UNIVERSITY OF PISA

DEPARTMENT OF INFORMATICS

Master Degree in Data Science and Business Informatics

Data Mining 2

Davide Piccoli – Matteo Di Giorgio – Lorenzo Parra

Academic Year 2022/2023

Contents

1 Introduction 1

2 Data understanding and preparation 1

3 Dimensionality reduction 2

4 Outlier detection 4

5 Imbalanced learning 7

6 Advanced classification 10
- 6.1 Logistic regression 10
- 6.2 Random forest 11
- 6.3 Bagging 12
- 6.4 AdaBoost 13
- 6.5 XGBoost 14
- 6.6 Neural network 15
- 6.7 Support vector machine 16

7 Advanced regression 18

8 Time series 18
- 8.1 Clustering 19
- 8.2 Motifs and discords 21
- 8.3 Classification 23
- 8.4 Explainability 24

1. Introduction

The aim of this report is to provide some analysis related to a dataset created from the RAVDESS one. In particular, basic statistics were firstly extracted from the RAVDESS dataset and secondly transformed using differencing, zero-crossing-rate, Mel-Frequency Cepstral Coefficients, spectral centroid, and the stft chromagram. These features were extracted from a total of 2452 wav files and were also obtained by dividing each time series into 4 non overlapping windows.
The analysis conducted is then discussed and an explanation of its results is given.

2. Data understanding and preparation

In the first place, we looked for missing values, but none was found. Then, we focused on the features and decided to exclude from the dataset all those having only one possible value for each record: this led to a decrease in the number of attributes, from 434 to 382.
The next step regarded the analysis of the correlation between each couple of attributes. Since the attributes were too many, the resulting correlation matrix was unreadable. Thus, we decided to temporarily drop the attributes related to time-series. This operation led to the result below, where only those couples of attributes having a correlation greater than 0.9 were highlighted. As a consequence, the following features were removed: "stft_skew", "mfcc_min", "zc_mean", "zc_kur", "zc_skew", "min", "max", "sc_sum".

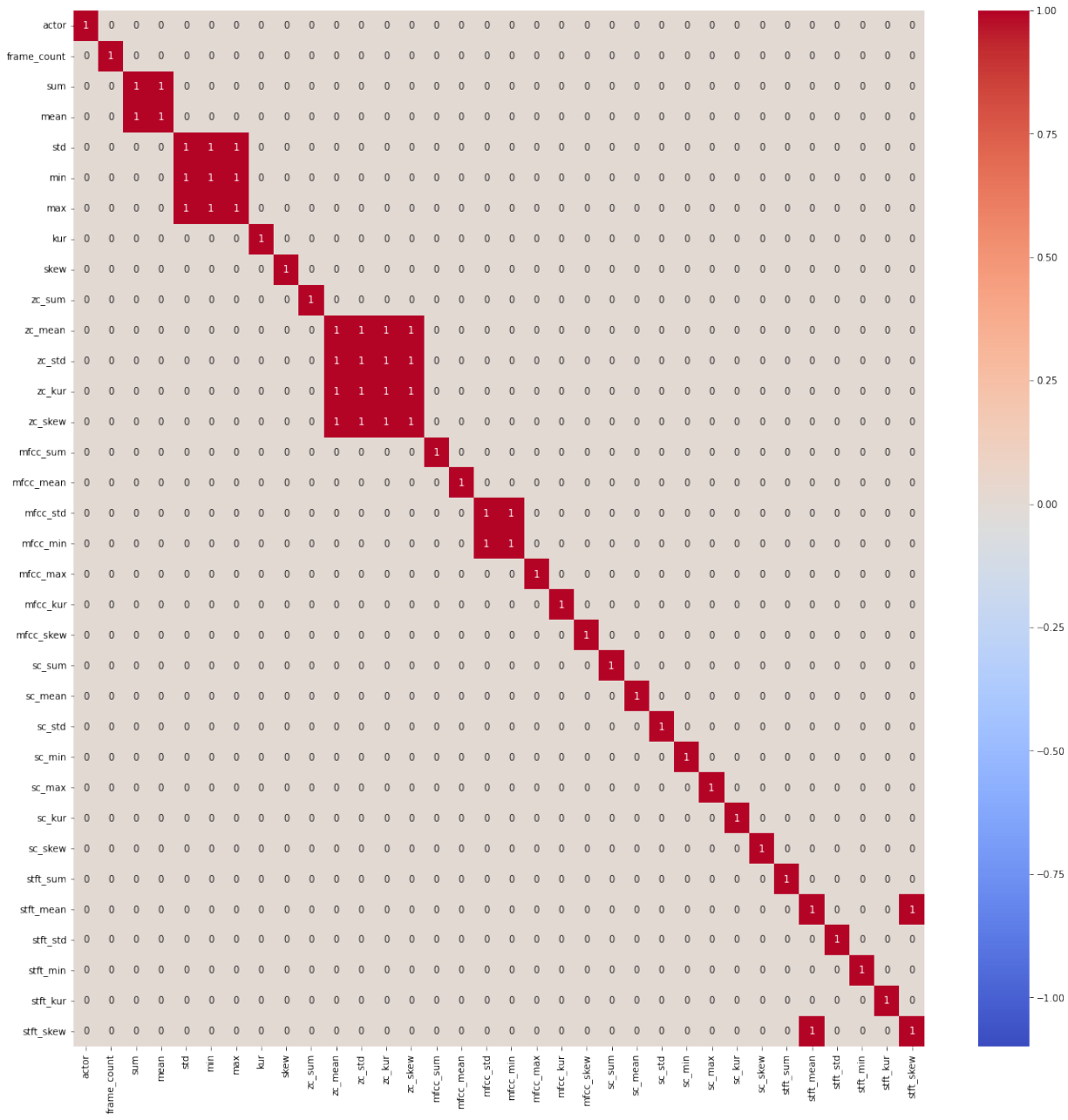


Figure 1: Correlation matrix of a subset of attributes.

While looking for possible errors, the attribute "filename" was found to be a primary key for this dataset. This would have compromised the results of the following classification tasks, thus it was removed too.

3. Dimensionality reduction

In order to find a good subset of the features, a classification task was chosen such that the final subset could be the one optimising its accuracy score. More specifically, a simple decision tree was implemented to classify the records based on their related emotion. Before reducing the number of dimensions, this classifier got an accuracy score equal to 0.371.

Then, the variance of the different features was examined, so that the Variance threshold method could be implemented. The histogram on the left in Figure2 shows its original distribution.
Since almost all the features had a variance included in the first bin, we repeatedly "zoomed" on it, until the boundaries of the distribution were much closer to each other and slightly greater than zero. At this point, the accuracy of the decision tree classifier was measured on a validation set, considering different thresholds between these boundaries that caused different subsets of the original features set to be taken into account. The result is reported by the plot on the right in Figure2.

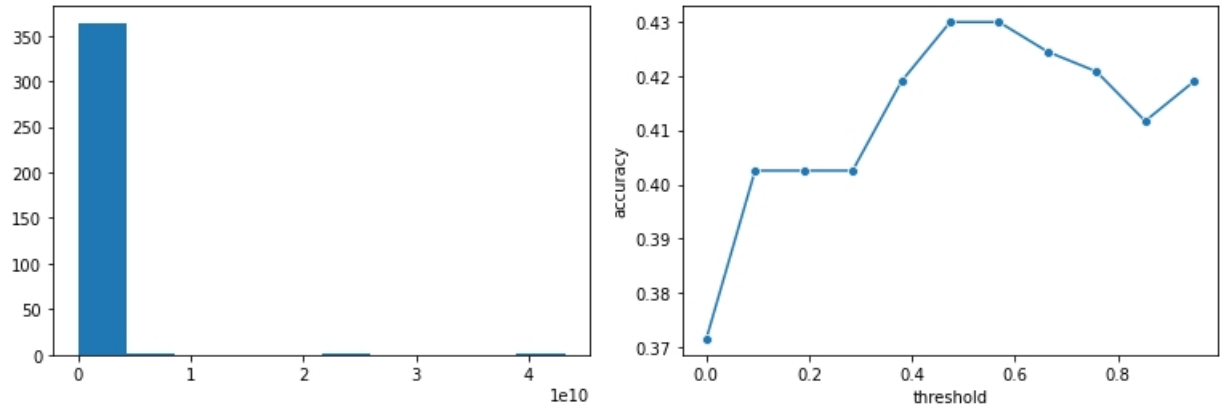


Figure 2: On the left: variance distribution of the features. On the right: accuracy score on the validation set of the classifier, considering different variance thresholds for dimensionality reduction.

The plot on the right lead us to discard all the features having variance greater than 0.47, causing the new dataset to be made up of 174 attributes. Then, the classifier was implemented on this new dataset and the resulting accuracy score on the test set was 0.41, highlighting a better performance on this reduced dataset rather than on the original one.
The following step focused on PCA, in order to visualize these 174 attributes. The following graph shows that the first two components already explain a large majority of the features variance, with other two "smaller elbows" observed in correspondence of 3 and 5 as numbers of principal components.

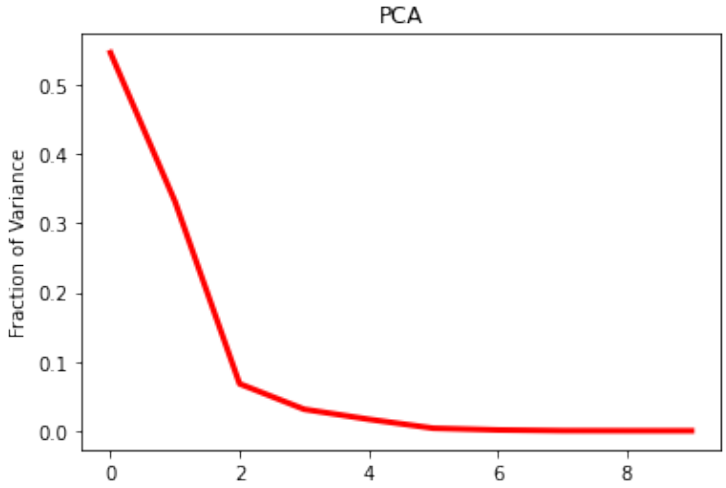


Figure 3: Fraction of variance explained with respect to the number of PCA components.

The next step focused on the visualization of all the features projected on new sets of two and three dimensions and the result can be observed below. The dimensions of the scatterplots are the components, while data points are coloured according to their emotion label.

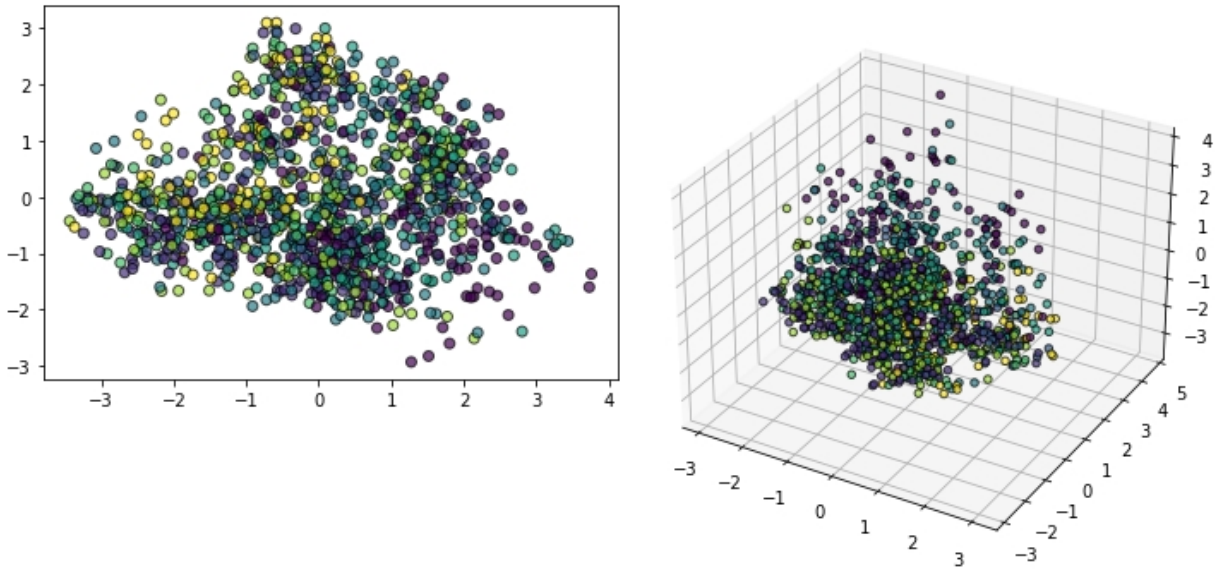


Figure 4: Scatterplots representing PCA with two components (on the left) and three components (on the right).

Back to our original classification task, which at this point had an accuracy score equal to 0.41, the same Decision Tree applied to these two reduced datasets was not able to predict emotions that well anymore. Indeed, the accuracy score is equal to 0.17 when the number of features is reduced to two, while it is equal to 0.26 when the number of features is three. This basically shows that, although two and three features are already enough to cover almost all the variance among all the attributes in the dataset, they are not enough to obtain acceptable accuracy scores in our classification task.

4. Outlier detection

The first method adopted to identify outliers is HBOS, which is a visualization-based approach. HBOS is a scoring method and indeed the distribution of the resulting outlier scores is shown by the histogram below. The records whose outlier score is greater than 265 (the value indicted by the black vertical line) are classified as outliers.

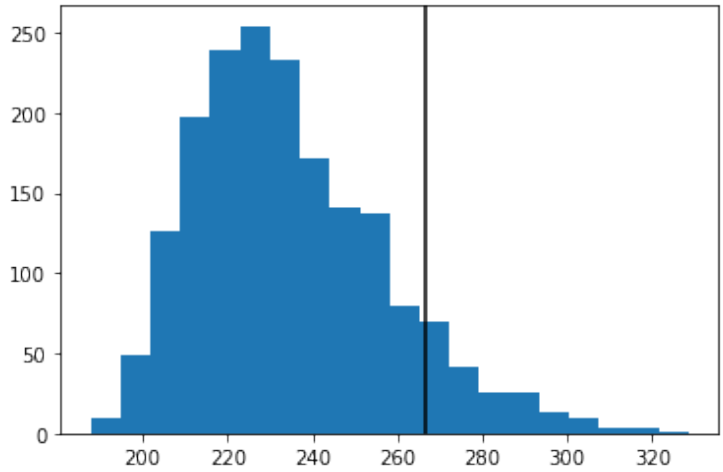


Figure 5: Histogram representing outlier scores according to HBOS.

In total, HBOS found 183 outliers: they are the data points coloured in green in the 2d and 3d scatterplots below, obtained through PCA.

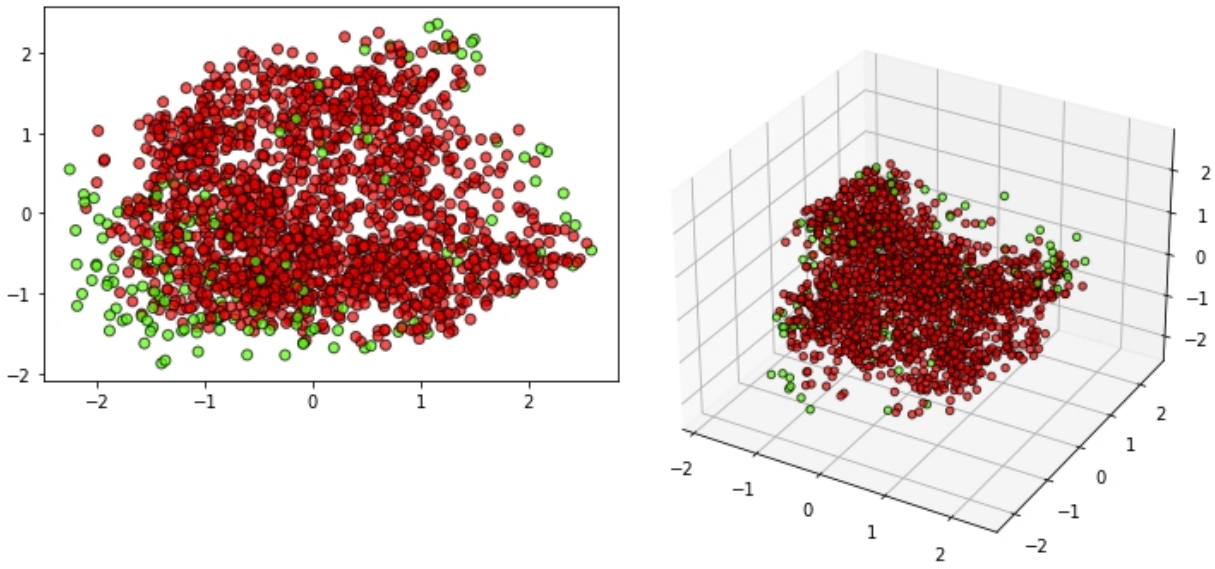


Figure 6: PCA applied on the dataset and highlighting outlier data points according to HBOS.

Another method we implemented is the LOF density-based approach. This method outputs an outlier score for each record, which is the inverse of its relative density. A record is an outlier if this score is "much greater" than 1, as better explained by the histogram below from our dataset. Points whose score is on the right of the vertical line are outlier and LOF found a total of 168.

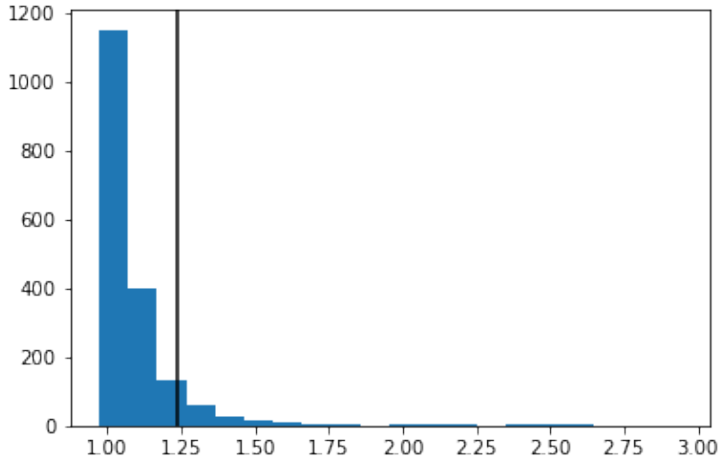


Figure 7: Histogram representing outlier scores according to LOF.

Then again PCA allowed us to visualize these outliers projecting the data points on two and three dimensions.

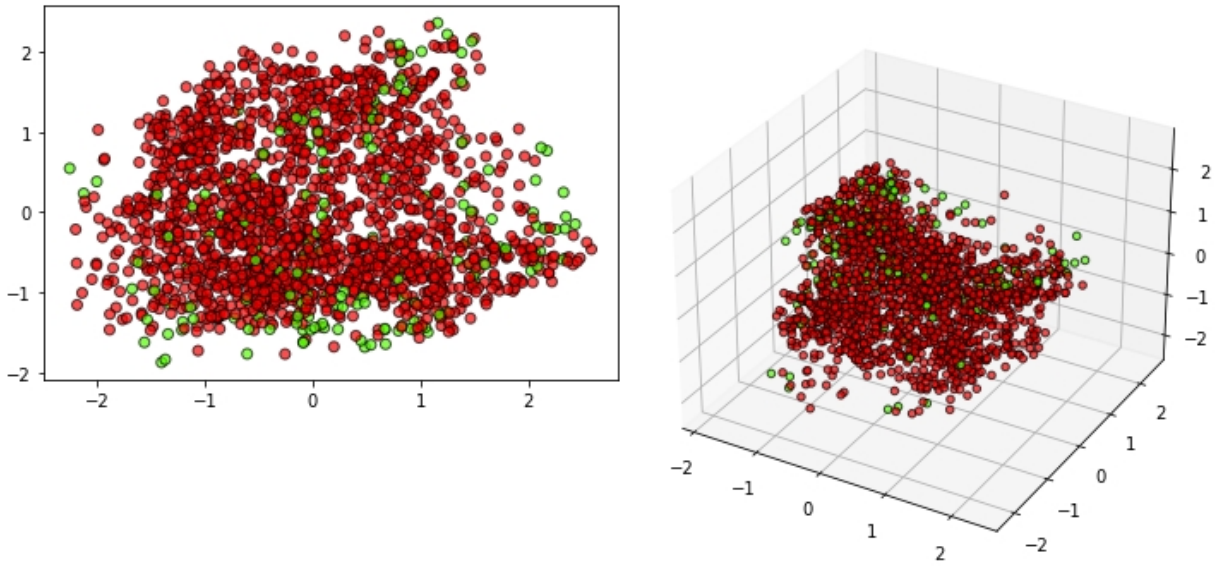


Figure 8: PCA applied on the dataset and highlighting outlier data points according to HBOS.

The LOF approach found a slightly smaller amount of outliers compared to HBOS. This might have happened because some of the outliers found by HBOS were in a low-density region, where LOF usually has some trouble in detecting them. Comparing figures 6 and 8 suggests that LOF was not able to recognise as outliers some points at the border of the cloud, where indeed the density is lower.
However, these two methods are not the best to implement when dealing with high-dimensional data like in this case. This is the reason why another method we implemented is the ABOD, thus exploiting angles between data points instead of their distances. The histogram below shows that those points having an ABOD score on the left of the vertical line are outliers.

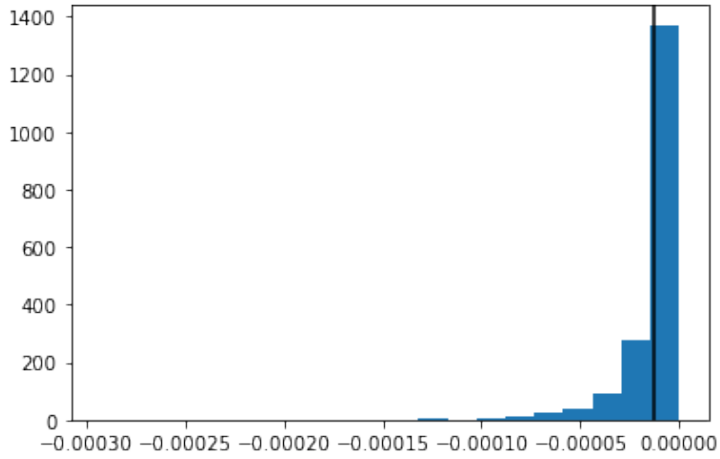


Figure 9: Histogram representing outlier scores according to ABOD.

In this case, the outliers detected are 242, represented in green in the two PCA scatterplots below.

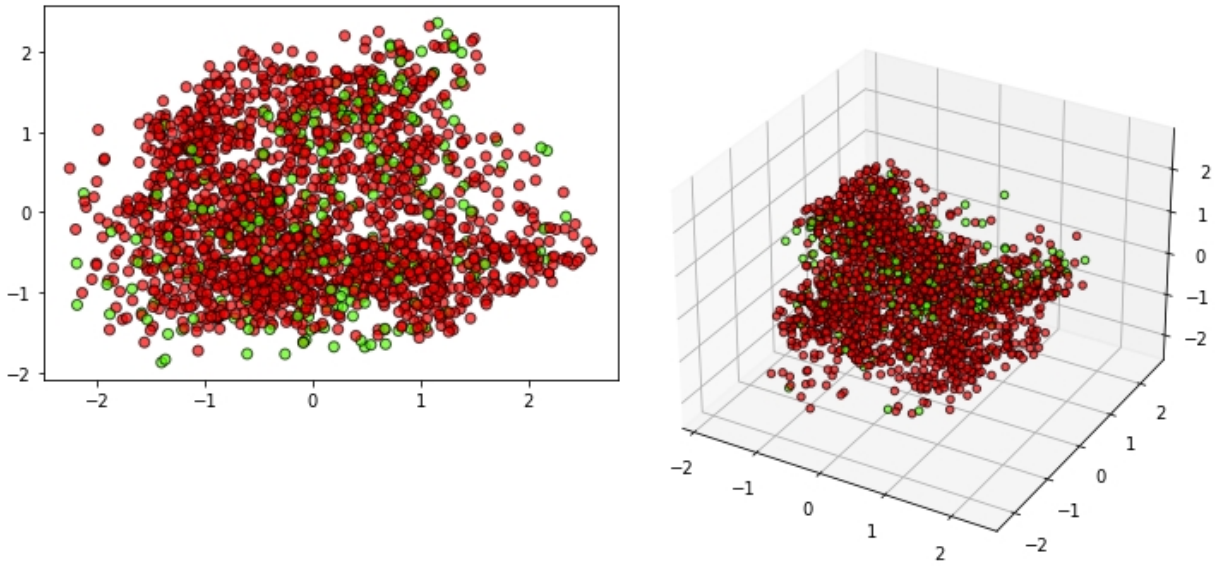


Figure 10: PCA applied on the dataset and highlighting outlier data points according to ABOD.

As expected, the angle-based approach worked better against the curse of dimensionality and was indeed able to detect a larger number of outliers.
In order to remove the top 1% outliers, we first sorted all the records according to their outlier score returned by each method. Then we selected a number of records such that the intersection among the three resulting sets would be made up of 18 records, which is indeed 1% of the total number of data points. Once this intersection was found, the 18 records were discarded from the dataset.

5. Imbalanced learning

To test how various imbalanced learning methods perform on the dataset first we defined a classification task. In order to better visualize the outputs of the task we decided to choose the attribute “sex” as our target variable due to the fact it has only two possible values, thus the visualizations (for instance PCA) would result clearer. To solve the task we adopted a decision tree classifier.
Firstly, we solved the task on the original dataset and the resulting accuracy was found to be around 0.92.

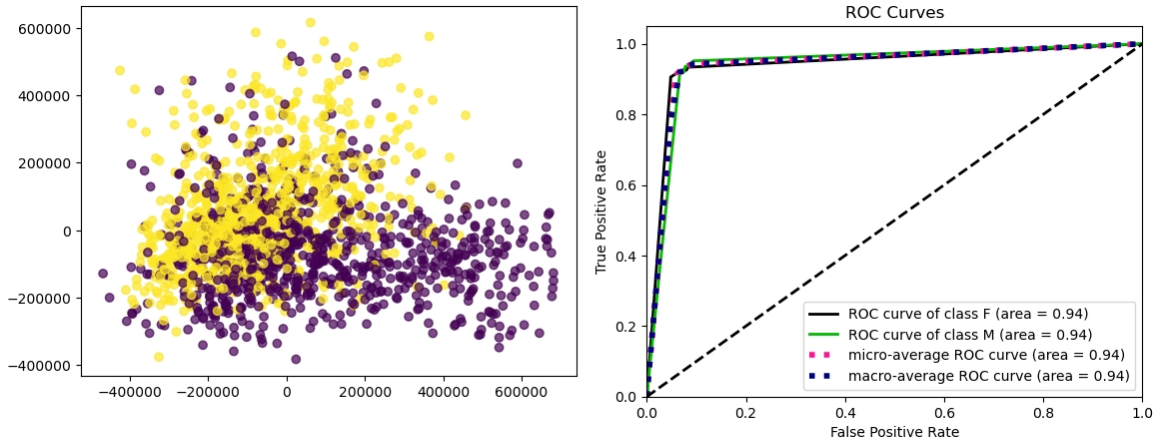


Figure 11: On the left: PCA allows to visualize the (balanced) original distribution of data points based on their ”sex”. On the right: ROC curves on both classes of the decision tree classifier applied on the original dataset.

Due to the fact that the distribution of the two classes was balanced, the majority of the rows with ‘M’ class were

removed from the dataset ending up in a situation where the rows with 'F' class were around 95% of the total instances of the dataset. The decision tree was then implemented again and, even though the overall accuracy increased from 0.92 on the balanced dataset up to 0.96 on the imbalanced dataset, the capability of the model to predict the least represented class significantly worsened. Precision, recall and F-1 score dropped by around 35% for the records whose "sex" was F.

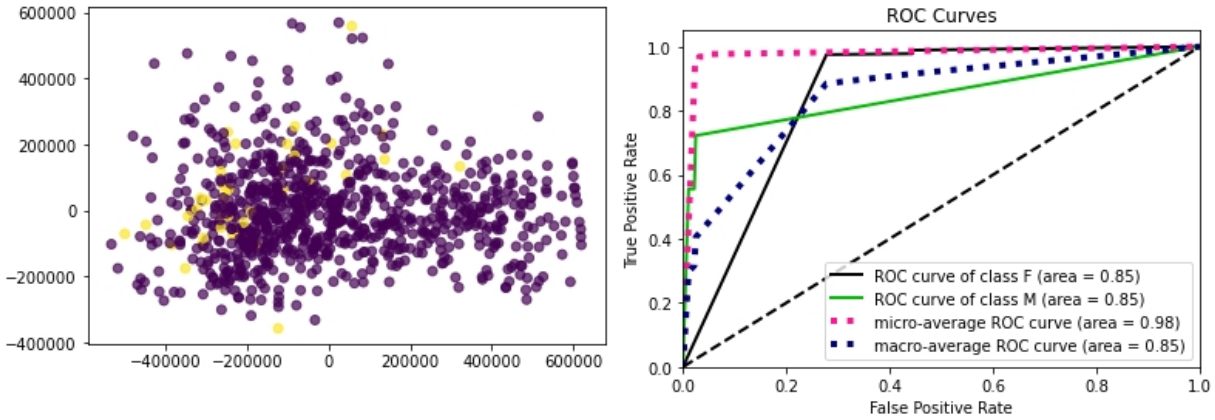


Figure 12: On the left: PCA allows to visualize the now imbalanced distribution of data points based on their "sex". On the right: ROC curves on both classes of the decision tree classifier applied on the original dataset.

Starting from random undersampling, its result was a dataset including 84 data points: 42 of which belonging to the class "M" and the other 42 belonging to the class "F". Of course this balance made the classification job tougher for the decision tree and its accuracy decreased down to 0.83.

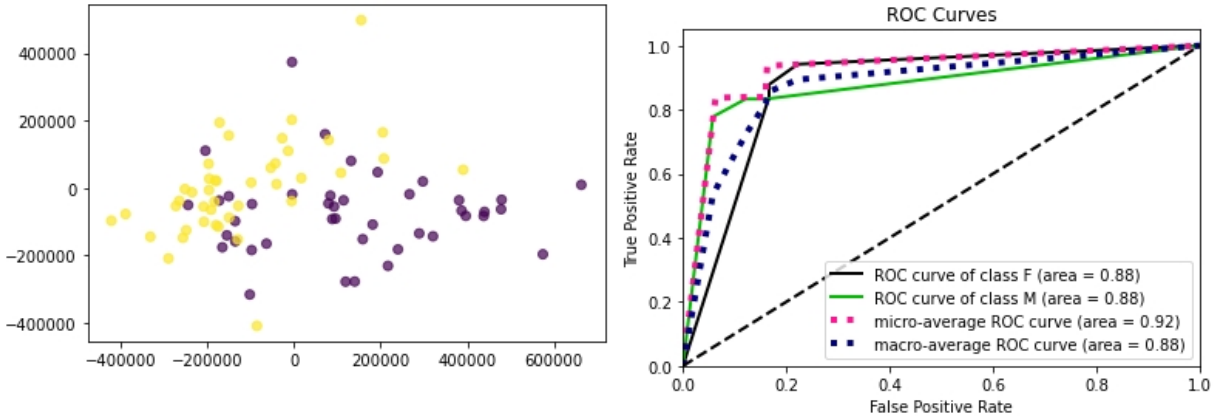


Figure 13: On the left: PCA allows to visualize the distribution of data points based on their "sex", balanced by random undersampling. On the right: ROC curves on both classes of the decision tree classifier applied on the dataset after random undersampling.

In order to observe the differences between a random method and a structured one, the Cluster Centroids method was also applied on the imbalanced dataset. The chosen estimator was the MiniBatchKMeans and the result was a dataset of 120 records, half of which of each class. The accuracy of the decision trees on this dataset worsened again and was equal to 0.78.

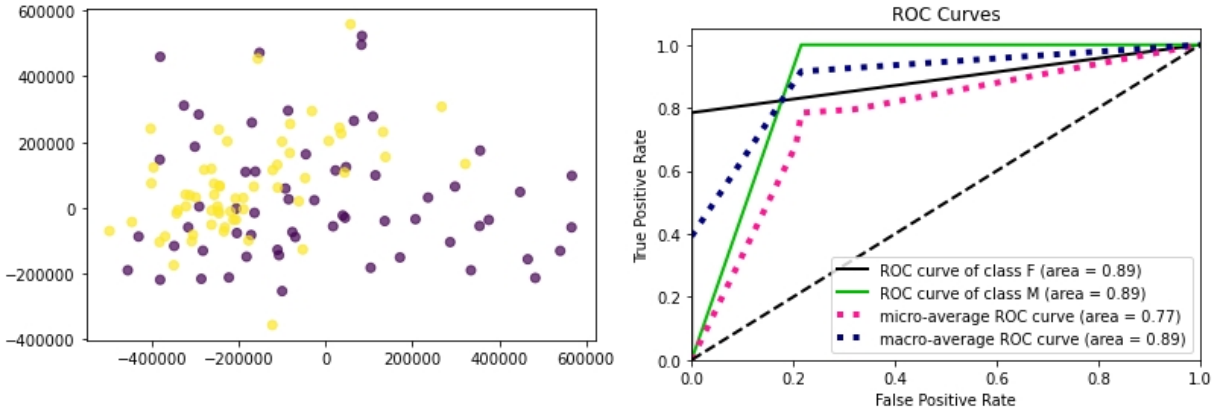


Figure 14: On the left: PCA allows to visualize the distribution of data points based on their "sex", balanced by cluster centroids. On the right: ROC curves on both classes of the decision tree classifier applied on the dataset after cluster centroids.

Moving on to oversampling, also in this case the first attempt was made through a random method. As a result, 842 "M" records and 842 "F" records were obtained. The decision tree classifier performed better on this dataset, reaching an accuracy score equal to 0.97.

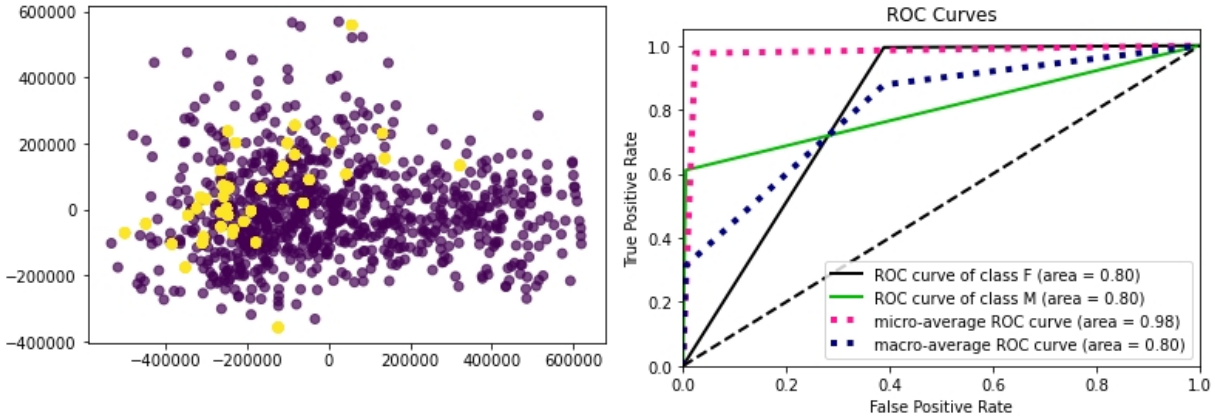


Figure 15: On the left: PCA allows to visualize the distribution of data points based on their "sex", balanced by random oversampling. On the right: ROC curves on both classes of the decision tree classifier applied on the dataset after random oversampling.

ADASYN, instead, returned a resampled dataset including 1204 "F" instances and 1191 "M" instances. The larger amount of records helped the decision tree classifier reach an accuracy score equal to 0.99.

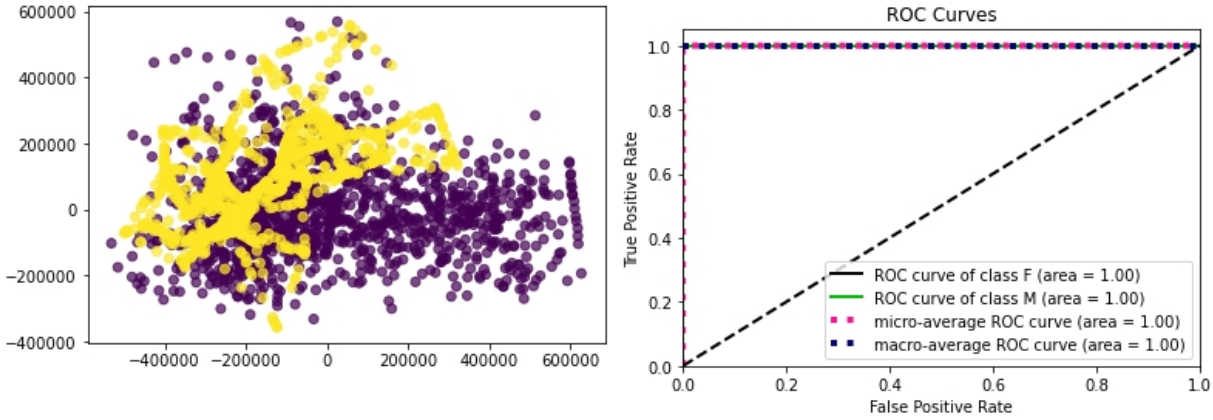


Figure 16: On the left: PCA allows to visualize the distribution of data points based on their "sex", balanced by ADASYN. On the right: ROC curves on both classes of the decision tree classifier applied on the dataset after ADASYN.

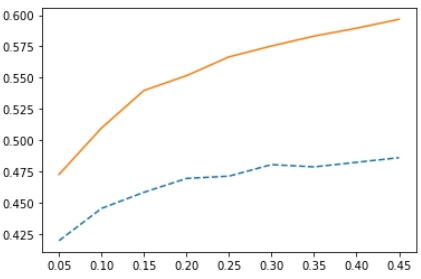
Generally speaking, it is better to use oversampling methods rather than undersampling ones in terms of performance, this is because in the second case we would have some information loss. Using the random undersampling and oversampling techniques the performance of the decision trees are quite similar, meaning that, even though the majority class was undersampled, there was enough information left in the dataset.
When applying more complex methods, oversampling performed better than undersampling. Indeed the Cluster Centroids method resulted in a better classification performance than both the random techniques, but still worse than the one obtained applying the Adasyn metod.

6. Advanced classification

The classification task that we chose is the prediction of the feature "emotion" given all the other features. What follows is the implementation, as well as the performance evaluation of different classification methods on this task.

6.1 Logistic regression

The logistic regression method was implemented firstly tuning its parameters regarding regularization terms. As for the solver, "saga" was chosen over "lbfgs" because of its capability of dealing with high-dimensional datasets. The best performance on the validation set was obtained by a model with "l2" as a penalty term and 1 as a regularization parameter. The regularization parameter C, however, needed some additional tuning, in order to address the overfitting issue. The graph below on the left shows how an increasing value of C causes an increasing difference between the accuracy on the train set (orange line) and the accuracy on the validation set (blue dashed line). Indeed, the parameter C was set equal to 0.05 in order to minimize this difference and the performance of the resulting classifier can be observed on the right below.



Colonna1	PRECISION	RECALL	F1-SCORE	SUPPORT
ANGRY	0.40	0.77	0.53	96
CALM	0.52	0.59	0.56	96
DISGUST	0.36	0.25	0.30	48
FEARFUL	0.48	0.31	0.38	96
HAPPY	0.36	0.31	0.34	96
NEUTRAL	1.00	0.02	0.04	48
SAD	0.44	0.36	0.40	96
SURPRISED	0.38	0.56	0.45	48
Accuracy			0.43	624
Macro avg	0.49	0.40	0.37	624
Weighted avg	0.47	0.43	0.40	624

Figure 17: On the left: comparison between accuracy on the train set (orange) and accuracy on the validation set (blue dashed), for different values of the regularization parameter c. On the right: performance evaluation of the resulting model applied on the test set.

Besides the numerical result, the performance can be better visualized on the confusion matrix and the ROC curve below.

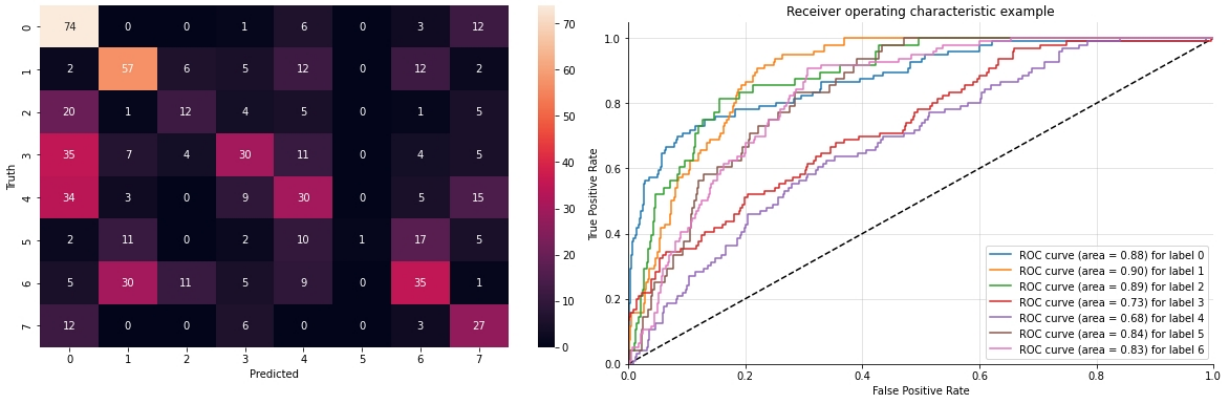
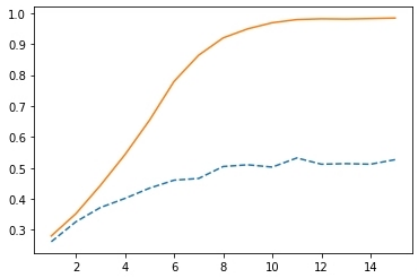


Figure 18: Performance evaluation of the logistic regression model on each class, where 'angry': 0,'calm': 1,'disgust': 2,'fearful': 3,'happy': 4,'neutral': 5,'sad': 6,'surprised': 7.

Overall this method performed quite well on some emotions, such as the "angry" records. However, the small precision suggests that the model predicted "angry" too often, especially when the ground truth was either "happy" or "neutral", as can be observed by the confusion matrix. These two emotions are indeed also those with the smallest areas under the related ROC curves, suggesting some troubles of the model to correctly predict them.

6.2 Random forest

While implementing Random Forest, the hyperparameter tuning regarded the number of trees in each forest, their maximum depth, the minimum number of samples required to split an internal node as well as the minimum number of samples required to be at a leaf node. The model with the most accurate performance on the validation set turned out to be a Random Forest made up of 300 trees, whose maximum depth is 10 and with 15 as the minimum number of samples to split an internal node. Then, we again looked for any overfitting issue and indeed found out that the maximum depth of the trees has a large influence in this case. On the left below, the difference in the accuracy between train and validation sets can be observed. The graph suggests that a value of 4 for the maximum depth guarantees both less overfitting and an acceptable accuracy. The table on the right summarises the performance evaluation of the forest made up of 300 trees, whose maximum depth is 4 and with 15 as the minimum number of samples to split an internal node.



	PRECISION	RECALL	F1-SCORE	SUPPORT
ANGRY	0.36	0.86	0.51	96
CALM	0.47	0.68	0.56	96
DISGUST	0.40	0.04	0.08	48
FEARFUL	0.47	0.21	0.29	96
HAPPY	0.26	0.17	0.20	96
NEUTRAL	0.00	0.00	0.00	48
SAD	0.27	0.27	0.27	96
SURPRISED	0.40	0.40	0.40	48
Accuracy			0.37	624
Macro avg	0.33	0.33	0.29	624
Weighted avg	0.34	0.37	0.32	624

Figure 19: On the left: comparison between accuracy on the train set (orange) and accuracy (y-axis) on the validation set (blue dashed), for different values of the maximum depth of the trees (x-axis). On the right: performance evaluation of the resulting model applied on the test set.

At this point, the importance of each feature in the chosen random forest was examined. The permutation importance of a feature indicates the decrease of the model accuracy when that feature is randomly shuffled. The graph on the right in figure 14 shows that many of the features related to the Mel-Frequency Cepstral Coefficients of the audio recordings are involved in this case. For example, permuting the feature "mfcc_kur_w3" can cause the accuracy to drop by around 0.01. This is also confirmed by the plot on the left of figure 14, which instead ranks the different attributes in order of importance. The first 15 are shown and their importance is computed as the total reduction of the criterion brought by that feature. There is a light mismatch between the features on the right and those on the left: this is because the importance of the former ones is computed considering their prediction ability, thus including the test set, while the importance of the latter ones in computed considering the training set only.

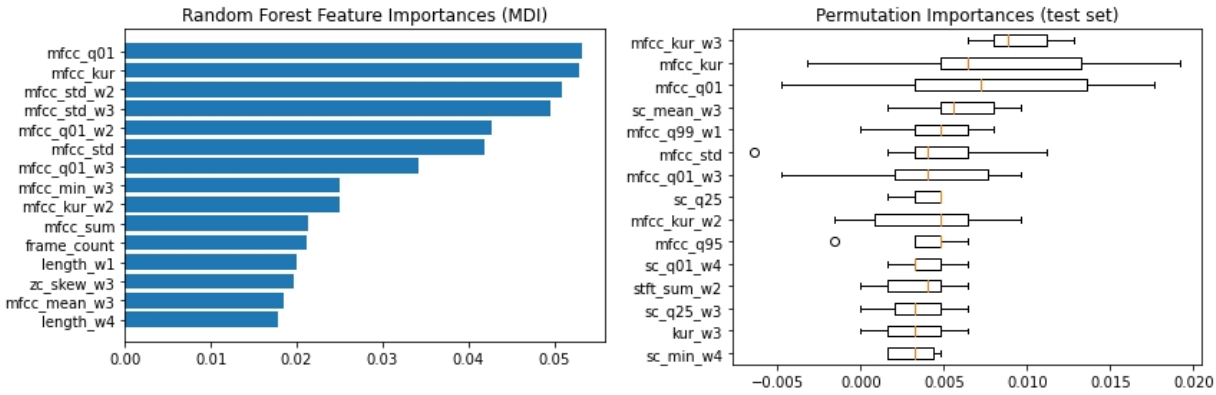


Figure 20: On the left: feature importance on the train set (first 15). On the right: permutation importance of some of the features on the test set (first 15).

Below, the confusion matrix and the ROC curve of the Random Forest classifier are reported.

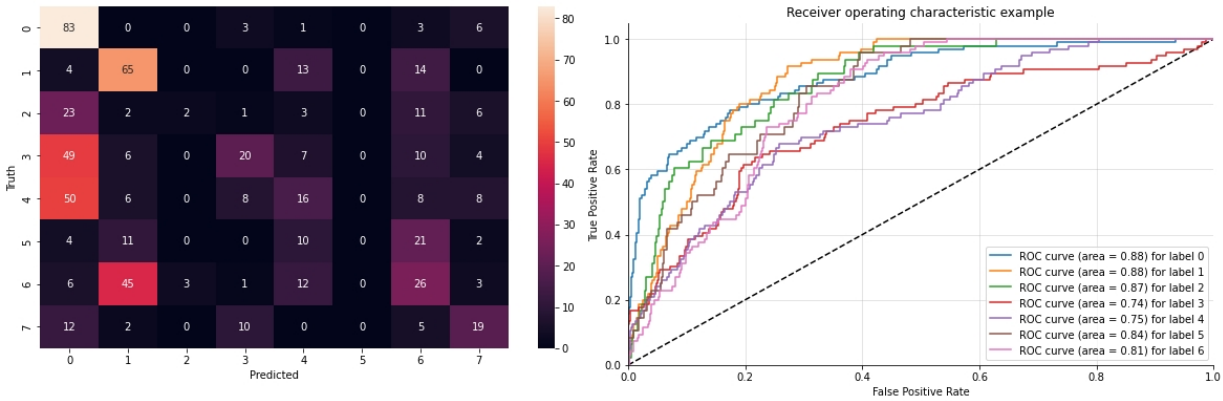


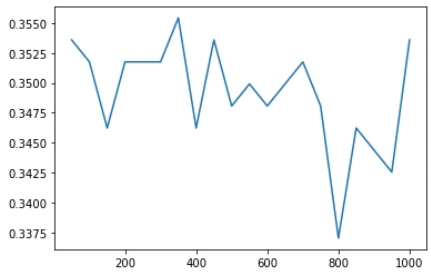
Figure 21: Performance evaluation of the random forest model on each class, where 'angry': 0,'calm': 1,'disgust': 2,'fearful': 3,'happy': 4,'neutral': 5,'sad': 6,'surprised': 7.

Random Forest performed slightly worse than Logistic regression: the overfitting issue was more evident with this model. The ROC curves show that "fearful" and "happy" are still the least correctly predicted emotions.

6.3 Bagging

At first, the hyperparameter tuning for Bagging regarded the choice of the base estimator. For this purpose, the two models previously seen were chosen: logistic regression and random forest, as well as the default estimator, which is

the decision tree. The decision tree seemed to be the one making Bagging perform the best on the validation set. Also in this case, their maximum depth played a crucial role in order to prevent overfitting. We were forced to tune this parameter since the accuracy in the train set would have reached 1.0 if it was ignored. A maximum depth of 3 turned out to be the best at reducing the difference between accuracy score on the train set and accuracy score on the validation set. Then, the parameter regarding the number of estimators was tuned too and the resulting plot is available on the left below. The performance evaluation of a bagging classifier fitting 350 decision trees can be observed on the right.



	PRECISION	RECALL	F1-SCORE	SUPPORT
ANGRY	0.35	0.84	0.49	96
CALM	0.52	0.48	0.50	96
DISGUST	0.00	0.00	0.00	48
FEARFUL	0.37	0.21	0.27	96
HAPPY	0.20	0.09	0.13	96
NEUTRAL	0.00	0.00	0.00	48
SAD	0.27	0.44	0.33	96
SURPRISED	0.33	0.33	0.33	48
Accuracy			0.34	624
Macro avg	0.26	0.30	0.26	624
Weighted avg	0.29	0.34	0.29	624

Figure 22: On the left: performance of the bagging classifier on the validation set with respect to different numbers of estimators, accuracy on the y-axis and number of estimators on the x-axis. On the right: performance evaluation of the bagging classifier with 350 decision trees as base estimators.

Based on this performance, the resulting confusion matrix and ROC curve can be observed below.

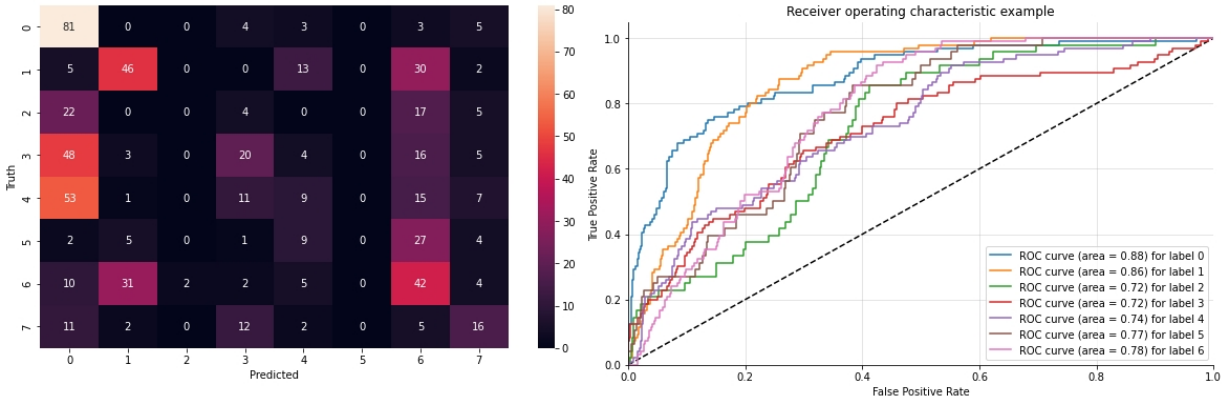


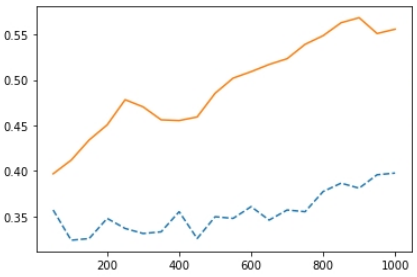
Figure 23: Performance evaluation of the bagging classifier on each class, where 'angry': 0,'calm': 1,'disgust': 2,'fearful': 3,'happy': 4,'neutral': 5,'sad': 6,'surprised': 7.

This model did not perform that well, especially with respect to "disgust" and "neutral", which are two of the three emotions that are represented by 48 records instead of 96. The imbalancing is not that large, but oversampling these three classes like we previously did for the "sex" feature would probably solve this issue.

6.4 AdaBoost

In the case of AdaBoost, the first step regarded the choice of the model to be used as a base estimator. The random forest estimators were those performing the best on the validation set. However, we decided to stick on the original formulation of AdaBoost, which uses weak learners only as base estimators: thus, the Decision Tree estimator was used, as by default. For what concerns the number of base estimators, performances of different values were compared. Again, we wondered whether the classifier had any overfitting issues. The graph on the left in figure

24 compares the performance of the classifier on the validation set with its performance on the train set, for different numbers of base estimators. The orange line represents the performances on the train set, while the dashed blue one represents the performances on the validation set. Their trends are quite different, although the performance on the validation set is always worse. In order to minimize this difference while keeping an acceptable accuracy, we decided to choose an AdaBoost classifier with 200 base estimators (50 base estimators seemed to be too few in our opinion). Other parameter needed to be tuned to address overfitting: the learning rate was the most effective at reducing it and at the end a value of 0.005 was chosen.



	PRECISION	RECALL	F1-SCORE	SUPPORT
ANGRY	0.59	0.65	0.62	96
CALM	0.39	0.27	0.32	96
DISGUST	0.61	0.23	0.33	48
FEARFUL	0.38	0.14	0.20	96
HAPPY	0.38	0.60	0.46	96
NEUTRAL	0.43	0.06	0.11	48
SAD	0.26	0.60	0.36	96
SURPRISED	0.58	0.15	0.23	48
Accuracy			0.38	624
Macro avg	0.45	0.34	0.33	624
Weighted avg	0.43	0.38	0.35	624

Figure 24: On the left: performance of the AdaBoost classifier with respect to different numbers of estimators, on the train set (orange) and on the validation set (dashed blue). Accuracy on the y-axis and number of estimators on the x-axis. On the right: performance evaluation of the AdaBoost classifier with 200 decision trees as base estimators, maximum depth equal to 3 and learning rate equal to 0.005.

The performance of the classifier on each emotion can be better visualized below.

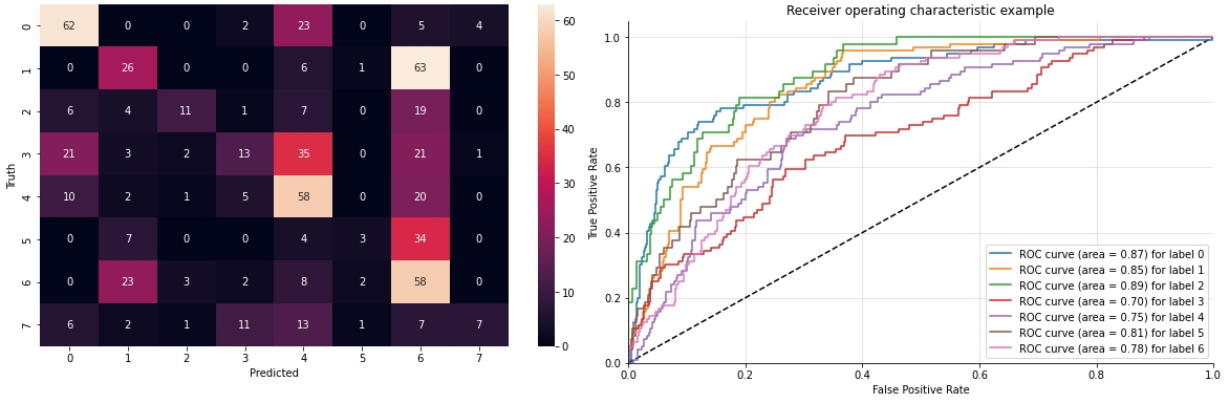


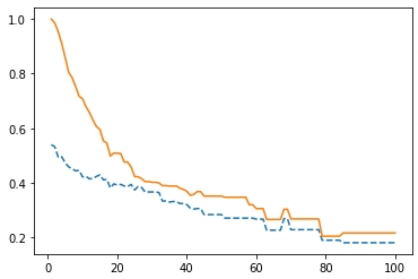
Figure 25: Performance evaluation of the AdaBoost classifier on each class, where 'angry': 0,'calm': 1,'disgust': 2,'fearful': 3,'happy': 4,'neutral': 5,'sad': 6,'surprised': 7.

Choosing random forest as the base estimator, instead of decision trees, would have helped get a higher accuracy score. On the other hand, a more complex model as base estimator, would have probably caused the final classifier to be even less robust to overfitting. We actually do not know whether or not there is some noise in this data, causing some records to be classified as one emotion instead of another one in the original dataset. If this is the case, this might be one reason why AdaBoost did not perform better than this. Another reason might regard the dimensionality curse that did not let AdaBoost work at its best.

6.5 XGBoost

The hyperparameter tuning in the case of XGBoost regarded first of all the maximum depth of the decision trees, then the learning rate and the regularization parameters lambda and gamma. The parameter gamma turned out to be

the most effective at reducing the overfitting issue of the model: its impact can indeed be observed on the left below. This graph suggested that the optimal value for gamma in order to minimize the difference of accuracy between the validation set (blue dashed line) and the training set (orange line) was around 25 and thus the performance evaluation of an XGBoost classifier with learning rate equal to 0.3, maximum depth of the trees equal to 4 and gamma parameter equal to 25 is on the right below.



	PRECISION	RECALL	F1-SCORE	SUPPORT
ANGRY	0.37	0.80	0.50	96
CALM	0.47	0.49	0.48	96
DISGUST	0.29	0.04	0.07	48
FEARFUL	0.40	0.17	0.24	96
HAPPY	0.36	0.09	0.15	96
NEUTRAL	0.00	0.00	0.00	48
SAD	0.27	0.54	0.36	96
SURPRISED	0.35	0.38	0.36	48
Accuracy			0.35	624
Macro avg	0.31	0.31	0.27	624
Weighted avg	0.34	0.35	0.30	624

Figure 26: On the left: gap between the accuracies computed on the training set (orange line) and on the validation set (blue dashed line), for increasing values of the gamma parameter, accuracy on the y-axis and gamma parameters on the x-axis. On the right: performance evaluation of the resulting XGBoost classifier.

Consequently, the confusion matrix and the ROC curve of the model were computed.

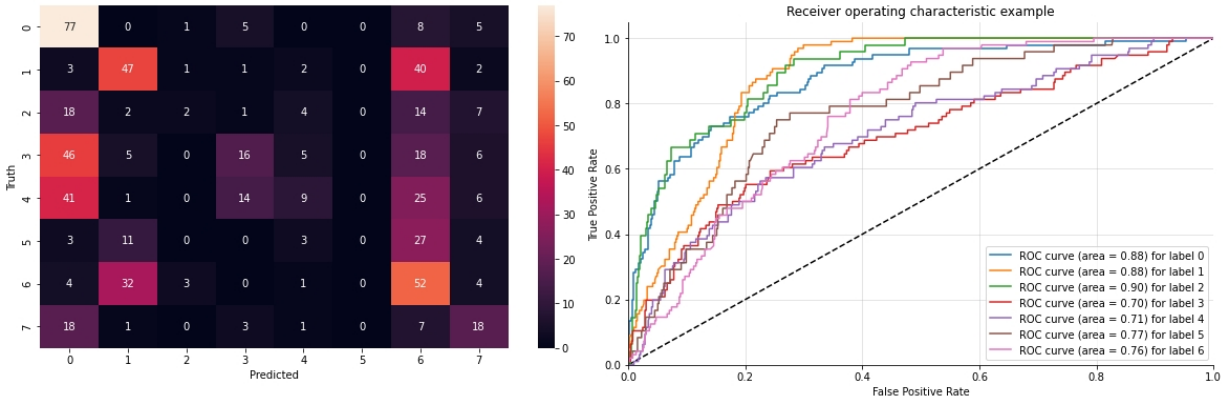


Figure 27: Performance evaluation of the XGBoost classifier on each class, where 'angry': 0,'calm': 1,'disgust': 2,'fearful': 3,'happy': 4,'neutral': 5,'sad': 6,'surprised': 7.

The XGBoost classifer also had some issues identifying one of the least represented classes: "neutral" in this case was never predicted correctly, while "neutral" was twice only. Again, oversampling these classes would have probably helped improve the performance of XGBoost at predicting them.

6.6 Neural network

The activation function, the initial learning rate, the number of layers, as well as the number of nodes in each of these layers, were the crucial aspects to observe and tune. Also, in order to avoid overfitting, the tuning was implemented with an early stopping condition, which resulted much more effective than the any l1 or l2 regularization. As a result, the neural network performing the best on the validation set was one with two hidden layers made up of 100 nodes each, relu as activation function and initial learning rate equal to 0.001. Its performance evaluation can be observed below.

	PRECISION	RECALL	F1-SCORE	SUPPORT
ANGRY	0.47	0.77	0.58	96
CALM	0.54	0.48	0.51	96
DISGUST	0.29	0.42	0.34	48
FEARFUL	0.39	0.33	0.36	96
HAPPY	0.39	0.32	0.35	96
NEUTRAL	0.57	0.25	0.35	48
SAD	0.49	0.25	0.33	96
SURPRISED	0.40	0.69	0.51	48
Accuracy			0.44	624
Macro avg	0.44	0.44	0.42	624
Weighted avg	0.45	0.44	0.42	624

Figure 28: Performance evaluation on the test set of the neural network performing the best on the validation set.

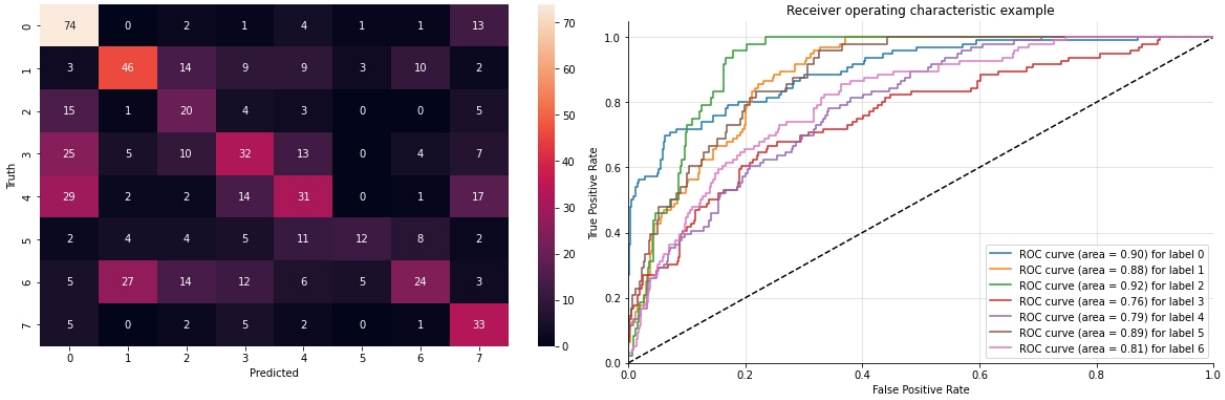
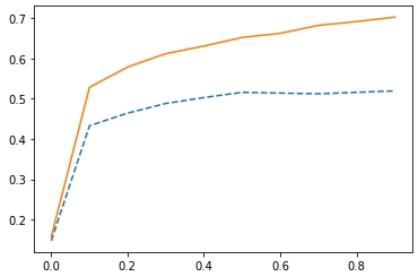


Figure 29: Performance evaluation of the neural network classifier on each class, where 'angry': 0,'calm': 1,'disgust': 2,'fearful': 3,'happy': 4,'neutral': 5,'sad': 6,'surprised': 7.

The neural network classifier performed quite good in comparison with the models previously implemented. Probably the early stopping condition is what helped it reach both a good level of accuracy on the test set and a low level of overfitting on the validation set. Also, any classification task on this dataset can be basically defined as sound recognition, which falls in the field of deep learning. For this reason, the neural network would be perhaps the most suitable classifier for this task and this would explain its largest accuracy.

6.7 Support vector machine

As expected, also in this case overfitting was an issue to solve. The parameter C, inversely proportional to the l2 regularization strength, helped for this purpose. As for the kernel, the polynomial would have performed better than the linear one, if overfitting was not considered. Thus, the graph on the left below shows how the parameter C of a linear SVM model influenced the accuracy on both the validation (blue dashed line) and the training set (orange line). As a result, a value of C equal to 0.2 was chosen and the performance evaluation of the corresponding model is on the right below. Also, the gamma parameter was tuned, but there was no effect on the resulting accuracy. A low gamma means that the model is only considering the closest data points in order to build the hyperplane separating them, while larger values for gamma make the model consider also further data points. The fact that gamma had no effect on the accuracy on the validation set probably means that there are too many dimensions and thus the distance between the data points and the hyperplane is not that meaningful.



	PRECISION	RECALL	F1-SCORE	SUPPORT
ANGRY	0.43	0.77	0.55	96
CALM	0.62	0.50	0.55	96
DISGUST	0.35	0.44	0.39	48
FEARFUL	0.44	0.32	0.37	96
HAPPY	0.33	0.26	0.29	96
NEUTRAL	0.48	0.21	0.29	48
SAD	0.47	0.42	0.44	96
SURPRISED	0.44	0.56	0.50	48
Accuracy			0.44	624
Macro avg	0.44	0.43	0.42	624
Weighted avg	0.45	0.44	0.43	624

Figure 30: On the left: gap between the accuracies computed on the training set (orange line) and on the validation set (blue dashed line), for increasing values of the C parameter, accuracy on the y-axis and C parameters on the x-axis. On the right: performance evaluation of the resulting SVM classifier.

Based on this performance, the confusion matrix and the ROC curves of this model are available below.

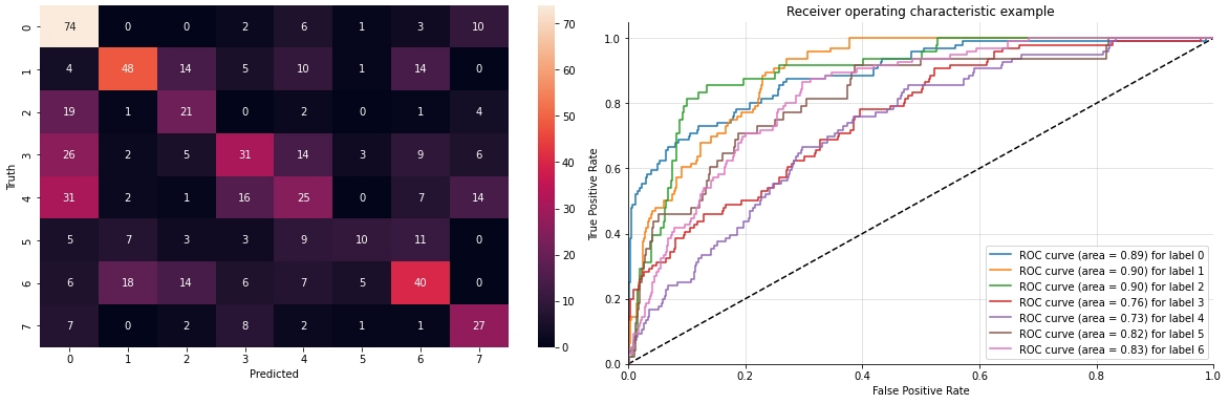


Figure 31: Performance evaluation of the support vector machine classifier on each class, where 'angry': 0,'calm': 1,'disgust': 2,'fearful': 3,'happy': 4,'neutral': 5,'sad': 6,'surprised': 7.

SVM and neural networks had the best performances on the test set. SVM had some major issues at recognizing the fearful and happy recordings, exactly what happened with all the classifiers previously implemented too.
PCA allows to visualize the data points identified as support vectors by the model, i.e. those that are closest to the decision boundary or hyperplane drawn by the classifier.

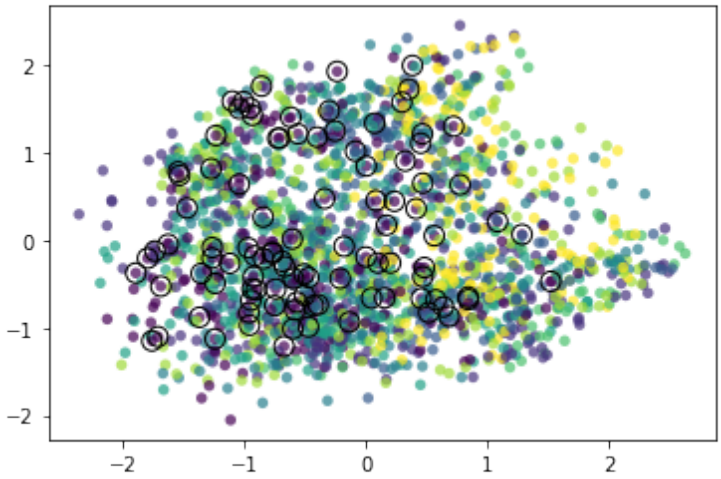


Figure 32: Data points visualized in a two-dimensional space through PCA. Colours based on their class and black border identifying the support vectors.

7. Advanced regression

For the advanced regression task, two regressors were implemented in order to predict the attribute "frame count". Specifically, the models chosen were random forest and gradient boosting.
The random forest regressor was implemented with 100 estimators, 2 as the minimum number of samples in a node to perform a split and 1 as minimum number of samples in a node to be considered as a leaf node.
The gradient boosting regressor was implemented with same parameters chosen for the random forest regressor, as well as a learning rate equal to 0.1.
The performance evaluation of the two models on the test data is available below.

	MSE	RMSE	R^2	Adj. R^2
Random Forest	1.24e-05	0.003	0.99	0.99
Gradient Boosting	9.16e-06	0.003	0.99	0.99

Figure 33: Performance evaluation of random forest and gradient boosting classifiers.

This table shows that both MSE and the RMSe of the regressors are very small. However, since these two metrics are hard to interpret, a look at R-squared and adjusted R-squared is needed. The fact that both of them are very close to 1 indicates that the two regressors performed very well on the test data.

8. Time series

After importing the dataset, some experiments on the time series were made in order to see the best transformation and approximation to apply. We noticed how about the first and last 25000 values of each time series were zeros, and therefore useless, do we decided to cut them.
A comparison among different approximation methods on the same piece of time series can be observed below.

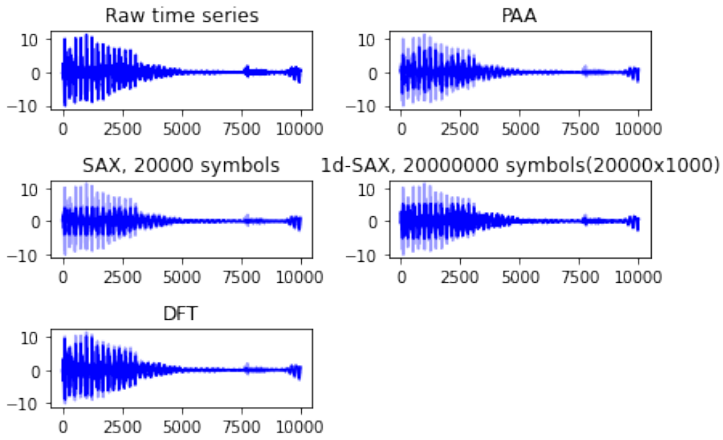


Figure 34: Comparison among the raw time series and PAA, SAX, 1d-SAX and DFT approximation methods.

However, some changes were applied to the time series after this visualization. First of all, the "tails" were cut, such that every time series would be 80.000 timestamps long. Downsampling allowed us to decimate every time series with a factor of 20. Also trend removal, amplitude scaling and smoothing were applied. Amplitude scaling in particular allowed the time series to have zero mean and unit standard deviation, while smoothing was crucial in order to remove the noise.
As for the approximation, PAA was chosen. From now on, we would work with 2452 time series approximated to 300 segments each through PAA. This method was chosen because it provided a good compromise between a useful approximation and not too much loss of data.

8.1 Time series: clustering

The clustering algorithms applied are K-Means and hierarchical clustering. The first K-Means implementation was set up in order to look for 8 clusters, one for each emotion, with euclidean distance. The following result was obtained, with a Silhouette coefficient equal to 0,86 and SSE equal to 12,8.

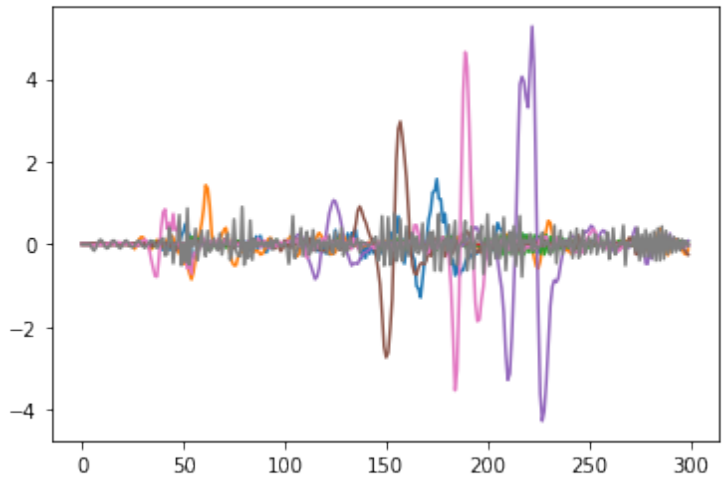


Figure 35: Clusters found by TimeSeries KMeans with k=8.

A comparison of the SSE and Silhouette trends when the number of clusters varies from 2 to 8 can be observed below.

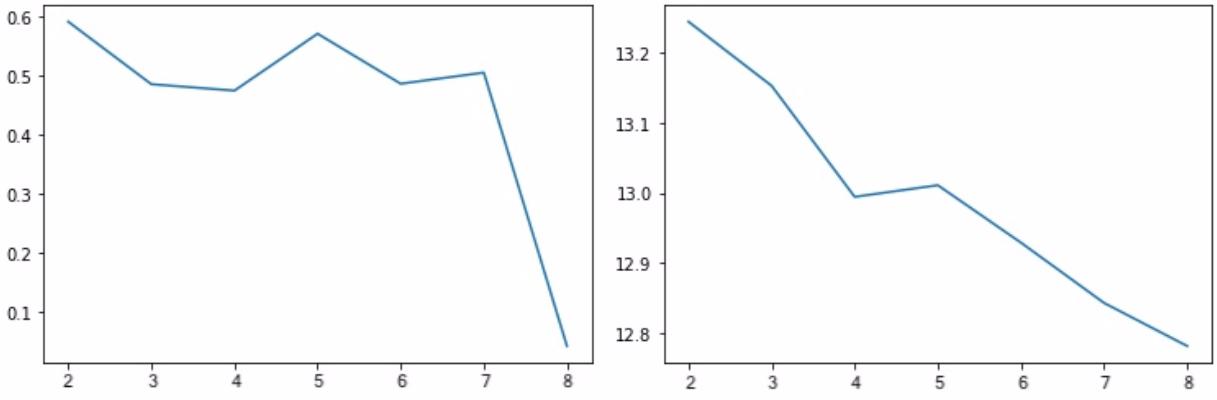


Figure 36: On the left: Silhouette trend for different K-Means implementations with k between 2 and 8. On the right: SSE trend for different K-Means implementations with k between 2 and 8.

Hierarchical clustering was implemented with euclidean distance and the criterion chosen to merge clusters was the Ward linkage. Based on where the resulting dendrogram is cut, hierarchical clustering returns different numbers of clusters. These were compared through the plot below, showing the Silhouette score trend.

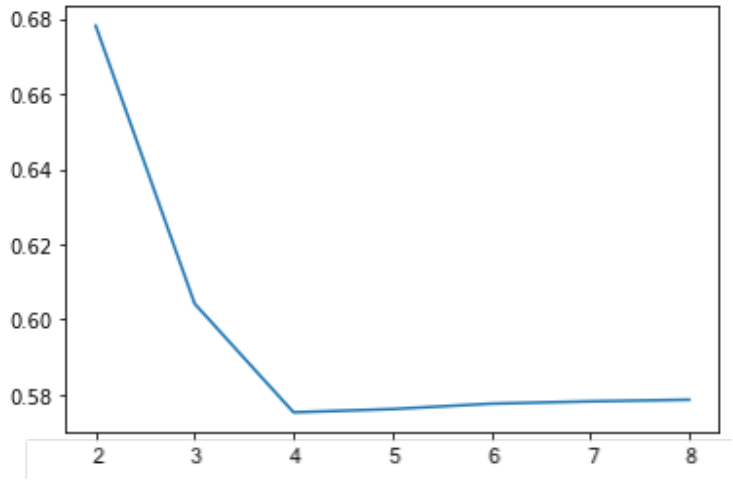


Figure 37: Silhouette trend considering different amounts of clusters on the dendrogram returned by the implementation of hierarchical clustering.

From the analysis above, the best results are obtained when Kmeans is applied with n_clusters = 6, because this value provides a good compromise between a Silhouette score closer to 1 and and a smaller SSE, and when the hierarchical clustering takes 2 clusters from the dendrogram, because this is the case returning the highest Silhouette score. Thus these two cases are considered for the following visualizations through dimensionality reduction.
The techniques implemented to visualize the clusters are PCA and t-SNE.
The representation below shows the 6 clusters found through K-Means. Three dimensions obviously are not enough to observe a meaningful result. However, a notable difference between PCA and t-SNE can be observed: the latter provides a more compact visualization, because it prioritizes the local structure of the time series, while the first prioritizes the components that explain the most variance in the data, regardless of the class labels or specific local patterns.

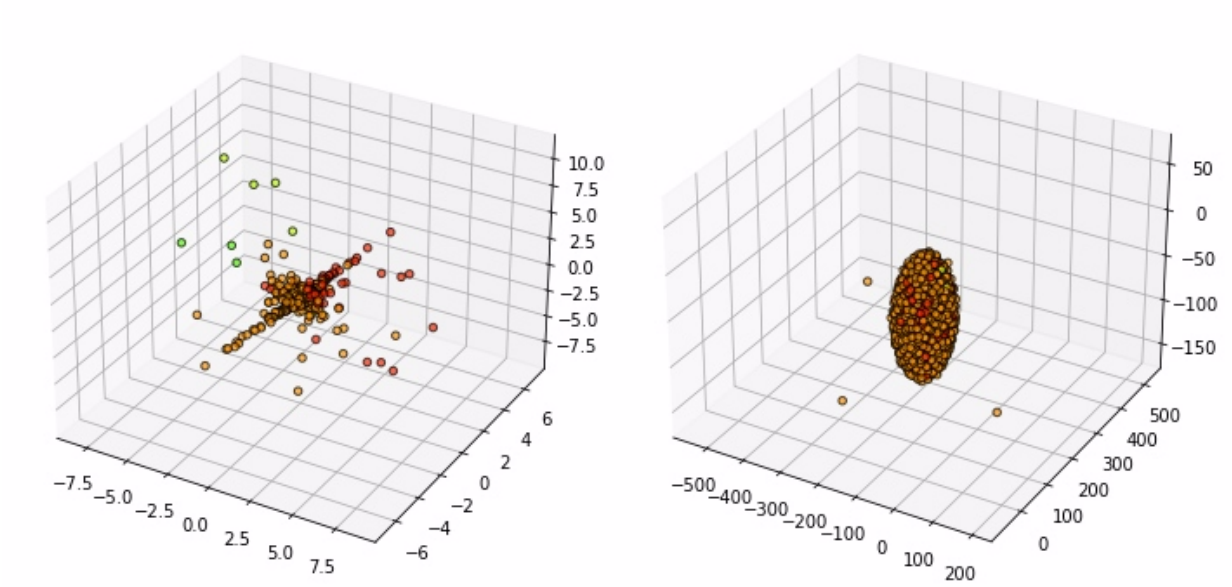


Figure 38: On the left: PCA representation of K-Means with n_clusters=6. On the right: t-SNE representation of the same algorithm.

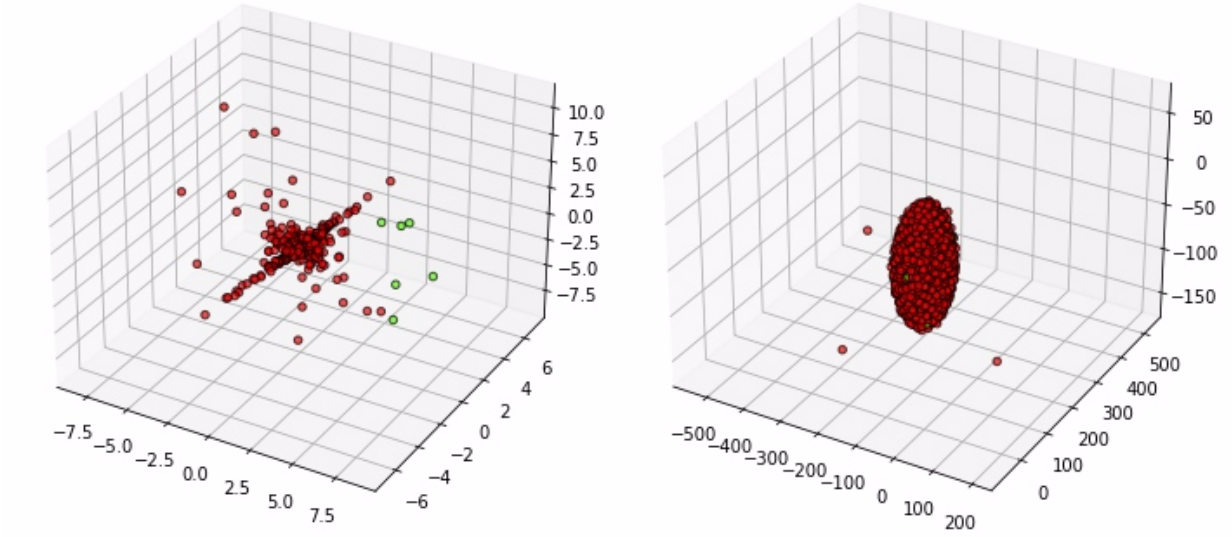


Figure 39: On the left: PCA representation of hierarchical clustering returning 2 clusters. On the right: t-SNE representation of the same algorithm.

Another remarkable aspect regards the fact that the t-SNE technique seems to detect some outliers among the time series.

8.2 Time series: motifs and discords

Motifs and discords were looked for on the time series below, which is one of the cluster centers previously returned by K-Means. Its matrix profile was computed through the STOMP algorithm with a window size equal to 6.

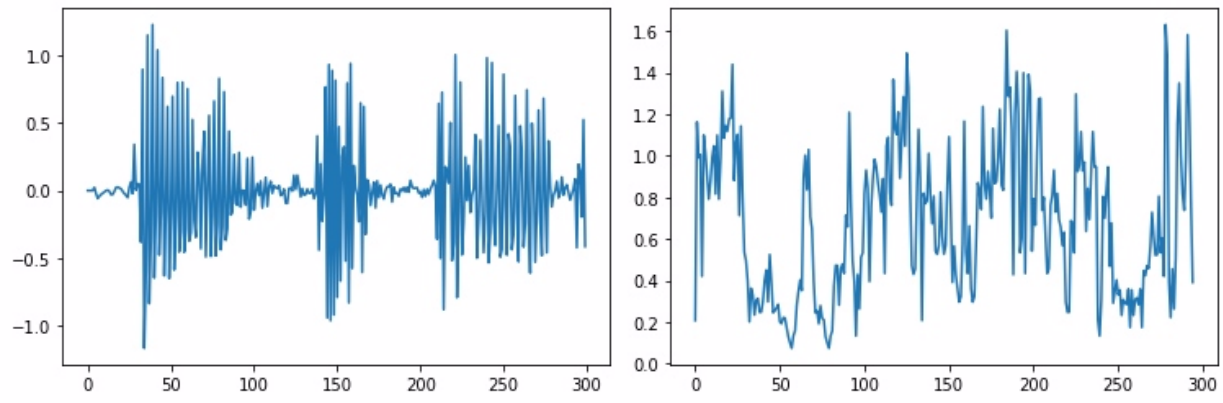


Figure 40: On the left: time series randomly chosen from the cluster centers previously returned by K-Means. On the right: corresponding matrix profile.

Considering the matrix profile above, we looked for the top 5 motifs and discords. As for motifs, the following result was obtained.

INDEXES	DISTANCES
[57, 79]	0.074
[95, 239]	0.133
[250, 257, 264]	0.174
[36, 51, 73]	0.192
[32, 47, 281]	0.201

Figure 41: Indexes representing the motif starting locations and related distances for each of them.

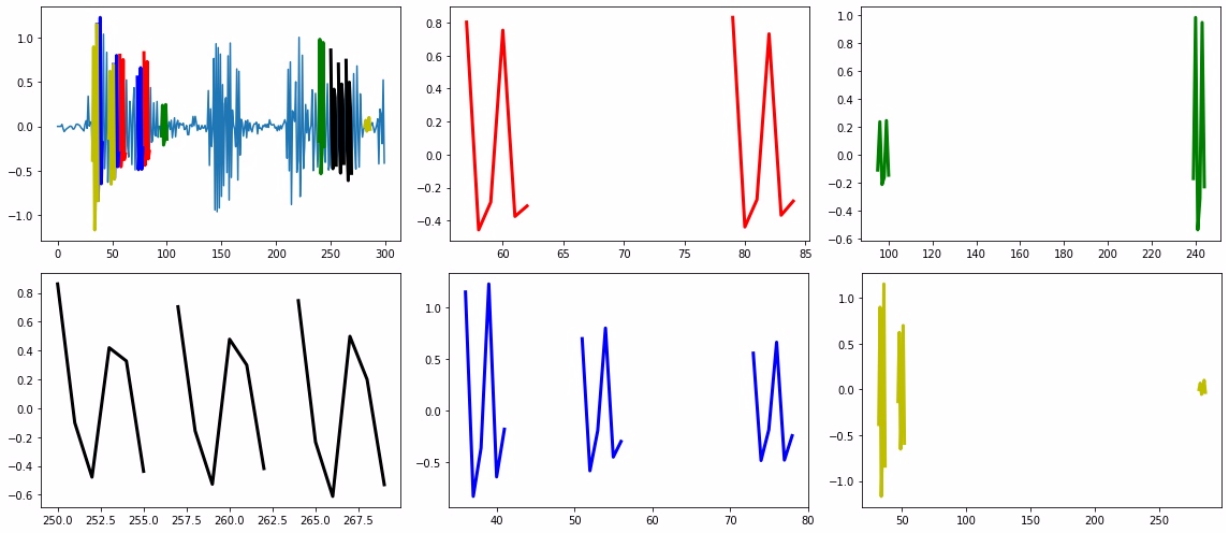


Figure 42: Top 5 motifs found on the time series on the left above.

Then, the top 5 discords were found and they can be observed below on the original time series.

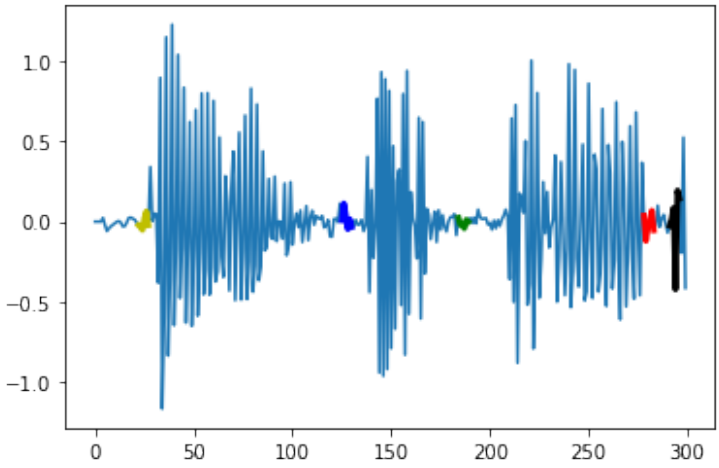


Figure 43: Top 5 discords found on the time series.

8.3 Time series: classification

The classification task we chose regards labelling time series based on the underlying emotion.
The involved classifier is KNN and it was implemented twice: firstly with euclidean distance and then with DTW distance. The resulting accuracy scores on the validation set with varying number of neighbors can be observed below.

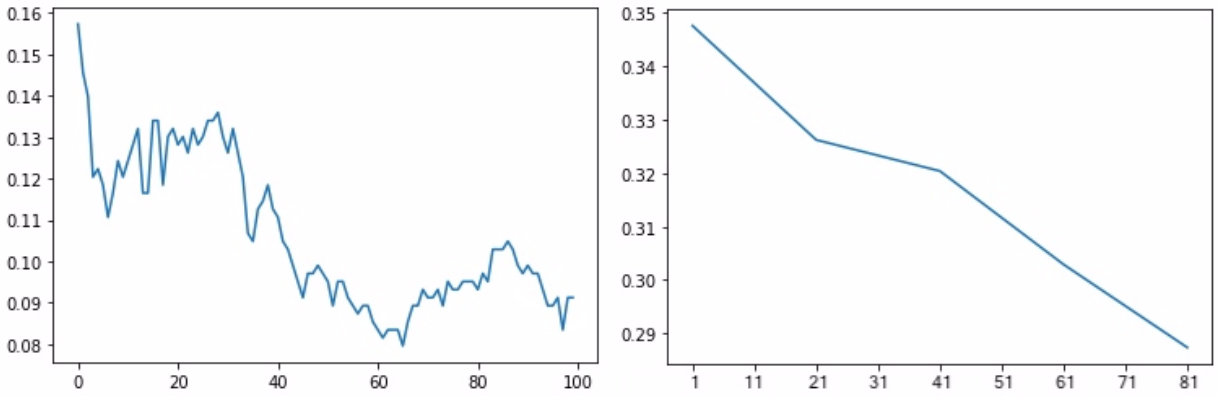


Figure 44: On the left: accuracy score on the validation set of KNN with euclidean distance. On the right: accuracy score on the validation set of KNN with DTW sakoechiba distance.

The implementation of the algorithm based on the DTW distance is much more expensive than the one based on the Euclidean distance: that is why KNN with DTW distance was not run with every number of neighbors between 1 and 100, but instead taking steps equal to 10 in this interval. However, as shown by the plots, the greater cost of DTW leads to better results in terms of accuracy on the validation set.
A problem with the complexity of the algorithm was faced also for the classification task based on shapelets. In particular, the extraction of shapelets took too much time with the current dataset including 2452 time series approximated through 300 timestamps each through PAA. For this reason, not only were the number of segments returned by PAA decreased to 80, but also a sample of the original dataset was isolated, such that every emotion could be represented by 10 records, leading to a subset of 80 time series. Then at most 8 shapelets where extracted from this subset and a shapelet transformation was applied. However, as expected, the classifier (decision tree) performed very poorly on this subset, with an accuracy score around 0.10. This was caused by the fact that the dataset was simplified too much in order to provide meaningful results. The algorithm still managed to retrieve 8 shapelets and their information is available below.

INDEXES	SHAPELETS
[4, 17, 20]	[0.10413839, -0.01052623, -0.05194426]
[14, 61, 68]	[-0.04024367, 0.04795 , -0.05862334, -0.00886426, 0.10547476, -0.0178135 , -0.0915329]
[54, 66, 71]	[-0.01419064, 0.00012191, -0.01002786, -0.01559149, -0.0157687]
[27, 31, 34]	[0.00358797, 0.00418976, -0.01757585]
[53, 15, 18]	[-0.01927395, -0.09845714, 0.00878557]
[31, 82, 85]	[-0.09612067, -0.04009487, 0.00798015]
[48, 6, 15]	[-0.01879896, -0.03162762, 0.07900642, -0.09744167, 0.09853616, -0.07504842, 0.03429384, -0.08814384, 0.09089559]
[32, 16, 25]	[0.04821035, -0.03106256, 0.02889642, -0.03452001, 0.04925118, -0.1457838 , 0.10719372, 0.013422 , -0.03991067]

Figure 45: Indices and selected shapelets. As for the indices, the first value corresponds to the indices of the samples, the second value corresponds to the starting indices (included) of the shapelets and the third value consists of the ending indices (excluded) of the shapelets.

A CNN classifier was implemented on the original dataset including 2452 time series approximated through 300 timestamps each through PAA. The challenge in this case regarded avoiding making the CNN too complex in order to avoid overfitting, but still reaching an acceptable level of accuracy on the test set. The amount of epochs implemented in the training phase influenced the final accuracy score too. Some attempts with different topologies were made and the one providing the highest accuracy level on the test set involved a CNN with 4 hidden layers, made up of 32, 64, 128 and 256 nodes respectively, all of them including a ReLU activation function, while the output layes included a sigmoid activation function. The optimizer chosen was ADAM. The related classification report can be observed below.

	PRECISION	RECALL	F1-SCORE	SUPPORT
ANGRY	0.47	0.13	0.21	113
CALM	0.36	0.78	0.49	113
DISGUST	0.09	0.02	0.03	58
FEARFUL	0.30	0.25	0.27	113
HAPPY	0.29	0.28	0.28	113
NEUTRAL	0.24	0.62	0.34	56
SAD	0.41	0.23	0.29	113
SURPRISED	0.39	0.21	0.27	57
Accuracy			0.32	736
Macro avg	0.32	0.32	0.27	736
Weighted avg	0.33	0.32	0.29	736

Figure 46: Performance evaluation of the CNN classifier.

8.4 Time series: explainability

In this section, what is "behind the scenes" of a Random Forest classifier was explained at first globally through a decision tree and then locally through the LIME explainer. A Random Forest classifier including 20 trees was implemented on the dataset, leading to quite good accuracy score on the test set: 0,43. The overall behaviour of these 20 trees could be summed up by a decision tree as a global explainer and a portion of it can be observed below.

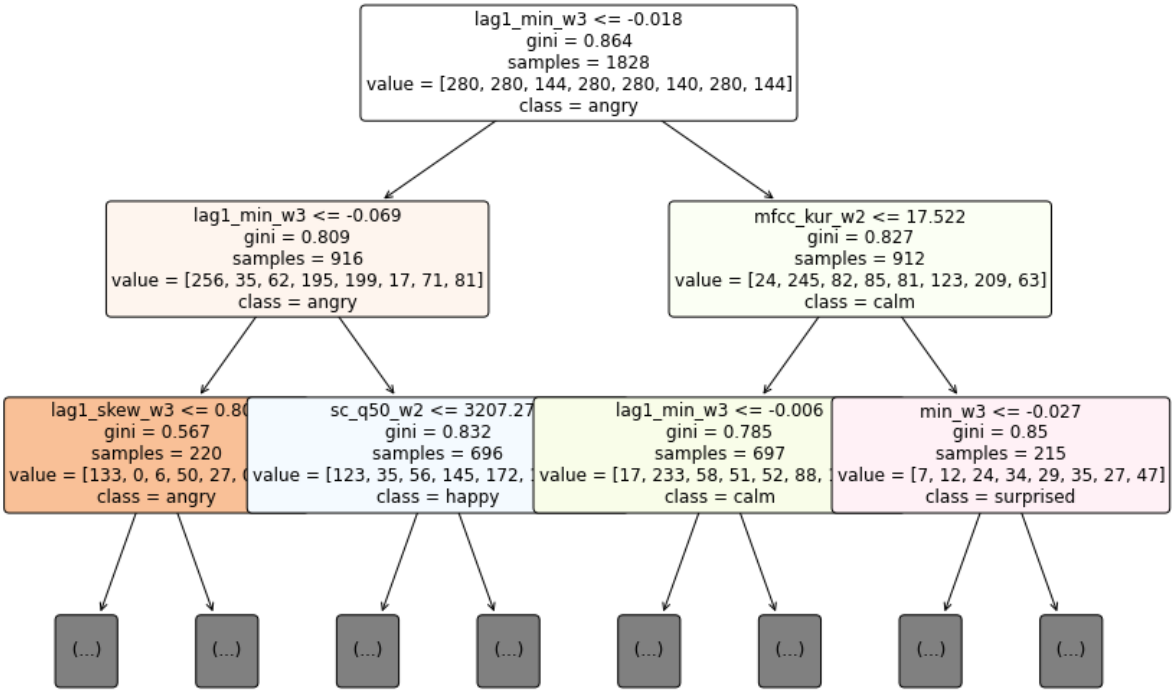


Figure 47: Portion of the decision tree explaining the Random Forest classifier from a global point of view.

The LIME explainer was then implemented in order to visualize how important each feature was for the final prediction of the Random Forest classifier on every instance of the dataset. An example on a random instance is shown below.

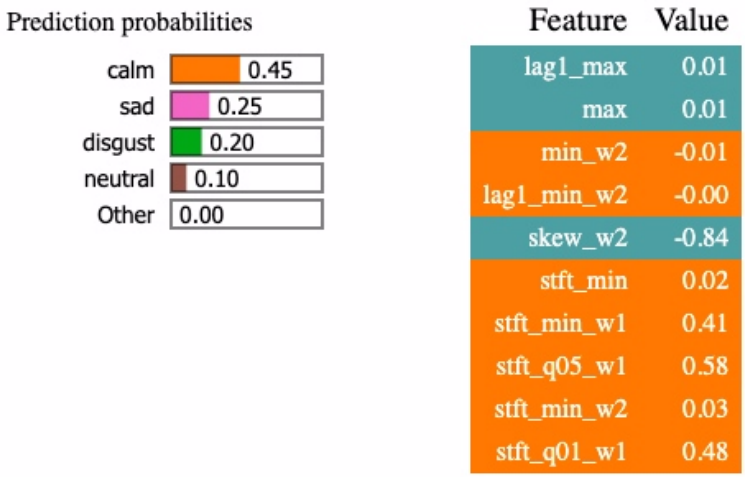


Figure 48: On the left: prediction probabilities for the selected record, showing that the class "calm" is the one suggested by the classifier. On the right: Feature importance on the final prediction.

The example above regards an instance which was classified as “calm” and some of the features which contributed the most to this prediction are “stft_min_w1”, “stft_q05_w1” and “stft_q01_w1”. In general, considering the subset of features returned by the table on the right above, the blue rows indicate those features that contributed to let the classifier predict another class, while orange rows show the features that contributed to the final prediction.