similarly, further steps of the same idea are employed, and then the ensemble of all these previous classifiers is computed to make the final prediction on the test data. In order to test method's performance, it's obtained for learning rate intervals [0.2, 0.4, 0.6, 0.8] following values of accuracy: [0.355, 0.336, 0.327, 0.333].

6.3.5 Gradient Boosting Machine

Boosting is a method of converting weak learners into strong learners. Gradient Boosting trains many models in a gradual, additive and sequential manner. The major

(a)



(b)

Figure 9: Random forest features importance: (a) color gradient from lowest to biggest value, (b) sorted rating .

	Precision	**Recall**	**F1 - score**	**Accuracy**
Bagging	0,470	0,470	0,460	0,470
Random Forest	0,480	0,480	0,460	0,480
Adaboost - learning rate 0.2	0,370	0,360	0,340	0,360

Table 5: Summary about previous methods on test set considering weighted avg values and overall accuracy.

difference between AdaBoost and Gradient Boosting Algorithm is how the two algorithms identify the shortcomings of weak learners (e.g. decision trees). While the AdaBoost model identifies the shortcomings by using high weight data points, gradient boosting performs the same by using gradients in the loss function. Each model calculate estimating and selecting the best learning rate in order to maximize accuracy of the model, using a number of estimators of 50 and a width for trees different for each applied algorithm. For learning rates of 0.2,0.4,0.6 and 0.8 it is possible to obtain an accuracy, respectively, of 0.573, 0.546, 0.511, 0.526. Accuracy of the model, referring to emotions, it is around 0,57 on validation set (increasing performances compared to decision tree basic classifier).

	Precision	Recall	F1 - score	Accuracy
GB - machine Classification	0,490	0,480	0,470	0,480
XGB	0,500	0,500	0,490	0,500
LXGB	0,480	0,480	0,470	0,480

Table 6: Gradient Boosting machine performances considering weighted average and global accuracy on test set.

6.3.6 GB - methods: results

Best algorithm of classification of GDB's family is resulting XGB-Classifier. Gridsearch is applied to XGB on train set: in fact, it obtains best results using learning_rate = [0.2 , 0.4, 0.6, 0.8], gamma = [0, 0.1, 0.2] and max_depth = [4, 5, 6]. Algorithm chooses as parameters: learning_rate= 0.4, gamma=0 and max_depth = 6. In order to give an idea of differences between results, here there's a summary on table 6.

6.4 Logistic

In this part, we implemented logistic regression in order to solve our emotion classification problem. In the data preprocessing phase, we therefore standardised the data with StandardScaler. Our focus was mainly on the parameters penalty, considering 'l2' or None, and C, 0.01, 0.1, 1, 10, which for smaller values indicates greater regularisation. We therefore looked at the perforamance during validation and obtained the best configuration via GridSearch, which gave us C: 0.1, Penalty: "l2".

	precision	recall	f1-score	support
angry	0.54	0.76	0.63	96
calm	0.44	0.75	0.56	96
disgust	0.48	0.65	0.55	48
fearful	0.57	0.33	0.42	96
happy	0.40	0.43	0.41	96
neutral	0.36	0.25	0.30	48
sad	0.42	0.15	0.22	96
surprised	0.50	0.38	0.43	48
accuracy			0.47	624
macro avg	0.46	0.46	0.44	624
weighted avg	0.47	0.47	0.44	624

Figure 10: Results for logistic applied to emotions.

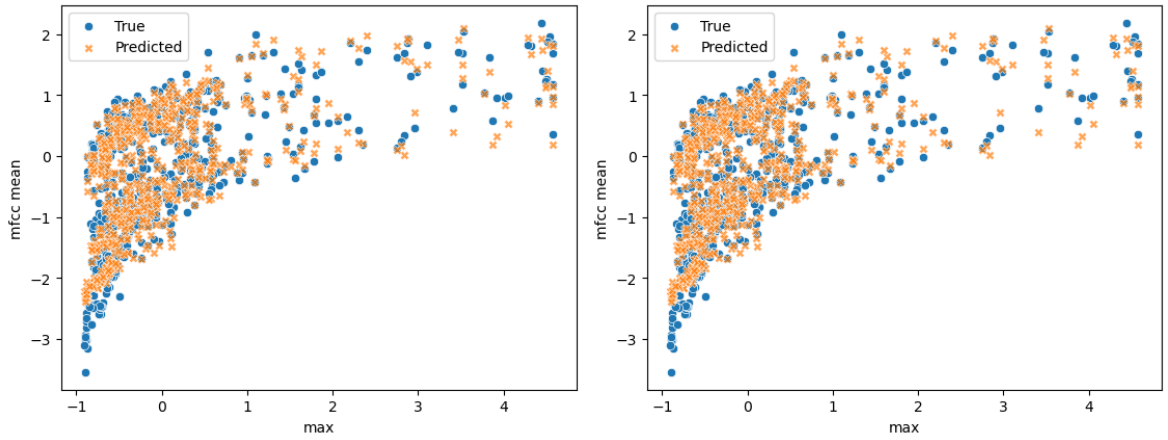
7 Regression

In this section, our objective is to solve a non-linear regression problem. We therefore opted to use Gradient Boosting for regression and Random Forest for regression using

mfcc mean as the target variable. For both approaches, the dataset with all features and data standardised with StandardScaler was used.

7.1 Gradient Boosting for regression

We observed early that the performance was very good during validation, and from this stage the most interesting parameters were found for min samples split:4, validation fraction: 0.15 and number iteration no change = 10. In particular, the last parameter tells us that if for 10 consecutive iterations the score of the validation set does not improve then we can stop, so in order to be more efficient and not fall into the problems of overfitting. Applying this model to the test set, we obtain good results for the 3 measures considered: R^2: 0.981 MSE: 0.023 MAE: 0.108 We can also observe from the graph, which shows the max variable on the x-axis and our target variable, mfcc_mean, on the y-axis, how the points predicted by our model turn out to be very close, if not overlapping, further confirming the goodness of the fit.



(a) (b)

Figure 11: An idea about regression's performance on mfcc_mean with GB (a) and using Random forest regressor (b).

7.2 Random forest regressor

Even with this approach, results were very satisfactory and from the validation phase we decided to stay with the default parameters for our model; i.e. max_depth of a tree until it becomes a pure leaf, having entered no conditions for min samples split different than 2 and min samples leaf different than 1. Applying this model to the test set we again obtain good results for three measurements taken into consideration: R^2: 0.967 MSE: 0.040 MAE: 0.145 . Figures above show no impact in terms of visualization:

that's related to a R^2 coefficient that is similar for GB and Random forest (0,981 and 0,967 respectively).

8 Time Series

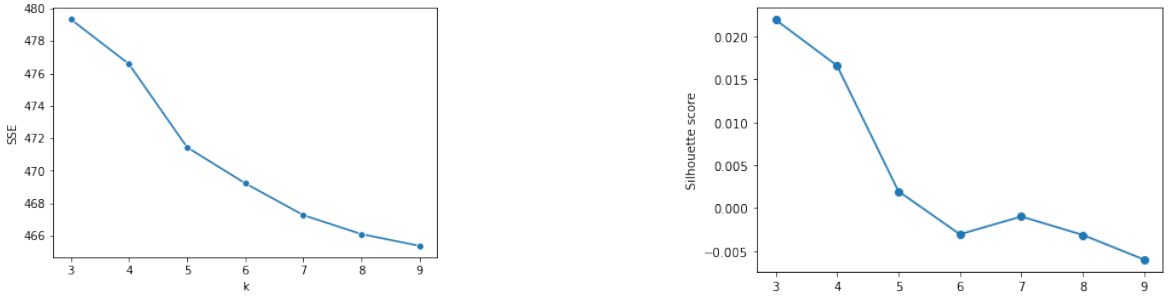
For this section, using *librosa* library, Time Series were extracted starting from frequency obtaining a 305906 timestamps for each one of them. Then, it was decided to approximate with **SAX** (Symbolic Aggregate Approximation) the number of timestamps in each of the following subsections with specifications given for each steps. It is important to say that all ts were kept, all of them were centered around the zero value and no trend were found in terms of signals repetition. For what concerns variance, we decided to keep it in order not to lose information about emotions (which were the focus of our analysis).

8.1 Clustering

The scope of clustering in this section is to discover and group together possible aggregates of similar time series. In order to accomplish it, two different distance methods were used with K-means approach. Globally, only train set were used in this part due to high computational time response with these approximations and following the same logic of the first part of this project in which unsupervised methods were applied on the same set.

8.1.1 K-means (Euclidean distance)

Before discussing results of K-means with euclidean distance criteria, it is important to underline that **SAX** approximation is applied with 3000 segments and 8 region of interest equi-probable in order to reduce computational cost and complexity in terms of algorithm's response. In this range, SSE and Silhouette represent a trade-off to



(a) (b)

Figure 12: SSE (a) and Silhouette (b) scores for k in range [3,9].

consider before assessing results: for k greater than 4 the SSE (which decrease sensibly for high k, arriving at a minimum for k = 1828) is still high but the silhouette score is one of the highest (silhouette decrease sensibly for high k, same as SSE). For these reasons, it was decided to consider k = 4 and keeping a positive value for silhouette. Number of iterations considered in each step were 50 to let the algorithm converge in reasonable times.

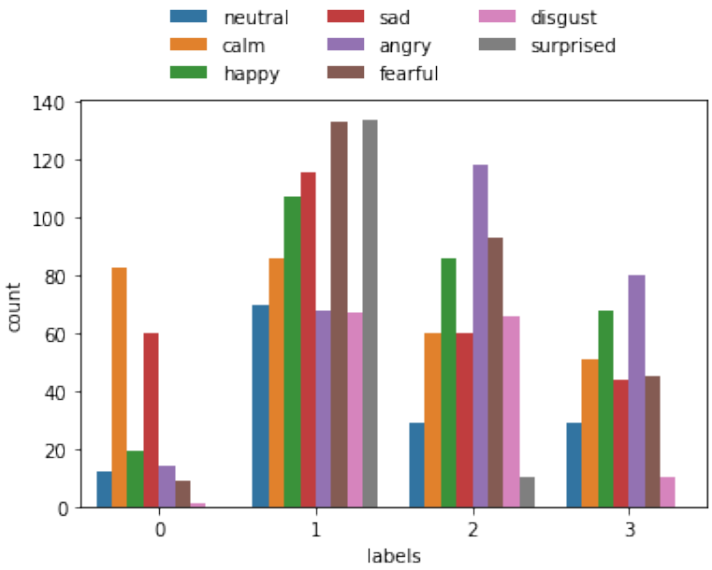
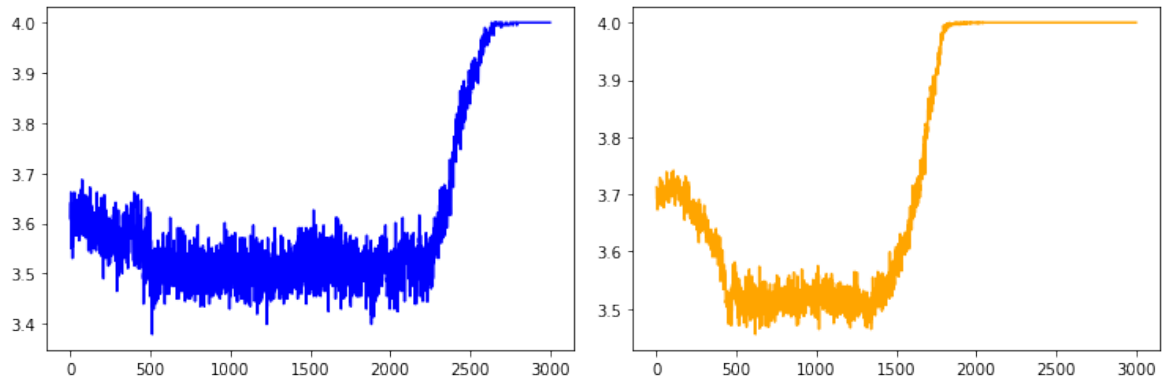


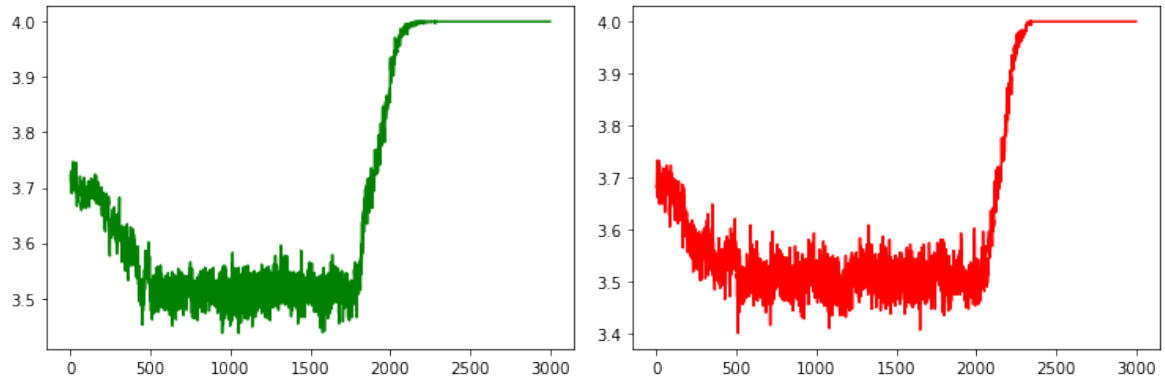
Figure 13: Centroids divided par emotions.



(a) (b)

Figure 14: Centroids 0 (a) and 1 (b) representations.

In figures 14 and 15, it's possible to appreciate the correlation between the single aggregates and the countplot divided par emotions and cluster centroid. Peaks for



(a) (b)

Figure 15: Centroids 2 (a) and 3 (b) representations.

calm and sad are capture by cluster 0, angry and fearful underlined by cluster 1 and 2, happy and angry for cluster 3. Differences between each aggregates is made up by the queue of the signal and the amplitude of the audio tracks (structure along x-axis and peaks reached on y-axis in the central part of the aggregate). Those charachteristics will be recalled during motif analysis clarifications because it's possible to associate also physical meanings to this phenomenon (especially in case of the queue).

8.1.2 K-means (DTW)

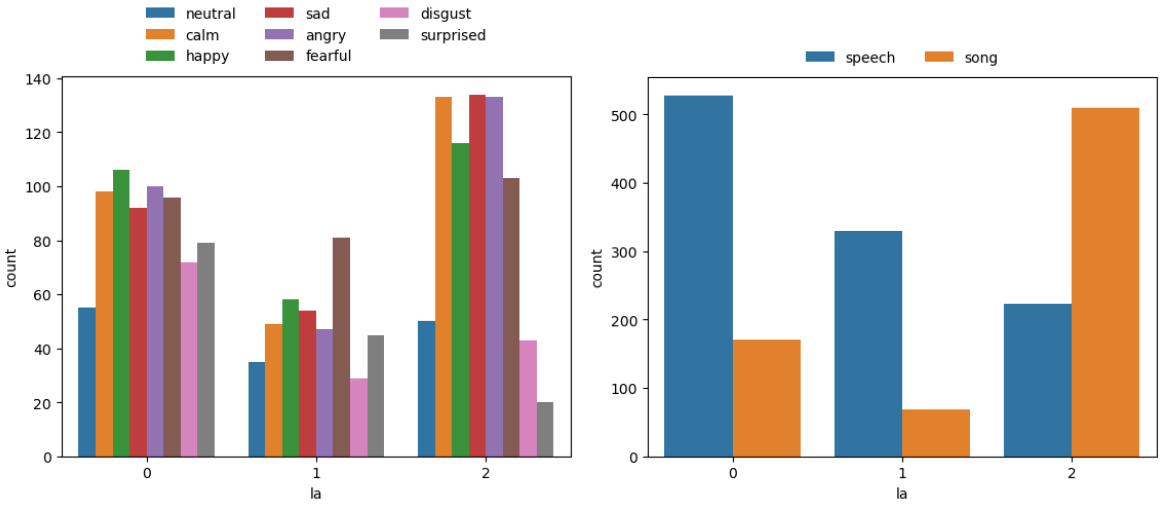
Before discussing this section, it's important to say that only 30 segments are used in this part, in order to obtain a reasonable answer in terms of computational time and complexity. So as in previous section, other setting were tested but with no appreciable results. So as in K-means with Euclidean, it's decided to use k = 3 losing

(a) (b)

Figure 16: SSE (a) and Silhouette (b) scores representations for k in range [3,7].

in terms of SSE and keeping a positive value of silhouette which, considering an high

approximation degree, leads not to wrong clusters decision (like in other cases, having a silhouette below zero). As it's possible to see from previous images, **dtw** distance is



(a) (b)

Figure 17: Emotions (a) and vocal channel (b) countplot representations.

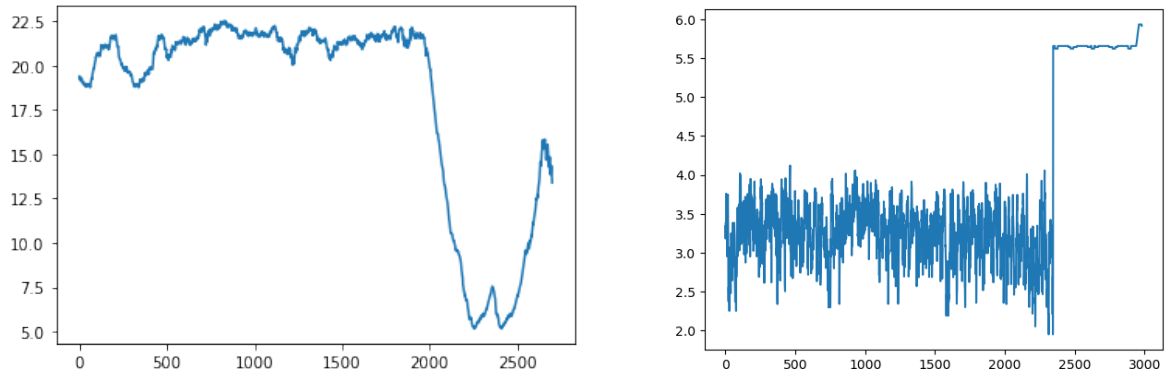
more capable to underline differences in vocal channel. For what concerns emotions, they are aggregated underlining peaks for fearful in cluster 1 and disgust, calm and sad emotions for cluster 2. In general, no big discrimination is made in between emotions for each cluster.

8.2 Motifs and Discords

In this section, scope is to discover repeated patterns in centroids already found with K-means using Euclidean distance criteria. Following pictures will be used in order to explain what kind of interesting structures emerged after our simplifications.

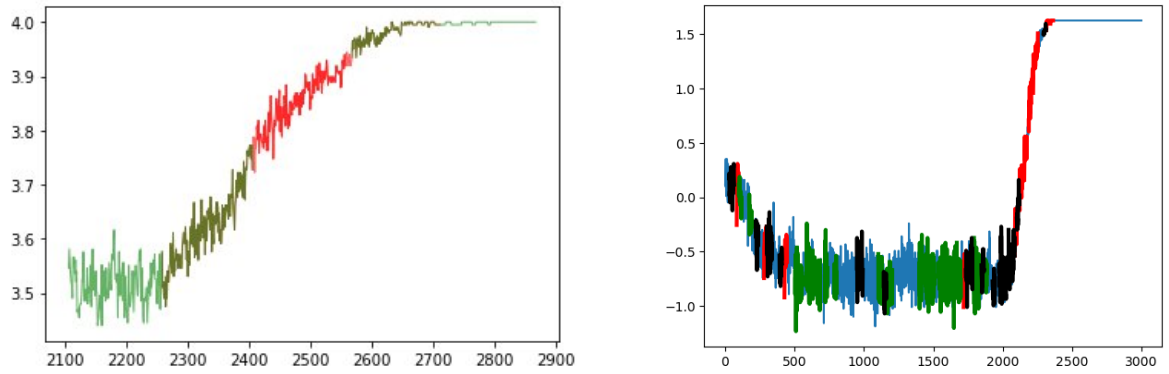
Just as an example, it is provided in Figure 18 the representation of the matrix profile of centroid 0 at a certain w. Clearly, this is influenced by the windows size selected. Considering this specific case, it's possible to see that for lower peaks (indexes 2256 and 2413) distances tend to zero and in these point motifs are identified.

For what concerns centroid 3, the chosen window size was 20 and the lower peaks were found for indexes 2314 and 2344. At this point, motifs found in the same centroids as before were represented in Figure 19. It's possible to notice that for motif of centroid 0 there are two motifs overlapped in dark green. In both cases (independently from the centroid), before the end of the aggregation there's a motif structure: this could



(a) (b)

Figure 18: Matrix profile of centroid 0 and w = 300 (a) and matrix profile of centroid 3 and w = 20 (b).



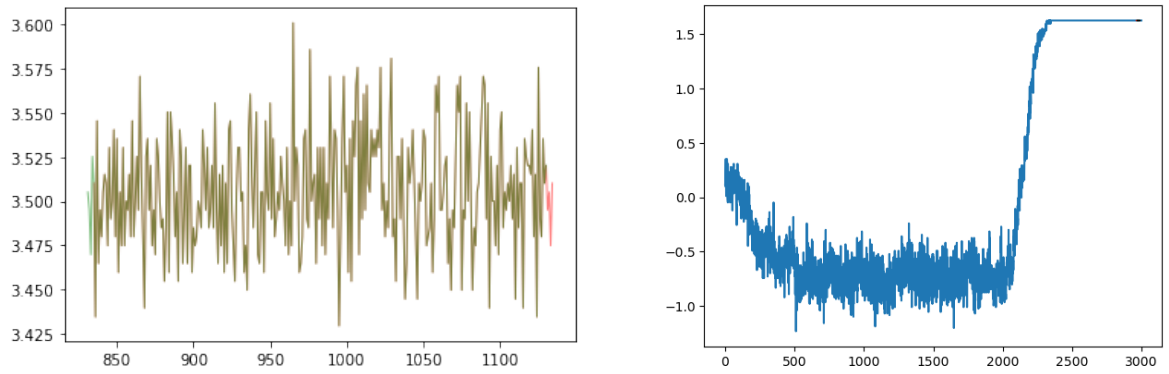
(a) (b)

Figure 19: Motif overlapped of centroid 0 and w = 300 (a) and maotif of centroid 3 and w = 20 (b).

have a physical explanation in the fact that at the end of recorded part, by the point of view of audio signal noise and disturb components of sound are maximized. This was captured also in a discord in centroid 3 and noticeable at the beginning from matrix profile.

In figure 20 there are details related to discords in centroids 0 and 3, respectively. In particular, in the first case there's completely overlapping (except for beginning and ending of the aggregates) in a range of timestamps from 850 and 1150.

In conclusion, we mention that differences in results obtained with reference to discords and motifs are also related to the two different window sizes used. This parameter



(a) (b)

Figure 20: Discords overlapped of centroid 0 and w = 300 (a) and discords of centroid 3 and w = 20 (b).

is clearly influencing methods in terms of complexity, response and results.

8.3 Classification

In this section, our aim was to solve a classification problem related to emotions present in our dataset. In order to do that, we implemented three algorithms to understand which could be performances depending from their application.

8.3.1 Shapelet

Considering this algorithm and the data structure, building a transformed dataframe from shapelets is very expensive. In our specific case, two simultaneously strategies were applied: a strong approximation with SAX at 150 segments and a sampling stratification in order to obtain an output. A test was made choosing 75 shapelets over 150 was not satisfactory. So, we tried with 8 shapelets trying to catch possibly the 8 various emotions. Obviously, considering strategies adopted, results were not as ones ideally supposed, and globally performance were very poor (see Table 8). Another consideration to be mentioned regards one of the first hypothesis made for simplification: we forced to use all ts in the same domain between 0 and 150 timestamps with no consequences in emotion recognition structure. This compression influence the possibility to separate and identify distinct emotion shapelets for each one of them. In the following Table 7 is possible to see the 8 shapelets found, including their index position (referred to the number of the time series took as reference) and the start/end position referred to the timestamps, each one has an emotion. It's important to say that we took 12 as windows size, and as in previous cases it's one of the parameters which influence the structure obtained as an output. Clearly, each of these shapelets refer to an actor, a sex, a repetition and a statement. From the point of view of this

index	start	end	emotion
98	20	32	happy
313	17	29	happy
600	103	115	calm
252	15	27	calm
103	82	94	angry
531	58	70	fearful
490	65	77	fearful
504	76	88	sad

Table 7: Window size equals to 12 for the 8 shapelets obtained, divided par emotions and identified by index.

	Precision	Recall	F1 - score	Support
angry	0,140	0,290	0,190	96
calm	0,180	0,400	0,250	96
disgust	0,040	0,020	0,030	48
fearful	0,090	0,030	0,050	96
happy	0,140	0,140	0,140	96
neutral	0,110	0,040	0,060	48
sad	0,150	0,050	0,080	96
surprised	0,000	0,000	0,000	48
accuracy			0,140	624
macro avg	0,110	0,120	0,100	624
weighted avg	0,120	0,140	0,110	624

Table 8: An overview of results obtained with decision-tree classifier on datastet transformed with shapelets. Test set were used to assess performances.

analysis, and considering the number of shapelets took as samples, it's not fundamental to report them because there is no relevance to asses all various combinations between each possible value in categorical attributes.

8.3.2 KNN

In this section dataset used were approximated using SAX with 3000 segments for Euclidean distance and with 150 segments for Dynamic Time Warping. For best k selection a cross validation were used with k in range [1,100]. As we can see in Figure 21, best k is equal to 20. Performances are a little bit higher than decision tree after shapelets transformation, but not very satisfactory in general. All results are displayed in Table 9. After that, KNN was tried using dtw as distance criteria (same k = 20 kept), but instead of the standard one, Sakoe-Chiba Band and Itakura Parallelogram were adopted in order to see possible differences in terms performances and speed up calculations preventing patological warpings.

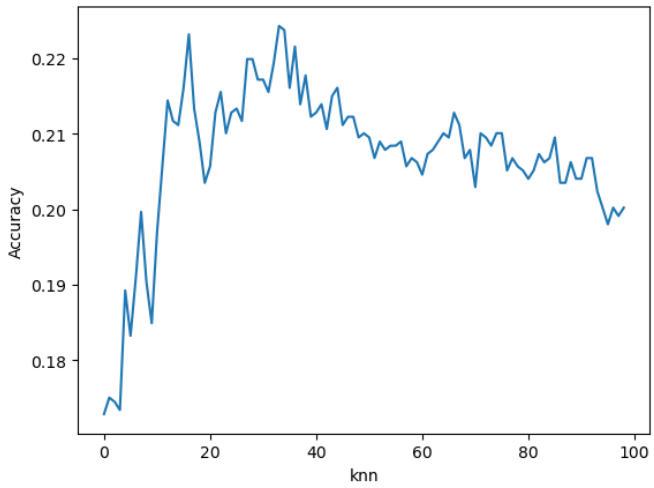


Figure 21: Cross validation for k in range [1,100].

	Precision	**Recall**	**F1 - score**	**Support**
angry	0,270	0,180	0,210	96
calm	0,160	0,250	0,200	96
disgust	0,170	0,100	0,130	48
fearful	0,250	0,170	0,200	96
happy	0,150	0,080	0,110	96
neutral	0,130	0,170	0,150	48
sad	0,060	0,030	0,040	96
surprised	0,160	0,500	0,240	48
accuracy			0,170	624
macro avg	0,170	0,180	0,160	624
weighted avg	0,170	0,170	0,160	624

Table 9: An overview of results obtained with KNN classifier on test set.

8.3.3 Rocket and Mini-Rocket

Random convolutional kernel transform uses a large number of random convolutional kernels to transform the time series exploiting with this method and it's lighter version (mini-rocket). The aim is obtaining better performances. For Rocket, number of kernels used is 2000 and for mini-Rocket 10000. In both cases, ridge classifier was used. Performances obtained are the best so far 'til now.

	Precision	Recall	F1 - score	Accuracy
Sakoe-Chiba Band	0,210	0,210	0,190	0,210
Itakura Parallelogram	0,210	0,210	0,200	0,210

Table 10: Sakoe-Chiba Band and Itakura Parallelogram performances considering weighted average and global accuracy on test set.

	Precision	Recall	F1 - score	Accuracy
Rocket	0,360	0,370	0,330	0,370
mini-Rocket	0,390	0,380	0,370	0,380

Table 11: Rocket and mini-Rocket performances considering weighted average and global accuracy on test set.

9 Explanaible AI

In this section, model used is the one referred to Support Vector Machine (kernel RBF). The first global approach is based on a decision tree classifier in order to simulate model behaviour of our "black box" (in our case represented by SVM). Fitting on predicted labels was made as first step (max_depth = 3 for interpretability reasons), results available in Figure 22. Best discrimination for a single emotion is obtained for angry

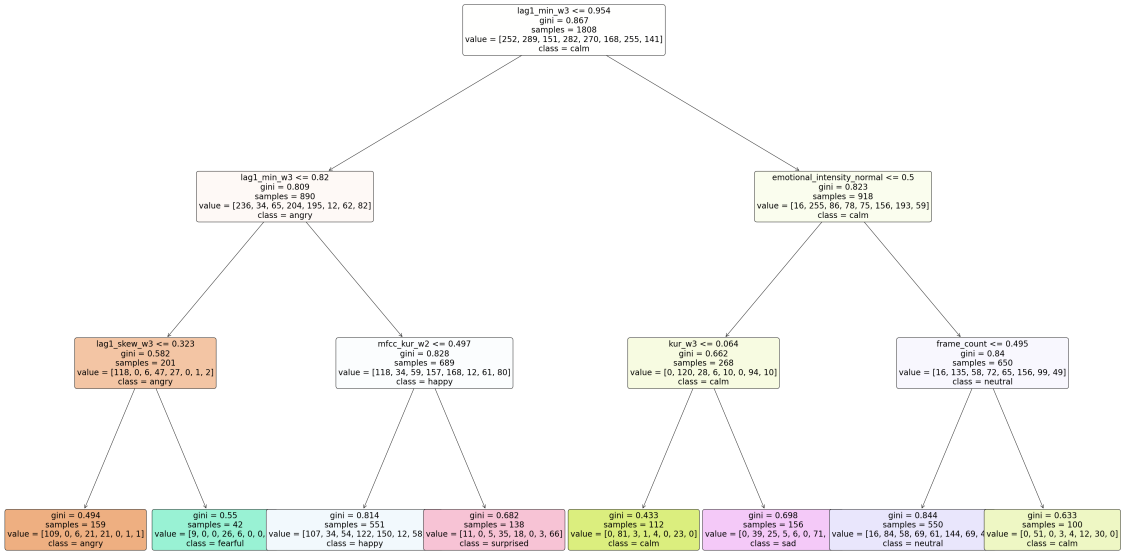
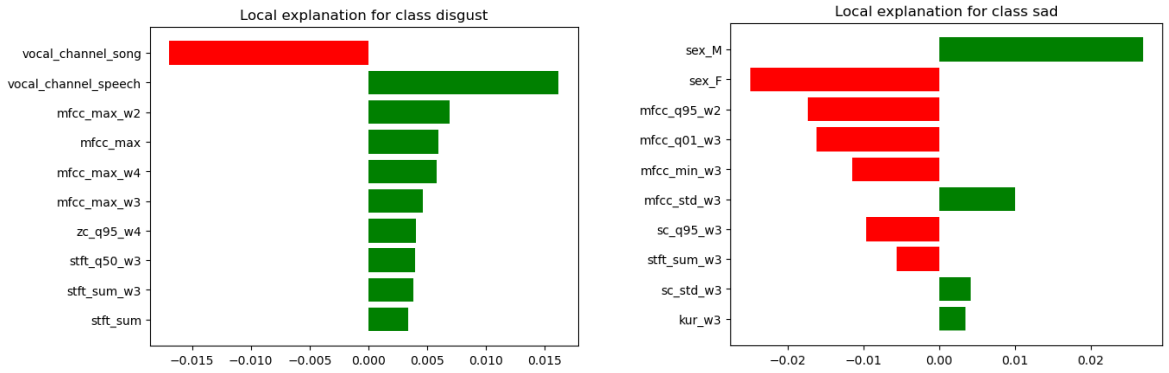


Figure 22: Decision tree built on prediction of SVM.

using lag1_skew_w3 less or equals to 0,323 by Gini index. Leaf in this case has a Gini

coefficient of 0,494 and a samples number of 159 (109 of them are properly identified as angry itself).

After that, a local approach was used with **Lime**. In this case, records used was the positional 228 in test set, predicted as disgust whilst sad is the correct class. As it's



(a) (b)

Figure 23: Lime explanation for disgust (a) and sad (b) class in record 228.

possible to see from the images, in the first top 10 coefficients, dominance of negative features importance in sad class make probability to obtain results in model critically reduced; in fact, prediction probabilities assigned from the model to disgust class is 77% (for sad, instead, is only 4%).